Lecture Notes in Artificial Intelligence 5211

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Walter Daelemans Bart Goethals
Katharina Morik (Eds.)

Machine Learning and Knowledge Discovery in Databases

European Conference, ECML PKDD 2008
Antwerp, Belgium, September 15-19, 2008
Proceedings, Part I

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Walter Daelemans
University of Antwerp, Department of Linguistics
CNTS Language Technology Group
Prinsstraat 13, L-203, 2000 Antwerp, Belgium
E-mail: walter.daelemans@ua.ac.be

Bart Goethals
University of Antwerp
Mathematics and Computer Science Department
Middelheimlaan 1, 2020 Antwerp, Belgium
E-mail: bart.goethals@ua.ac.be

Katharina Morik
Technische Universität Dortmund
Computer Science VIII, Artificial Intelligence Unit
44221 Dortmund, Germany
E-mail: katharina.morik@tu-dortmund.de

Library of Congress Control Number: 2008934755

CR Subject Classification (1998): I.2, H.2.8, H.2, H.3, G.3, J.1, I.7, F.2.2, F.4.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-87478-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-87478-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12521968 06/3180 5 4 3 2 1 0

Preface

When in 1986 Yves Kodratoff started the European Working Session on Learning at Orsay, France, it could not be foreseen that the conference would grow year by year and become the premier European conference of the field, attracting submissions from all over the world. The first European Conference on Principles of Data Mining and Knowledge Discovery was organized by Henryk Jan Komorowski and Jan Zytkow in 1997 in Trondheim, Norway. Since 2001 the two conferences have been collocated, offering participants from both areas the opportunity to listen to each other's talks. This year, the integration has moved even further. Instead of first splitting the field according to ECML or PKDD topics, we flattened the structure of the field to a single set of topics. For each of the topics, experts were invited to the Program Committee. Submitted papers were gathered into one collection and characterized according to their topics. The reviewers were then asked to bid on all papers, regardless of the conference. This allowed us to allocate papers more precisely.

The hierarchical reviewing process as introduced in 2005 was continued. We nominated 30 Area Chairs, each supervising the reviews and discussions of about 17 papers. In addition, 307 reviewers completed the Program Committee. Many thanks to all of them! It was a considerable effort for the reviewers to carefully review the papers, some providing us with additional reviews even at short notice. Based on their reviews and internal discussions, which were concluded by the recommendations of the Area Chairs, we could manage the final selection for the program. We received 521 submissions, of which 100 were presented at the conferences, giving us an acceptance rate of 20%. This high selectivity means, on the one hand, that some good papers could not make it into the conference program. On the other hand, it supports the traditionally high standards of the joint conference. We thank the authors from all over the world for submitting their contributions!

Following the tradition, the first and the last day of the joint conference were dedicated to workshops and tutorials. ECML PKDD 2008 offered 8 tutorials and 11 workshops. We thank the Workshop and Tutorial Chairs Siegfried Nijssen and Arno Siebes for their excellent selection. The discovery challenge is also a tradition of ECML PKDD that we continued. We are grateful to Andreas Hotho and his colleagues from the Bibsonomy project for organizing the discovery challenge of this year. The results were presented at the Web 2.0 Mining Workshop.

One of the pleasures of chairing a conference is the opportunity to invite colleagues whose work we esteem highly. We are grateful to Françoise Fogelman Soulié (KXEN) for opening the industrial track, Yoav Freund (University of California, San Diego), Anil K. Jain (Michigan State University), Ray Mooney (University of Texas at Austin), and Raghu Ramakrishnan (Yahoo! Research) for accepting our invitation to present recent work at the conference.

Some novelties were introduced to the joint conference this year.

First, there was no distinction into long and short papers. Instead, paper length was raised to 16 pages for all submissions.

Second, 14 papers were selected for publication in Springer Journals Seven papers were published in the *Machine Learning Journal* 72:3 (September 2008), and 7 papers were published in the *Data Mining and Knowledge Discovery Journal* 17:1 (August 2008). This LNAI volume includes the abstracts of these papers, each containing a reference to the respective full journal contribution. At the conference, participants received the proceedings, the tutorial notes and workshop proceedings on a USB memory stick.

Third, all papers were additionally allowed to be presented as posters. Since the number of participants has become larger, questions and discussions after a talk are no longer possible for all those interested. Introducing poster presentations for all accepted papers allows for more detailed discussions. Hence, we did not reserve this opportunity for a minority of papers and it was not an alternative to an oral presentation.

Fourth, a special demonstration session was held that is intended to be a forum for showcasing the state of the art in machine learning and knowledge discovery software. The focus lies on innovative prototype implementations in machine learning and data analysis. The demo descriptions are included in the proceedings. We thank Christian Borgelt for reviewing the submitted demos. Finally, for the first time, the conference took place in Belgium!

September 2008

Walter Daelemans
Bart Goethals
Katharina Morik

Organization

Program Chairs

Walter Daelemans	University of Antwerp, Belgium
Bart Goethals	University of Antwerp, Belgium
Katharina Morik	Technische Universität Dortmund, Germany

Workshop and Tutorial Chairs

Siegfried Nijssen	Utrecht University, the Netherlands
Arno Siebes	Katholieke Universiteit Leuven, Belgium

Demo Chair

Christian Borgelt	European Centre for Soft Computing, Spain

Publicity Chairs

Hendrik Blockeel	Katholieke Universiteit Leuven, Belgium
Pierre Dupont	Université catholique de Louvain, Belgium
Jef Wijsen	University of Mons-Hainaut, Belgium

Steering Committee

Pavel Brazdil
Rui Camacho
Johannes Fürnkranz
João Gama
Alípio Jorge
Joost N. Kok
Jacek Koronacki
Ramon Lopez de Mantaras
Stan Matwin
Dunja Mladenic
Tobias Scheffer
Andrzej Skowron
Myra Spiliopoulou

Local Organization Team

Boris Cule
Guy De Pauw
Calin Garboni
Joris Gillis
Iris Hendrickx
Wim Le Page
Kim Luyckx
Michael Mampaey
Roser Morante
Adriana Prado
Koen Smets
Vincent Van Asch
Koen Van Leemput

Area Chairs

Dimitris Achlioptas
Jean-François Boulicaut
Toon Calders
James Cussens
Tapio Elomaa
Johannes Fürnkranz
Szymon Jaroszewicz
Hillol Kargupta
Heikki Mannila
Srinivasan Parthasarathy
Lorenza Saitta
Martin Scholz
John Shawe-Taylor
Maarten van Someren
Stefan Wrobel
Roberto Bayardo
Wray Buntine
Padraig Cunningham
Ian Davidson
Martin Ester
Aris Gionis
Thorsten Joachims
Eamonn Keogh
Taneli Mielikainen
Jian Pei
Tobias Scheffer
Michèle Sébag
Antal van den Bosch
Michalis Vazirgiannis
Osmar Zaiane

Program Committee

Niall Adams
Alekh Agarwal
Charu Aggarwal
Gagan Agrawal
David Aha
Florence d'Alche-Buc
Enrique Alfonseca
Aris Anagnostopoulos
Annalisa Appice
Thierry Artieres
Yonatan Aumann
Paolo Avesani
Ricardo Baeza-Yates
Jose Balcazar
Aharon Bar Hillel
Ron Bekkerman
Bettina Berendt
Michael Berthold
Steffen Bickel
Ella Bingham
Gilles Blanchard
Hendrik Blockeel
Axel Blumenstock
Francesco Bonchi
Christian Borgelt
Karsten Borgwardt
Henrik Bostrom
Thorsten Brants
Ivan Bratko
Pavel Brazdil
Ulf Brefeld
Bjorn Bringmann
Greg Buehrer
Rui Camacho
Stephane Canu
Carlos Castillo
Deepayan Chakrabarti
Kamalika Chaudhuri
Nitesh Chawla
Sanjay Chawla
David Cheung
Alok Choudhary
Wei Chu
Alexander Clark
Chris Clifton
Ira Cohen
Antoine Cornuejols
Bruno Cremilleux
Michel Crucianu
Juan-Carlos Cubero

Gautam Das
Tijl De Bie
Luc De Raedt
Janez Demsar
Francois Denis
Giuseppe Di Fatta
Laura Dietz
Debora Donato
Dejing Dou
Kurt Driessens
Chris Drummond
Pierre Dupont
Haimonti Dutta
Pinar Duygulu-Sahin
Sašo Dzeroski
Tina Eliassi-Rad
Charles Elkan
Roberto Esposito
Floriana Esposito
Christos Faloutsos
Thomas Finley
Peter Flach
George Forman
Blaz Fortuna
Eibe Frank
Dayne Freitag
Alex Freitas
Elisa Fromont
Thomas Gaertner
Patrick Gallinari
João Gama
Dragan Gamberger
Auroop Ganguly
Gemma Garriga
Eric Gaussier
Floris Geerts
Claudio Gentile
Pierre Geurts
Amol Ghoting
Chris Giannella
Fosca Giannotti
Attilio Giordana
Mark Girolami
Derek Greene
Marko Grobelnik
Robert Grossman
Dimitrios Gunopulos
Maria Halkidi
Lawrence Hall
Jiawei Han
Tom Heskes
Alexander Hinneburg
Frank Höppner
Thomas Hofmann
Jaakko Hollmen
Tamas Horvath
Andreas Hotho
Eyke Hüllermeier
Manfred Jaeger
Sarah Jane Delany
Ruoming Jin
Alipio Jorge
Matti Kaariainen
Alexandros Kalousis
Benjamin Kao
George Karypis
Samuel Kaski
Petteri Kaski
Kristian Kersting
Ralf Klinkenberg
Adam Klivans
Jukka Kohonen
Joost Kok
Aleksander Kolcz
Stefan Kramer
Andreas Krause
Marzena Kryszkiewicz
Jussi Kujala
Pavel Laskov
Dominique Laurent
Nada Lavrac
David Leake
Chris Leckie
Philippe Leray
Carson Leung
Tao Li
Hang Li
Jinyan Li
Chih-Jen Lin
Jessica Lin

Michael Lindenbaum
Giuseppe Lipori
Kun Liu
Xiaohui Liu
Huan Liu
Michael Madden
Donato Malerba
Brad Malin
Giuseppe Manco
Bernard Manderick
Lluis Marquez
Yuji Matsumoto
Stan Matwin
Michael May
Prem Melville
Rosa Meo
Ingo Mierswa
Pericles Mitkas
Dunja Mladenic
Fabian Moerchen
Alessandro Moschitti
Claire Nedellec
John Nerbonne
Jenniver Neville
Joakim Nivre
Richard Nock
Alexandros Ntoulas
Siegfried Nijssen
Arlindo Oliveira
Salvatore Orlando
Miles Osborne
Gerhard Paass
Balaji Padmanabhan
George Paliouras
Themis Palpanas
Spiros Papadimitriou
Mykola Pechenizkiy
Dino Pedreschi
Ruggero Pensa
Jose-Maria Peña
Bernhard Pfahringer
Petra Philips
Enric Plaza
Pascal Poncelet
Doina Precup
Kai Puolamaki
Filip Radlinski
Shyamsundar Rajaram
Jan Ramon
Chotirat Ratanamahatana
Jan Rauch
Christophe Rigotti
Celine Robardet
Marko Robnik-Sikonja
Juho Rousu
Céline Rouveirol
Ulrich Rueckert
Stefan Rüping
Hichem Sahbi
Ansaf Salleb-Aouissi
Taisuke Sato
Yucel Saygin
Bruno Scherrer
Lars Schmidt-Thieme
Matthias Schubert
Marc Sebban
Nicu Sebe
Giovanni Semeraro
Pierre Senellart
Jouni Seppanen
Benyah Shaparenko
Arno Siebes
Tomi Silander
Dan Simovici
Andrzej Skowron
Kate Smith-Miles
Soeren Sonnenburg
Alessandro Sperduti
Myra Spiliopoulou
Ramakrishnan Srikant
Jerzy Stefanowski
Michael Steinbach
Jan Struyf
Gerd Stumme
Tamas Sziranyi
Domenico Talia
Pang-Ning Tan
Zhaohui Tang
Evimaria Terzi
Yannis Theodoridis

Dilys Thomas
Kai Ming Ting
Michalis Titsias
Ljupco Todorovski
Hannu Toivonen
Marc Tommasi
Luis Torgo
Volker Tresp
Panayiotis Tsaparas
Shusaku Tsumoto
Alexey Tsymbal
Koen Van Hoof
Jan Van den Bussche
Celine Vens
Michail Vlachos
Gorodetsky Vladimir
Ioannis Vlahavas
Christel Vrain
Jianyong Wang
Catherine Wang
Wei Wang
Louis Wehenkel
Li Wei
Ton Weijters
Shimon Whiteson
Jef Wijsen
Nirmalie Wiratunga
Anthony Wirth
Ran Wolff
Xintao Wu
Michael Wurst
Charles X. Ling
Dong Xin
Hui Xiong
Dragomir Yankov
Lexiang Ye
Dit-Yan Yeung
Shipeng Yu
Chun-Nam Yu
Yisong Yue
Bianca Zadrozny
Mohammed Zaki
Gerson Zaverucha
Changshui Zhang
Shi Zhong
Djamel Abdelkader Zighed
Albrecht Zimmermann
Jean-Daniel Zucker
Menno van Zaanen

Additional Reviewers

Franz Acherman
Ole Agesen
Xavier Alvarez
Davide Ancona
Joaquim Aparício
João Araújo
Ulf Asklund
Dharini Balasubramaniam
Carlos Baquero
Luís Barbosa
Lodewijk Bergmans
Joshua Bloch
Noury Bouraqadi
Johan Brichau
Fernando Brito e Abreu
Pim van den Broek
Kim Bruce
Luis Caires
Giuseppe Castagna
Barbara Catania
Walter Cazzola
Shigeru Chiba
Tal Cohen
Aino Cornils
Erik Corry
Juan-Carlos Cruz
Gianpaolo Cugola
Padraig Cunningham
Christian D. Jensen
Silvano Dal-Zilio
Wolfgang De Meuter
Kris De Volder
Giorgio Delzanno
David Detlefs

Anne Doucet
Rémi Douence
Jim Dowling
Karel Driesen
Sophia Drossopoulou
Stéphane Ducasse
Natalie Eckel
Marc Evers
Johan Fabry
Leonidas Fegaras
Luca Ferrarini
Rony Flatscher
Jacques Garrigue
Marie-Pierre Gervais
Miguel Goulão
Thomas Gschwind
Pedro Guerreiro
I. Hakki Toroslu
Görel Hedin
Christian Heide Damm
Roger Henriksson
Martin Hitz
David Holmes
James Hoover
Antony Hosking
Cengiz Icdem
Yuuji Ichisugi
Anders Ive
Hannu-Matti Järvinen
Andrew Kennedy
Graham Kirby
Svetlana Kouznetsova
Kresten Krab Thorup
Reino Kurki-Suonio
Thomas Ledoux
Yuri Leontiev
David Lorenz
Steve MacDonald
Ole Lehrmann Madsen
Eva Magnusson
Margarida Mamede
Klaus Marius Hansen
Kim Mens
Tom Mens
Isabella Merlo
Marco Mesiti
Thomas Meurisse
Mattia Monga
Sandro Morasca
M. Murat Ezbiderli
Oner N. Hamali
Hidemoto Nakada
Jacques Noye
Deniz Oguz
José Orlando Pereira
Alessandro Orso
Johan Ovlinger
Marc Pantel
Jean-François Perrot
Patrik Persson
Frédéric Peschanski
Gian Pietro Picco
Birgit Pröll
Christian Queinnec
Osmar R. Zaiane
Barry Redmond
Sigi Reich
Arend Rensink
Werner Retschitzegger
Nicolas Revault
Matthias Rieger
Mario Südholt
Paulo Sérgio Almeida
Ichiro Satoh
Tilman Schaefer
Jean-Guy Schneider
Pierre Sens
Veikko Seppänen
Magnus Steinby
Don Syme
Tarja Systä
Duane Szafron
Yusuf Tambag
Kenjiro Taura
Michael Thomsen
Sander Tichelaar
Mads Torgersen
Tom Tourwé
Arif Tumer
Ozgur Ulusoy

Werner Van Belle
Dries Van Dyck
Vasco Vasconcelos
Karsten Verelst
Cristina Videira Lopes
Juha Vihavainen
John Whaley
Mario Wolzko
Mikal Ziane
Gabi Zodik
Elena Zucca

Sponsors

We wish to express our gratitude to the sponsors of ECML PKDD 2008 for their essential contribution to the conference.

Table of Contents – Part I

Invited Talks (Abstracts)

Machine Learning Journal Abstracts

Data Mining and Knowledge Discovery Journal Abstracts

Regular Papers

Table of Contents – Part II

Regular Papers

Demo Papers

Industrializing Data Mining, Challenges and Perspectives

Françoise Fogelman-Soulié

KXEN, France

Business Intelligence is a very active sector in all industrial domains. Classical techniques (reporting and Olap), mainly concerned with presenting data, are already widely deployed. Meanwhile, Data Mining has long been used in companies as a niche-technique, reserved for experts only and for very specific problems (credit scoring, fraud detection for example). But with the increasing availability of large data volumes (in particular, but not only, from the Web), companies are more and more turning to data mining to provide them with high added-value predictive analytics. However producing models in large numbers, making use of large data volumes in an industrial context can only happen if solutions to challenges, both theoretic and operational, are found: we need algorithms which can be used to produce models when datasets have thousands of variables and millions of observations; we need to learn how to run and control the correct execution of hundreds of models; we need ways to automate the data mining process.

I will present these constraints in industrial contexts and show how KXEN has exploited theoretical results (coming from Vladimir Vapnik's work) to provide answers to the above-mentioned challenges. I will give a few examples of real-life applications and will conclude with some remarks on the future of data mining in the industrial domain.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 1, 2008.

From Microscopy Images to Models of Cellular Processes

Yoav Freund

Computer Science and Engineering
University of California, San Diego, USA

The advance of fluorescent tagging and of confocal microscopy is allowing biologists to image biochemical processes at a level of detail that was unimaginable just a few years ago. However, as the analysis of these images is done mostly by hand, there is a severe bottleneck in transforming these images into useful quantitative data that can be used to evaluate mathematical models.

One of the inherent challenges involved in automating this transformation is that image data is highly variable. This requires a recalibration of the image processing algorithms for each experiment. We use machine learning methods to enable the experimentalist to calibrate the image processing methods without having any knowledge of how these methods work. This, we believe, will allow the rapid integration of computer vision methods with confocal microscopy and open the way to the development of quantitative spatial models of cellular processes.

For more information, see `http://seed.ucsd.edu/~yfreund/NewHomePage/Applications/Biomedical_Imaging.html`

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 2, 2008.

Data Clustering: 50 Years Beyond K-means

Anil K. Jain

Computer Science and Engineering
Michigan State University, USA

The practice of classifying objects according to perceived similarities is the basis for much of science. Organizing data into sensible groupings is one of the most fundamental modes of understanding and learning. As an example, a common scheme of scientific classification puts organisms in to taxonomic ranks: domain, kingdom, phylum, class, etc.). Cluster analysis is the formal study of algorithms and methods for grouping objects according to measured or perceived intrinsic characteristics. Cluster analysis does not use category labels that tag objects with prior identifiers, i.e., class labels. The absence of category information distinguishes cluster analysis (unsupervised learning) from discriminant analysis (supervised learning). The objective of cluster analysis is to simply find a convenient and valid organization of the data, not to establish rules for separating future data into categories.

The development of clustering methodology has been a truly interdisciplinary endeavor. Taxonomists, social scientists, psychologists, biologists, statisticians, engineers, computer scientists, medical researchers, and others who collect and process real data have all contributed to clustering methodology. According to JSTOR, data clustering first appeared in the title of a 1954 article dealing with anthropological data. One of the most well-known, simplest and popular clustering algorithms is K-means. It was independently discovered by Steinhaus (1955), Lloyd (1957), Ball and Hall (1965) and McQueen (1967)! A search via Google Scholar found 22,000 entries with the word clustering and 1,560 entries with the words data clustering in 2007 alone. Among all the papers presented at CVPR, ECML, ICDM, ICML, NIPS and SDM in 2006 and 2007, 150 dealt with clustering. This vast literature speaks to the importance of clustering in machine learning, data mining and pattern recognition.

A cluster is comprised of a number of similar objects grouped together. While it is easy to give a functional definition of a cluster, it is very difficult to give an operational definition of a cluster. This is because objects can be grouped into clusters with different purposes in mind. Data can reveal clusters of different shapes and sizes. Thus the crucial problem in identifying clusters in data is to specify or learn a similarity measure. In spite of thousands of clustering algorithms that have been published, a user still faces a dilemma regarding the choice of algorithm, distance metric, data normalization, number of clusters, and validation criteria. A familiarity with the application domain and clustering goals will certainly help in making an intelligent choice. This talk will provide background, discuss major challenges and key issues in designing clustering

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 3–4, 2008.

algorithms, summarize well known clustering methods, and point out some of the emerging research directions, including semi-supervised clustering that exploits pairwise constraints, ensemble clustering that combines results of multiple clusterings, learning distance metrics from side information, and simultaneous feature selection and clustering.

Learning Language from Its Perceptual Context

Raymond J. Mooney

University of Texas at Austin, USA

Current systems that learn to process natural language require laboriously constructed human-annotated training data. Ideally, a computer would be able to acquire language like a child by being exposed to linguistic input in the context of a relevant but ambiguous perceptual environment. As a step in this direction, we present a system that learns to sportscast simulated robot soccer games by example. The training data consists of textual human commentaries on Robocup simulation games. A set of possible alternative meanings for each comment is automatically constructed from game event traces. Our previously developed systems for learning to parse and generate natural language (KRISP and WASP) were augmented to learn from this data and then commentate novel games. The system is evaluated based on its ability to parse sentences into correct meanings and generate accurate descriptions of game events. Human evaluation was also conducted on the overall quality of the generated sportscasts and compared to human-generated commentaries.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 5, 2008.

The Role of Hierarchies in Exploratory Data Mining

Raghu Ramakrishnan

Yahoo! Research, Santa Clara, CA, USA

In a broad range of data mining tasks, the fundamental challenge is to efficiently explore a very large space of alternatives. The difficulty is two-fold: first, the size of the space raises computational challenges, and second, it can introduce data sparsity issues even in the presence of very large datasets. In this talk, well consider how the use of hierarchies (e.g., taxonomies, or the OLAP multi-dimensional model) can help mitigate the problem.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 6, 2008.

Rollout Sampling Approximate Policy Iteration*

Christos Dimitrakakis[1] and Michail G. Lagoudakis[2]

[1] Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
dimitrak@science.uva.nl
[2] Department of Electronic and Computer Engineering
Technical University of Crete
Chania 73100, Crete, Greece
lagoudakis@intelligence.tuc.gr

Several researchers [2,3] have recently investigated the connection between reinforcement learning and classification. Our work builds on [2], which suggests an approximate policy iteration algorithm for learning a good policy represented as a classifier, without explicit value function representation. At each iteration, a new policy is produced using training data obtained through rollouts of the previous policy on a simulator. These rollouts aim at identifying better action choices over a subset of states in order to form a set of data for training the classifier representing the improved policy. Even though [2,3] examine how to distribute training states over the state space, their major limitation remains the large amount of sampling employed at each training state.

We suggest methods to reduce the number of samples needed to obtain a high-quality training set. This is done by viewing the setting as akin to a bandit problem over the states from which rollouts are performed. Our contribution is two-fold: (a) we suitably adapt existing bandit techniques for rollout management, and (b) we suggest a more appropriate statistical test for identifying states with dominating actions early and with high confidence. Experiments on two classical domains (inverted pendulum, mountain car) demonstrate an improvement in sample complexity that substantially increases the applicability of rollout-based algorithms. In future work, we aim to obtain algorithms specifically tuned to this task with even lower sample complexity and to address the question of the choice of sampling distribution.

References

1. Dimitrakakis, C., Lagoudakis, M.: Rollout sampling approximate policy iteration. Machine Learning 72(3), 157–171 (September 2008)
2. Lagoudakis, M.G., Parr, R.: Reinforcement learning as classification: Leveraging modern classifiers. In: Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, USA, August 2003, pp. 424–431 (2003)
3. Fern, A., Yoon, S., Givan, R.: Approximate policy iteration with a policy language bias. Advances in Neural Information Processing Systems 16(3) (2004)

* This is an extended abstract of an article published in the Machine Learning journal [1]. This project was partially supported by grant MCIRG-CT-2006-044980.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 7, 2008.

New Closed-Form Bounds on the Partition Function*

Krishnamurthy Dvijotham, Soumen Chakrabarti, and Subhasis Chaudhuri

IIT Bombay

Estimating the partition function is a key computation in graphical models that is required for important tasks like parameter estimation, model selection and structure learning. However, this computation is intractable in general. Thus, developing efficient and accurate approximations is of considerable interest. Variational methods express the computation of the partition function as an optimization problem (solving which intractable in general) and then relax the optimization problem in various ways to obtain approximations to the partition function. Two popular algorithms belonging to this framework are loopy belief propagation (LBP) and tree-reweighted belief propagation (TRW-BP). Both these algorithms are so-called *message passing* algorithms that work by updating local beliefs about probabilities at graph nodes by passing messages between the nodes in the graph until convergence is achieved. TRW-BP is guaranteed to give an upper bound on the partition function. However, neither algorithm is guaranteed to converge in general. Although convergent alternatives to TRW-BP have been proposed, they have no guarantees on the number of iterations required for convergence. Thus, these algorithms could be prohibitively expensive for applications requiring repeated inference on large graphs (like training large CRFs). In order to overcome this problem, Sutton et al. propose the *piecewise approximation* (PW) to the partition function. PW tends to be loose, but can be computed very fast, because it has a closed-form expression. Sutton et al. apply it successfully to common NLP tasks. In this paper, for a special class of potentials (that contain associative potentials), we prove that both LBP and TRW-BP converge in a single iteration. Using this fact, we obtain closed-form expressions for the TRW and LBP approximations to the partition function. In recent work, Wainwright et al. prove that LBP gives a lower bound on the partition function for binary attractive MRFs. We thus also get closed-form lower bounds for attractive associative MRFs. This enables us to obtain bounds on the error between the true partition function and these approximations for attractive associative MRFs. We also construct examples which show that TRW and LBP can give arbitrarily bad approximations to the marginal probabilities. Using the closed-form bounds for these special cases, we also develop novel closed-form upper bounds for arbitrary MRFs and closed-form lower bounds for associative binary MRFs. We also present experimental results showing that the new upper bounds are almost always tighter than the piecewise bound. The experiments also show that the novel lover bounds beat popular existing bounds like the mean-field bound on densely connected graphs.

References

1. Dvijotham, K., Chakrabarti, S., Chaudhuri, S.: New closed-form bounds on the partition function. Machine Learning 72(3), 205–229 (September 2008)

* This is an extended abstract of an article published in the Machine Learning journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 8, 2008.

Large Margin vs. Large Volume in Transductive Learning*

Ran El-Yaniv[1], Dmitry Pechyony[1], and Vladimir Vapnik[2]

[1] Computer Science Department, Technion - Israel Institute of Technology, Haifa, 32000, Israel
{rani,pechyony}@cs.technion.ac.il
[2] NEC Laboratories America, Princeton, NJ 08540, USA
vapnik@att.net

We focus on distribution-free *transductive learning*. In this setting the learning algorithm is given a 'full sample' of unlabeled points. Then, a training sample is selected uniformly at random from the full sample and the labels of the training points are revealed. The goal is to predict the labels of the remaining unlabeled points as accurately as possible. The full sample partitions the transductive hypothesis space into a finite number of *equivalence classes*. All hypotheses in the same equivalence class, generate the same dichotomy of the full sample. We consider a *large volume* principle, whereby the priority of each equivalence class is proportional to its "volume" in the hypothesis space.

The large volume principle was previously treated for the case of hyperplanes. In this paper, instead of hyperplanes, we consider soft classification vectors whose set of equivalence classes w.r.t. the full sample contains all possible dichotomies. Symmetry is broken by generating equivalence classes of non-uniform volume, defined via a non axis aligned data-dependent ellipsoid. Since exact or quantifiable approximate volume estimation is computationally hard, we resort to a cruder approach whereby volume is crudely related to the angles between hypotheses and the principal axes of the ellipsoid. This approach makes sense because long principal axes lie in regions of large volume. Our construction leads to a family of transductive algorithms and here we focus on one instantiation. Although the resulting algorithm is defined in terms of a non-convex optimization problem, we develop an efficient global optimum solution using a known technique. We also derive a data-dependent error bound for this algorithm.

Our experiments with the new Approximate Volume Regularization (AVR) algorithm over 31 datasets show its overwhelming advantage over TSVM and SVM in text categorization and image classification. However, on a different set of UCI datasets, TSVM and SVM are significantly superior to AVR. We identify some factors that influence the success and failure of our algorithm. One interesting observation is that AVR has significant advantage over TSVM when TSVM outperforms SVM, and vice versa.

* This is an extended abstract of an article published in the Machine Learning Journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 9–10, 2008.

References

1. El-Yaniv, R., Pechyony, D., Vapnik, V.: Large margin vs. large volume in transductive learning. Machine Learning 72(3), 173–188 (September 2008)

Incremental Exemplar Learning Schemes for Classification on Embedded Devices*

Ankur Jain and Daniel Nikovski

Mitsubishi Electric Research Laboratory. 201 Broadway, Cambridge, MA 02139
{jain,nikovski}@merl.com

In this paper, we focus on the data classification problem when the classifier operates on an embedded device (e.g., fault detection in device condition-monitoring data streams). Memory-based classifiers are an excellent choice in such cases, however, an embedded device is unlikely to be able to hold a large training dataset in memory (which could potentially keep increasing in size as new training data with new concepts arrive). A viable option then is to employ exemplar learning (EL) techniques to find a training subset comprising a few carefully selected *exemplars* of high functional value that fit in memory and effectively delineate the class boundaries. We propose two novel incremental EL schemes that unlike traditional EL approaches [3] are, (1) incremental (they naturally incorporate new training data streams), (2) offer ordered removal of instances (they can be customized to obtain exemplar sets of any user-defined size) and (3) robust (such that the exemplar sets generalize for other classifiers as well). Our proposed methods are as follows:

- EBEL (Entropy Based EL) – This method removes instances from the training set based on their *information content*. Instead of using an adhoc ranking scheme, it removes a training instance whose removal causes the least amount of drop in the conditional entropy of the class indicator variable insuring minimum loss of information.
- ABEL (AUC Based EL) – This method prunes data based on AUC (Area under ROC curve) performance. ABEL uses a *validation set* and prunes an instance if its removal offers the least drop in the AUC computed for this validation set.

We show that our schemes efficiently incorporate new training datasets while maintaining high-quality exemplar sets of any user-defined size. We present a comprehensive experimental analysis showing excellent classification-accuracy versus memory-usage tradeoffs of our proposed methods.

References

1. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. 6(1), 37–66 (1991)
2. Jain, A., Nikovski, D.: Incremental exemplar learning schemes for classification on embedded devices. Machine Learning 72(3), 189–203 (September 2008)
3. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning 38(3), 257–286 (2000)

* This is an extended abstract of an article published in the Machine Learning journal [2].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 11, 2008.

A Collaborative Filtering Framework Based on Both Local User Similarity and Global User Similarity*

Heng Luo, Changyong Niu, Ruimin Shen, and Carsten Ullrich

Department of Computer Science and Technology, Shanghai Jiao Tong University
1954 Huashan Road Shanghai 200030, China
{hengluo, cyniu, rmshen, ullrich_c}@sjtu.edu.cn

Collaborative filtering as a classical method of information retrieval is widely used in helping people to deal with information overload. In this paper, we introduce the concept of local user similarity and global user similarity, based on surprisal-based vector similarity and the application of the concept of maximin distance in graph theory. Surprisal-based vector similarity expresses the relationship between any two users based on the quantities of information (called surprisal) contained in their ratings, which is based on the intuition that less common ratings for a specific item tend to provide more discriminative information than the most common ratings. Furthermore, traditional methods of computing user similarity can not work if two users have not rated any identical item. To solve this problem, the global user similarity is introduced to define two users being similar if they can be connected through their locally similar neighbors. A weighted user graph is first constructed by using local similarity of any two users as the weight of the edge connecting them. Then the global similarity can be calculated as the maximin distance of any two nodes in the graph. Based on both of Local User Similarity and Global User Similarity, we develop a collaborative filtering framework called LS&GS. An empirical study using the MovieLens dataset shows that the proposed framework outperforms other state-of-the-art collaborative filtering algorithms.

References

1. Luo, H., Niu, C., Shen, R., Ullrich, C.: A collaborative ltering framework based on both local user similarity and global user similarity. Machine Learning 72(3), 231–245 (September 2008)

* This is an extended abstract of an article published in the Machine Learning journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 12, 2008.

A Critical Analysis of Variants of the AUC*

Stijn Vanderlooy[1] and Eyke Hüllermeier[2]

[1] MICC, Department of Computer Science, Maastricht University, The Netherlands
s.vanderlooy@micc.unimaas.nl
[2] Department of Mathematics and Computer Science, Marburg University, Germany
eyke@mathematik.uni-marburg.de

The area under the ROC curve, or AUC, has been widely used to assess the ranking performance of binary scoring classifiers. Given a sample of labeled instances, the metric considers the number of correctly ordered pairs of instances with different class label. Thus, its value only depends on the ordering of the scores but not on the "margin" between them. Consequently, it can happen that a small change in scores leads to a considerable change in AUC value. Such an effect is especially apparent when the number of instances used to calculate the AUC is small. On the other hand, two classifiers can have the same AUC value, even though one of them is a "better separator" in the sense that it increases the difference between scores of positive and negative instances, respectively. It has been argued that this insensitivity toward score differences is disadvantageous for model evaluation and selection. For this reason, three variants of the AUC metric that take the score differences into account have recently been proposed, along with first experimental results.

We present a unifying framework in which the conventional AUC and its variants can be modeled as special cases of a generalized AUC metric. Within this framework, we provide a formal analysis showing that the AUC variants produce estimates of the true AUC with a non-constant, model-specific bias, while the variance can decrease as well as increase. All things considered, the net effect on the quality of the estimations is thus not clear and, hereby, there is no solid theoretical foundation for the variants. Our analysis leads us to conjecture that actually none of the variants should be able to perform better in model selection than conventional AUC. This conjecture is corroborated by extensive experiments with synthetic data as well as real benchmark data, showing that the conventional AUC cannot be outperformed systematically by any variant, not in an ideal setting according to the theoretical analysis, and not in real model selection scenarios. Finally, our contribution also sheds light on recent research dealing with the construction of classifiers that (approximately) optimize the AUC directly, rather than accuracy or another performance metric.

References

1. Vanderlooy, S., Hüllermeier, E.: A critical analysis of variants of the AUC. Machine Learning 72(3), 247–262 (September 2008)

* This is an extended abstract of an article published in the Machine Learning Journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 13, 2008.

Improving Maximum Margin Matrix Factorization*

Markus Weimer[1], Alexandros Karatzoglou[2], and Alex Smola[3]

[1] Technische Universität Darmstadt, Germany
weimer@acm.org
[2] INSA de Rouen, LITIS, France
alexis@ci.tuwien.ac.at
[3] NICTA, Canberra 2601, Australia
alex.smola@gmail.com

Maximum Margin Matrix Factorization (MMMF) has been proposed as a learning approach to the task of collaborative filtering with promising results. In our recent paper [2], we proposed to extend the general MMMF framework to allow for structured (ranking) losses in addition to the squared error loss.

In this paper, we introduce a novel algorithm to compute the ordinal regression ranking loss which is significantly faster than the state of the art. In addition, we propose severals extensions to the MMMF model: We introduce offset terms to cater for user and item biases. Users exhibit vastly different rating frequencies ranging from only one rating per user to thousands of them. Similarly, some items get thousands of ratings while others get rated only once. We introduce an adaptive regularizer to allow for more complex models for those items and users with many ratings. Finally, we show equivalence between a recent extension introduced in [3] and a graph kernel approach described in [4]. Both aim at providing meaningful predictions for users with very little training data by virtue of the recommender graph.

We performed an evaluation of these extensions on two standard data sets: Eachmovie and Movielens. These experiments show that the introduced extensions do improve the predictive performance over the original MMMF formulation, even though we did not formally optimize the parameters.

References

1. Weimer, M., Karatzoglou, A., Smola, A.: Improving maximum margin matrix margin factorization. Machine Learning 72(3), 263–276 (September 2008)
2. Weimer, M., Karatzoglou, A., Le, Q., Smola, A.: Cofi rank - maximum margin matrix factorization for collaborative ranking. In: Advances in Neural Information Processing Systems 20. MIT Press, Cambridge (2008)
3. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems 20. MIT Press, Cambridge (2008)
4. Basilico, J., Hofmann, T.: Unifying collaborative and content-based filtering. In: Proc. Intl. Conf. Machine Learning, pp. 65–72. ACM Press, New York (2004)

* This is an extended abstract of an article published in the Machine Learning journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 14, 2008.

Finding Reliable Subgraphs from Large Probabilistic Graphs*

Petteri Hintsanen and Hannu Toivonen

HIIT Basic Research Unit, Department of Computer Science,
PO Box 68, FI–00014 University of Helsinki, Finland
{petteri.hintsanen,hannu.toivonen}@cs.helsinki.fi

Consider information search or discovery in a large graph of concepts and their weighted relationships. The user initiates a query by specifying some concepts ("search terms"), and wishes to obtain other concepts and relationships that connect the search concepts. An application example is in analysis of biological information, conveniently represented as a graph of biological concepts and their relations. A search engine we envision would allow a life scientist to query for connections between a gene and a phenotype, for instance, to find information supporting a hypothesis, or to help discover new hypothesis.

We propose two new methods for this problem, formalized as the *most reliable subgraph problem*. Such a problem is specified by a probabilistic graph G subject to random edge failures, a set of terminal vertices, and an integer K. The objective is to remove K edges from G such that the probability of connecting the terminals in the remaining subgraph is maximized. Extracting a subgraph of maximal reliability is a fundamental task. Besides the search problem described above, subgraph extraction is useful for visualization of large graphs and for preprocessing large graphs for other analysis methods.

The proposed methods, BPI and SPA, are based on greedy strategies for incrementally constructing a reliable subgraph of the desired size. BPI is based on simple use of best paths to span a subgraph, whereas SPA involves a more elaborate construction of series-parallel graphs. Unlike previous methods that prune the original graph until it reaches the given size, these incremental methods are relatively insensitive to the size of the original graph. We provide some technical details and a rough analysis of the algorithms. The practical performance of the methods is evaluated on real probabilistic graphs from the biological domain. The results indicate that the proposed methods scale much better to large input graphs, both computationally and in terms of the quality of the result. We use subgraph extraction as a component in Biomine[1], a search engine prototype for information discovery in biological databases.

References

1. Hintsanen, P., Toivonen, H.: Finding reliable subgraphs from large probabilistic graphs. Data Mining and Knowledge Discovery 17(1), 3–23 (August 2008)

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1].

[1] http://biomine.cs.helsinki.fi

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 15, 2008.

A Space Efficient Solution to the Frequent String Mining Problem for Many Databases*

Adrian Kügel and Enno Ohlebusch

Faculty of Engineering and Computer Sciences, University of Ulm, D-89069 Ulm

In the frequent string mining problem, one is given m databases $\mathcal{D}_1, ..., \mathcal{D}_m$ of strings and searches for strings that fulfill certain frequency constraints. The constraints consist of m pairs of thresholds $(minf_1, maxf_1)$, ..., $(minf_m, maxf_m)$ and one wants to find all strings ϕ that satisfy $minf_i \leq freq(\phi, \mathcal{D}_i) \leq maxf_i$ for all i with $1 \leq i \leq m$, where $freq(\phi, \mathcal{D}_i) = |\{\psi \in \mathcal{D}_i : \phi \text{ is a substring of } \psi\}|$.

Fischer et al. [2] presented an algorithm that solves the frequent string mining problem in linear time under the assumption that the number of databases is treated as a constant. The space consumption of this algorithm, however, is proportional to the total size of all databases. We improve this algorithm in such a way that its space consumption is proportional to the size of the largest database, and it takes linear time regardless of the number of databases. Also, our algorithm is more flexible in the sense that one of several databases can be replaced without having to recalculate everything, that is, intermediate data can be stored on file and be reused.

Algorithm for the Frequent String Mining Problem

- For each database $\mathcal{D}$ from the set of databases $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$ do:
  - Construct the enhanced suffix array of all strings in $\mathcal{D}$ and use a modified version of the algorithm of [2] to determine which substrings are relevant, i.e., satisfy the frequency constraint for database $\mathcal{D}$.
  - Store for each suffix ϕ the minimum and maximum length of relevant substrings ψ for which ϕ is the lexicographically smallest suffix which has ψ as a prefix.
- Iteratively calculate the intersection of the relevant substrings of databases $\mathcal{D}_1$ and $\mathcal{D}_2$, then the intersection of the result with the relevant substrings of $\mathcal{D}_3$, and so on. This is done by matching the respective database against the enhanced suffix array of $\mathcal{D}_1$.

References

1. Kügel, A., Ohlebusch, E.: A space efficient solution to the frequent string mining problem. Data Mining and Knowledge Discovery 17(1), 24–38 (August 2008)
2. Fischer, J., Heun, V., Kramer, S.: Optimal string mining under frequency constraints. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 139–150. Springer, Heidelberg (2006)

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 16, 2008.

The Boolean Column and Column-Row Matrix Decompositions*

Pauli Miettinen

Helsinki Institute for Information Technology
University of Helsinki
pauli.miettinen@cs.helsinki.fi

Matrix decompositions are used for many data mining purposes. One of these purposes is to find a concise but interpretable representation of a given data matrix. Different decomposition formulations have been proposed for this task, many of which assume a certain property of the input data (e.g., nonnegativity) and aim at preserving that property in the decomposition.

In this paper we propose two new decomposition formulations for binary matrices, namely the Boolean CX and CUR decompositions. They are natural combinations of two previously-presented decomposition formulations. The Boolean CX (BCX) decomposition assumes a binary input matrix A and decomposes it into two binary factor matrices, C and X, with matrix C containing a subset of A's columns. Matrix A is represented using the Boolean matrix product of C and X, $A \approx C \circ X$. The Boolean matrix product $\circ$ is like the normal matrix product, but with addition defined as $1 + 1 = 1$. In the Boolean CUR (BCUR) decomposition, A is decomposed into three matrices, C, U, and R, with matrix C containing a subset of A's columns and matrix R containing a subset of A's rows. Matrix A is represented as $A \approx C \circ U \circ R$.

We also study two subproblems of these decompositions, the Basis Usage (BU) problem and the Mixing Matrix (MM) problem. In the former we are given the matrices A and C and our goal is to find the matrix X such that $A \approx C \circ X$. In the latter we are given the matrices A, C, and R and our goal is to find the matrix U such that $A \approx C \circ U \circ R$. We give lower and upper bounds for the approximability of the BU and MM problems and use the results to show the NP-completeness of the BCX problem.

We give algorithms for the problems and study the performance of the algorithms via extensive experimental evaluation. Our results show that, despite the high theoretical complexity of the problems, even simple algorithms can perform well with both synthetic and real-world data.

References

1. Miettinen, P.: The Boolean column and column-row matrix decompositions. Data Mining and Knowledge Discovery 17(1), 39–56 (August 2008)

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 17, 2008.

SkyGraph: An Algorithm for Important Subgraph Discovery in Relational Graphs*

Apostolos N. Papadopoulos, Apostolos Lyritsis, and Yannis Manolopoulos

Data Engineering Research Lab.
Department of Informatics, Aristotle University
54124 Thessaloniki, Greece
{apostol,lyritsis,manolopo}@delab.csd.auth.gr

Graph mining is gaining importance due to the numerous applications that rely on graph-based data. Some example applications are: (i) analysis of microarray data in bioinformatics, (ii) pattern discovery in social networks, (iii) analysis of transportation networks, (iv) community discovery in Web data. Existing pattern discovery approaches operate by using simple constraints on the mined patterns. For example, given a database of graphs, a typical graph mining task is to report all subgraphs that appear in at least s graphs, where s is the frequency support threshold. In other cases, we are interested in the discovery of dense or highly-connected subgraphs. In such a case, a threshold is defined for the density or the connectivity of the returned patterns. Other constraints may be defined as well, towards restricting the number of mined patterns. There are three important limitations with this approach: (i) there is an on-off decision regarding the eligibility of patterns, i.e., a pattern either satisfies the constraints or not, (ii) in the case where the constraints are very strict we risk an empty answer or an answer with only a few patterns, and (iii) in the case where the constraints are too weak the number of patterns may be huge.

Towards dealing with the previous limitations, we address the problem of incorporating preferences in the pattern discovery process and we propose the SkyGraph algorithm which is based on min-cut computations. Each subgraph is seen as a record containing two attributes: (i) the order (number of vertices) and (ii) the edge connectivity. The importance of a discovered subgraph increases as both the order and the edge connectivity increase. Therefore, the best possible subgraphs (termed skyline subgraphs) are the ones that are maximized both in order and edge connectivity. To the best of the authors' knowledge, this is the first work studying the skyline problem in the process of knowledge discovery. The performance of the proposed technique is evaluated by using real-life as well as synthetically generated random graphs.

References

1. Papadopoulos, A.N., Lyritsis, A., Manolopoulos, Y.: Skygraph: An algorithm for important subgraph discovery in relational graphs. Data Mining and Knowledge Discovery 17(1), 57–76 (2008)

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 18, 2008.

Mining Conjunctive Sequential Patterns*

Chedy Raïssi[1,3], Toon Calders[2], and Pascal Poncelet[3]

[1] LIRMM, University of Montpellier, France
raissi@lirmm.fr
[2] Eindhoven University of Technology, Netherlands
t.calders@tue.nl
[3] LGI2P, Ecole des Mines d'Alès, France
pascal.poncelet@ema.fr

In this paper we study the discovery of frequent sequences and we aim at extending the non-derivable condensed representation in frequent itemset mining to sequential pattern mining. We start by showing a negative example: in the context of frequent sequences, the notion of non-derivability is *meaningless*.

This negative result motivated us to look at a slightly different problem: the mining of *conjunctions* of sequential patterns. This extended class of patterns turns out to have much nicer mathematical properties. For example, for this class of patterns we are able to extend the notion of non-derivable itemsets in a non-trivial way, based on a new unexploited theoretical definition of equivalence classes for sequential patterns. As a side-effect of considering conjunctions of sequences as the pattern type, we can easily form association rules between sequences. We believe that building a theoretical framework and an efficient approach for sequence association rules extraction problem is the first step toward the generalization of association rules to all complex and ordered patterns.

We also present a new depth-first approach to mine non-derivable conjunctive sequential patterns and show its use in mining association rules for sequences. This approach is based on a well known combinatorial theorem: the Möbius inversion. A performance study using both synthetic and real datasets illustrates the efficiency of our mining algorithm. These new introduced patterns have a high-potential for real-life applications, especially for network monitoring and biomedical fields with the ability to get non-redundant sequential association rules with all the classical statistical metrics such as confidence, conviction, lift etc.

References

1. Raïssi, C., Calders, T., Poncelet, P.: Mining conjunctive sequential patterns. Data Mining and Knowledge Discovery 17(1), 77–93 (August 2008)

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1]

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 19, 2008.

Adequate Condensed Representations of Patterns*

Arnaud Soulet[1] and Bruno Crémilleux[2]

[1]LI, Université François Rabelais de Tours
3 place Jean Jaurès
F-41029 Blois France
arnaud.soulet@univ-tours.fr
[2]GREYC-CNRS, Université de Caen
Campus Côte de Nacre
F-14032 Caen Cédex France
bruno.cremilleux@info.unicaen.fr

Patterns are at the core of the discovery of a lot of knowledge from data but their uses are limited due to their huge number and their mining cost. During the last decade, many works addressed the concept of condensed representation w.r.t. frequency queries. Such representations are several orders of magnitude smaller than the size of the whole collections of patterns, and also enable us to regenerate the frequency information of any pattern. Equivalence classes, based on the Galois closure, are at the core of the pattern condensed representations. However, in real-world applications, interestingness of patterns is evaluated by various many other user-defined measures (e.g., confidence, lift, minimum). To the best of our knowledge, these measures have received very little attention. The Galois closure is appropriate to frequency based measures but unfortunately not to other measures.

This paper extends the concept of pattern condensed representations. We propose a framework for condensed representations w.r.t. a large set of new and various queries named *condensable functions*. These queries encompass not only the frequency (conjunctive, disjunctive or negative) and frequency-based measures, but also address many other interestingness measures (e.g., minimum) and constraints having no suitable property of monotonicity. Condensed representations are achieved thanks to new closure operators automatically derived from each condensable function to get *adequate condensed representations*. We propose a sound and correct generic algorithm MicMac to efficiently mine the adequate condensed representations. Experiments show the conciseness of the adequate condensed representations, especially in dense and/or correlated data. They also demonstrate the scalability of our algorithm for measures or constraints which are intractible with naive methods.

We think that generalizing closure-based condensed representations will offer new tools for higher KDD tasks (e.g., non-redundant rules w.r.t. any measures), similarly there are many uses stemming from the frequency.

* This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 20–21, 2008.

References

1. Soulet, A., Crémilleux, B.: Adequate condensed representations. Data Mining and Knowledge Discovery 17(1), 94–110 (August 2008)

Two Heads Better Than One: Pattern Discovery in Time-Evolving Multi-aspect Data

Jimeng Sun[1], Charalampos E. Tsourakakis[2], Evan Hoke[4], Christos Faloutsos[2], and Tina Eliassi-Rad[3]

[1] IBM T.J. Watson Research Center
[2] Carnegie Mellon University
[3] Lawrence Livermore National Laboratory
[4] Apple Computer, Inc.

Data stream values are often associated with multiple *aspects*. For example, each value observed at a given time-stamp from environmental sensors may have an associated type (e.g., temperature, humidity, etc) as well as location. Time-stamp, type and location are the three aspects, which can be modeled using a tensor (high-order array). However, the time aspect is special, with a natural ordering, and with successive time-ticks having usually correlated values. Standard multiway analysis ignores this structure. To capture it, we propose *2 Heads Tensor Analysis* (2-heads), which provides a qualitatively different treatment on time. Unlike most existing approaches that use a PCA-like summarization scheme for all aspects, 2-heads treats the time aspect carefully. 2-heads combines the power of classic multilinear analysis (PARAFAC [1], Tucker [5], DTA/STA [3], WTA [2]) with wavelets, leading to a powerful mining tool. Furthermore, 2-heads has several other advantages as well: (a) it can be computed incrementally in a streaming fashion, (b) it has a provable error guarantee and, (c) it achieves significant compression ratio against competitors. Finally, we show experiments on real datasets, and we illustrate how 2-heads reveals interesting trends in the data.

This is an extended abstract of an article published in the Data Mining and Knowledge Discovery journal [4].

References

1. Harshman, R.: Foundations of the parafac procedure: model and conditions for an explanatory multi-mode factor analysis. UCLA working papers in phonetics 16 (1970)
2. Sun, J., Papadimitriou, S., Yu, P.: Window-based tensor analysis on highdimensional and multi-aspect streams. In: Proceedings of the International Conference on Data Mining (ICDM) (2006)
3. Sun, J., Tao, D., Faloutsos, C.: Beyond streams and graphs: Dynamic tensor analysis. In: KDD (2006)
4. Sun, J., Tsourakakis, C.E., Hoke, E., Faloutsos, C., Eliassi-Rad, T.: Two heads better than one: Pattern discovery in time-evolving multi-aspect data. Data Mining and Knowledge Discovery 17(1), 111–128 (August 2008)
5. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31(3) (1966)

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, p. 22, 2008.

TOPTMH: Topology Predictor for Transmembrane α-Helices*

Rezwan Ahmed, Huzefa Rangwala, and George Karypis

Department of Computer Science & Engineering,
University of Minnesota, Minneapolis, MN 55455
{ahmed,rangwala,karypis}@cs.umn.edu

Abstract. Alpha-helical transmembrane proteins mediate many key biological processes and represent 20%–30% of all genes in many organisms. Due to the difficulties in experimentally determining their high-resolution 3D structure, computational methods to predict the location and orientation of transmembrane helix segments using sequence information are essential. We present, TOPTMH a new transmembrane helix topology prediction method that combines support vector machines, hidden Markov models, and a widely-used rule-based scheme. The contribution of this work is the development of a prediction approach that first uses a binary SVM classifier to predict the helix residues and then it employs a pair of HMM models that incorporate the SVM predictions and hydropathy-based features to identify the entire transmembrane helix segments by capturing the structural characteristics of these proteins. TOPTMH outperforms state-of-the-art prediction methods and achieves the best performance on an independent static benchmark.

1 Introduction

Transmembrane helical (TMH) proteins play a crucial role in several cellular functions, such as cell-to-cell communication, cell signaling, and transportation of ions and small molecules [3], and are of key interest to the pharmaceutical industry as approximately 50% of all existing drugs are targeting transmembrane proteins [15]. Experimental determination of the three dimensional structure of TMH proteins is challenging, because they are difficult to crystallize and are too large for NMR studies [21]. As such, TMH proteins represent only 1% of known 3D protein structures [2], even though they account for about 20%–30% of the encoded proteins in several organisms [31]. Computational methods that can accurately predict the topology of TMH proteins by identifying the helical segments along with their orientation relative to the interior of the cell (also called cytoplasm) are currently the only high-throughput approach to characterize structural aspects of transmembrane proteins (See Figure 1).

* This work was supported by NSF EIA-9986042, ACI-0133464, IIS-0431135, NIH RLM008713A, NIH T32GM008347, the Digital Technology Center, University of Minnesota and the Minnesota Supercomputing Institute.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 23–38, 2008.

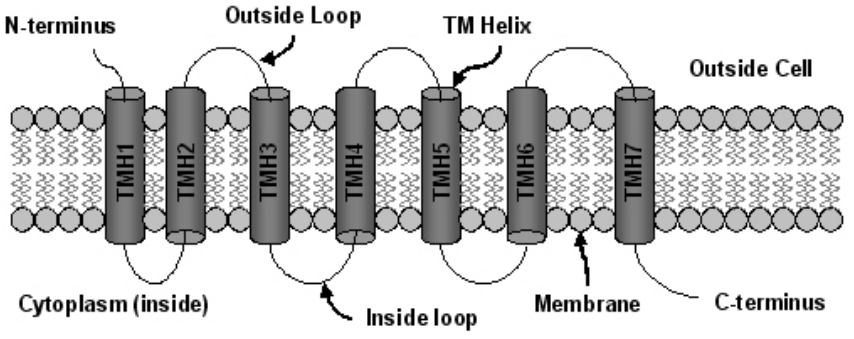


Fig. 1. Transmembrane α helix

Over the years, a number of different methods have been developed for predicting the topology of TMH proteins. In general, these methods need to predict the following items: (i) the type of each residue (e.g., helix, loop, etc.), (ii) the TMH segments, and (iii) their orientation. The various methods developed differ on the number of distinct steps that they use to predict the above items. Some methods predict each item individually, others utilize predictors that combine some of these steps, and others predict all three items in a single step. The residue types are predicted by either relying on the fact that membrane segments contain primarily hydrophobic residues (e.g., TopPred [29]) or by utilizing machine-learning approaches (e.g., neural networks, support vector machines) using as features the amino acid sequence of the protein or evolutionary information in the form of sequence profiles (e.g., PHDhtm [25], MEMSAT3 [10], SVMTop [20]). The segments are identified using simple hydrophobicity plots [16] to ascertain probable helical segments and then employ various rules based on the expected lengths of the TMH segments to either accept, reject, or break long segments [20,29,32]. The segment orientation is often determined by relying on the fact that the regions between TMH segments that are positively charged tend to reside in the intracellular regions of the membrane (*positive-inside* rule [29]). The approaches that combine segment identification with orientation determination (e.g., MEMSAT3) employ dynamic programming methods to determine the different segments of a TMH protein and its orientation relative to the cytoplasm. Finally, the approaches that predict all of the above items in a single step utilize hidden Markov models (HMM) that capture the different structural components of a TMH protein (e.g., TMH segment, inside loop, outside loop, signal peptide, etc.) as separate modules. These models are trained on either the amino acid sequence of the proteins (e.g., TMHMM [26] and HMMTOP [27]) or on sequence profiles (e.g., Phobius [17]) and predict the topology of a TMH protein by determining its most probable path through that model using Viterbi decoding [22].

This paper focuses on improving the accuracy of HMM-based approaches by combining them with an SVM-based approach that predicts the types of each

residue. Specifically, we developed a TMH topology prediction algorithm, called TOPTMH, that solves the residue-type prediction, segment identification, and orientation determination in three distinct steps. The type of each residue is annotated via an SVM-based approach utilizing a window-based encoding of the residues' profile information and a second order exponential kernel function [12,23,24]. The segments are identified by using a pair of HMMs that model the different structural components of TMH proteins. The first HMM uses as input the SVM predictions for each residue, whereas the second HMM uses as input hydropathy information as measured by a recently introduced hydrophobicity scale [8]. Finally, the orientation of the predicted segments is determined by applying the positive-inside rule.

The advantages of this approach are three-fold. First, by using a discriminative approach to learn a residue-type prediction model, the accuracy of these predictions are higher than those obtained (indirectly) by the HMM model. Second, by encoding the protein sequences via the SVM predictions, whose signal is significantly higher than that of the raw sequence profile, the demands imposed during HMM parameter estimation are substantially reduced allowing it to better focus on learning how to correctly identify the different segments. Third, by combining the outputs of the HMM models trained on the SVM predictions and on the hydrophobicity scores, it allows TOPTMH to correctly identify the TMH segments that have an amino acid composition that is similar to that of signal peptides.

We experimentally evaluated the performance of TOPTMH on three widely used datasets. Our evaluation was performed in two phases. First, we evaluated the gains obtained by TOPTMH by comparing it against an approach that uses a rule-based scheme to identify the TMH segments from the SVM predictions and another that uses just a single HMM model trained on the SVM predictions. Our evaluation showed that the HMM-based segment identification outperforms the rule-based approach by at least 50% in terms of the Q_{ok} score, which measures per-segment accuracy, and that by combining both the SVM- and the hydrophobicity-based HMM models, a further 3%–19% improvements can be obtained. Second, we evaluated its performance by comparing it against Phobius [17] and MEMSAT3 [10]. Our evaluation showed that TOPTMH outperforms both of them across the different datasets. We also evaluated the performance of TOPTMH on an independent static benchmark [14]. The results on this blind evaluation showed that TOPTMH achieves the highest scores on high-resolution sequences (Q_2 score of 84% and Q_{ok} score of 86%) against existing state-of-the-art systems while achieving low signal peptide error.

2 Background and Definitions

2.1 Transmembrane Helical Proteins

The structure of a typical TMH protein is shown in Figure 1. It consists of a series of helical segments passing through the cell's membrane (bilipid layer) separated by loop segments that are either on the inside or the outside side of

the membrane. TMH segments can have two orientations: they can be going from the inside to the outside or from the outside to the inside of the cell. This orientation is relative to the location of N-terminus of the TMH protein. The TMH topology prediction problem involves predicting the residues that make up the helical segments and their orientation.

2.2 Position Specific Scoring Matrices

The position specific scoring matrix (PSSM) of a protein is obtained from a multiple sequence alignment of that protein and a set of other proteins that have a statistical significant sequence similarity (i.e., they are expected to be homologs). For a sequence X of length n, its PSSM is represented by a $n \times 20$ matrix $\mathcal{P}_X$. The n rows of this matrix correspond to the various positions in X and the columns correspond to the 20 distinct amino acids. The position specific scoring matrices used by TOPTMH were generated using the latest version of the PSI-BLAST algorithm [1] (available in NCBI's blast release 2.2.13), and were derived from the multiple sequence alignment constructed after five iterations using an e value of 10^{-2} for initial and subsequent sequence inclusions (i.e., we used `blastpgp -j 5 -e 0.01 -h 0.01`). The PSI-BLAST was performed against the SWISS-PROT [4] database release 53.0 that contains 269,293 sequences. A post processing step was performed to extract the log-odds scores ($n \times 20$ matrix) of each protein sequence from the PSI-BLAST output to use as the input feature for residue classification.

2.3 Hydrophobicity Scale

A hydrophobicity (HP) scale assigns a value to each of the 20 standard amino acids based on its hydrophobicity. In the context of TMH prediction methods, the Kyte and Doolittle [16] and the GES [6] HP scales are commonly used. These scales are based on biophysical or statistical analysis of high-resolution membrane protein structures and do not fully capture the cellular context of the membrane proteins [8]. For this reason, TOPTMH uses a recently published [8] HP scale (ΔG^{aa}_{app} scale) that captures the energetics of the protein-lipid interaction in biological contexts and thus is more biologically relevant. It has been shown that this scale is able to determine the topology of membrane proteins with higher precision than other scales [30].

3 TOPTMH Algorithm

The TOPTMH algorithm solves the TMH prediction problem by first assigning a score to each residue based on its likelihood to be in a helix state (residue annotation step), then using these scores it determines the protein's TMH segments (segment identification step), and finally using the positive-inside rule it determines their orientation (orientation determination step). These steps are described in the rest of this section.

3.1 Residue Annotation Step

We developed an SVM-based TMH residue annotation approach that uses features obtained from the protein's PSSM. Its overall structure is similar to that used by existing methods for SVM-based structural and functional annotation of protein residues using position specific scoring matrices (e.g., secondary structure for globular proteins [12], solvent accessible surface area [24], disorder prediction [24], and DNA-binding [24]).

TOPTMH formulates the residue annotation problem as a binary classification problem whose goal is to predict if a residue belongs to a helix state or not. For each residue i of a protein sequence X, the input to the SVM is a $(2w+1)$-length subsequence (w`mer`) of X centered at position i. Each w`mer` is represented by a vector x_i of length $(2w+1) \times 20$ that is obtained by concatenating the rows of the PSSM for each position of the w`mer`. This w`mer`-based input is used for both training and prediction. The parameter w determines the length of the local environment around the ith sequence position used while building and applying the model and its optimal value is determined experimentally.

TOPTMH uses SVMlight [9] to learn the actual SVM model and utilizes the second order exponential function (*soe*) [12] as its kernel function. The *soe* kernel has been shown to produce better results than the traditional radial basis function (*rbf*) kernel for various sequence annotation prediction problems [12,24,23]. For a sequence, these predictions are available as a web service called MONSTER[1]. In the context of TOPTMH, the *soe* kernel function is given by

$$\mathcal{K}^{soe}(x_i, y_j) = \exp\left(1 + \frac{\mathcal{K}^2(x_i, y_j)}{\sqrt{\mathcal{K}^2(x_i, y_j)\,\mathcal{K}^2(x_i, y_j)}}\right), \tag{1}$$

where x_i and y_j are the vector representations of two w`mers`, $\mathcal{K}^2$ is given by

$$\mathcal{K}^2(x_i, y_j) = \langle x_i, y_j \rangle + \langle x_i, y_j \rangle^2, \tag{2}$$

and $\langle x_i, y_j \rangle$ denotes the dot-product of the x_i and y_j vectors.

3.2 Segment Identification Step

In order to determine the best approach for identifying the TMH segments we developed and studied three different approaches. The first approach utilizes a simple scheme based on empirical rules and the other two predict the topology by employing hidden Markov models (HMM) [22]. The first HMM-based approach uses a single HMM based solely on the SVM scores, whereas the second uses two HMMs—one based on SVM scores and one based on hydrophobicity scales.

Rule-Based. The rule-based segment identification approach post-processes the SVM-based residue annotations and identifies the segments by applying some heuristics rules that take into account the minimum and maximum lengths of

[1] http://bio.dtc.umn.edu/monster

the TMH segments. Specifically, for each protein, this approach traverses the SVM annotated residues and identifies all maximal contiguous segments that were annotated as TMHs by the SVM. Any TMH segment whose length l is shorter than the minimum length of L_{min} residues is rejected (i.e., converted into non-helix residues). If any of the remaining segments have $l > L_{max}$, they are split into two separate segments as follows. For the segments with $l \leq 2L_{opt} + C$, the segment is split by changing the middle C residues into loops. For segments with $l > 2L_{opt} + C$, the segment is split by creating two helical segments consisting of the first and last L_{opt} residues and converting the remaining central residues into loops. The threshold values L_{min}, L_{max}, L_{opt} and C are set as 9, 38, 19 and 6 respectively. These values were initially chosen based on a literature review [29,3,32] and then optimized to provide the best results given the SVM-based annotations produced by TOPTMH.

HMM-Based. The HMM-based segment identification approaches determine the segments of a TMH protein by *threading* the sequence into an HMM model that is designed to capture the various structural components of a TMH protein. These approaches were motivated by recent studies which showed that HMM-based TMH prediction methods are well-suited for predicting the topology of TMH proteins as they can directly learn from the data the various structural constraints associated with TMH protein segments and their relations to the protein's underlying sequence and/or PSSM [3,18,5]. However, unlike these methods, the HMM-based approaches that we developed take into account the SVM-scores produced by the residue annotation step, which provide better per-residue predictions for the helix/non-helix states than the maximum likelihood approaches used by HMMs. The architecture of our HMM model, shown in Figure 2, is designed to capture the known structural information of TMH proteins and is similar to that employed by Phobius [17]. The model contains four major compartments: (i) helix, (ii) inside loop, (iii) outside loop, and (iv) signal peptide. The helix compartment is composed of two submodels each containing 35 states. One submodel is used for modeling helix segments that go from inside towards the outside, and the other for the helix segments that go from outside towards the inside. In each of these submodes, states 1–8 contain transitions to only the next state, whereas states 9–34 can transition to the next state or to state 35 (last state). Thus, any predicted helix segment will be of length 9–35 residues long. The outside loop compartment is divided into two submodels to represent long and short non-cytoplasmic loops. Each of these submodels contains 20 states to model loops that are at least 1–20 residues long. Each submodel also has a state with self-transition to represent long cytoplasmic loops. The inside loop compartment also contains 20 states to allow it to model loops that are 1–20 residues long. The signal peptide compartment was designed based on Phobius model and it has three regions: the n-region (10 states), the h-region (20 states), and the c-region (20 states). The last state of the c-region represents a cleavage site transitioning to a outside loop state.

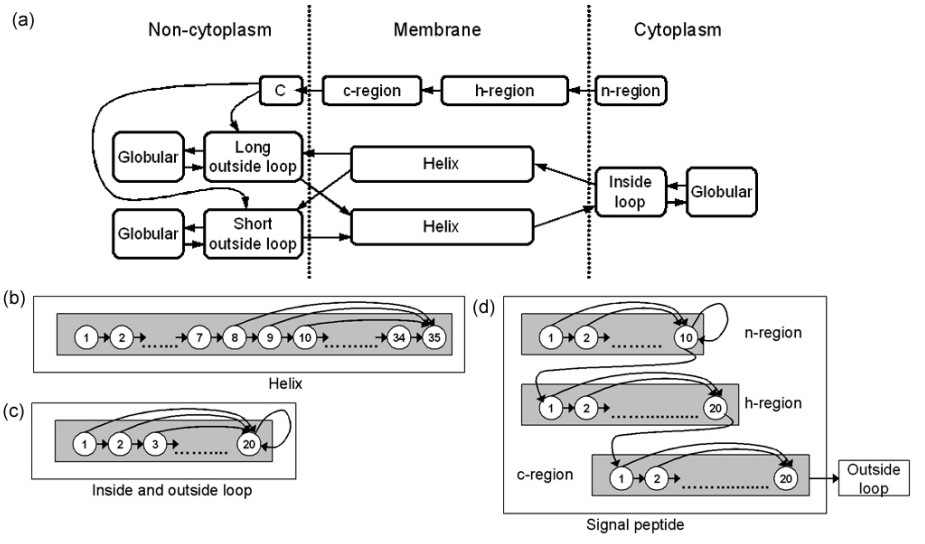


Fig. 2. The layout of the HMM model used in TOPTMH

The HMM models were built using the UMDHMM [11] package (version 1.02), which was modified to take as input annotated protein sequences. The threading of a sequence through the HMM model was done using the Viterbi [22] algorithm.

HMM Based on SVM Scores (HMM-SVM). This approach builds an HMM model that only takes into account the per-residue SVM scores produced by the annotation step. To construct the training set, the SVM score for each residue is computed. Since, HMMs are primarily designed to operate on finite size alphabets, the raw SVM scores are discretized into a finite number of bins with each bin corresponding to a distinct symbol. The final training set for the HMM corresponds to a set of proteins with known TMH topology represented as sequences of SVM-score based bins. A similar SVM-based prediction followed by discretization is performed when this model is used to predict the topology of a test protein. We discretized the SVM scores into equal-size intervals, and assigned all residues with scores ≤ -3 and ≥ 3 into the first and last bin, respectively.

HMM Based on SVM Scores and Hydrophobicity Scores (HMM-SVM+HP). This model builds a pair of HMM models—one based on SVM scores (HMM-SVM) and one based on the hydrophobicity values (HMM-HP) of known TMH sequences and combines the topology predictions from both HMM models. This approach was motivated by the fact that in certain cases, the SVM-based residue annotation may fail to identify certain hydrophobic TMH segments. This is further discussed in Section 5.

Table 1. Discretization of Hydrophobicity values

Labels	Amino Acids	HP Values
1	R, E, K, D	$2.5 < h$
2	N, H, P, Q	$1.0 < h < 2.5$
3	T, Y, G, S	$-0.1 < h < 0.9$
4	F, V, C, A, M, W	$-0.4 < h < -0.1$
5	I, L	$h < -0.5$

HP Values denotes a range of hydrophobicity values decided based on [8]

The HMM-SVM model is identical to that described in the previous section. The HMM-HP model is built by first encoding the amino acids of each TMH protein as a sequence of discretized hydrophobicity values. Table 1 shows the scheme used to discretize the hydrophobicity values for each amino acid. Both the HMM-SVM and HMM-HP models are used independently to predict the TMH segments. The final set of predictions consists of the segments predicted by HMM-SVM and those segments predicted by HMM-HP that do not overlap with any of the segments of HMM-SVM. Two segments are considered to overlap if they have more than five residues in common. Since this approach combines both the SVM- and HP-based HMM models, we will refer to it as HMM-SVM+HP.

3.3 Orientation Determination Step

Once the TMH segments have been identified, their orientation relative to the N-terminus is determined by applying the positive-inside rule [29] using the technique introduced in THUMBUP [32]. In this approach, each protein is first coded into a binary sequence by assigning a one to the first protein residue and all the arganine and lysine residues and a zero to the remaining residues. Then, a score is computed for each loop by adding the values of its 15 neighboring residues on each side. If the total score for odd-numbered loops is greater than or equal to that of even loops, the N-terminus is inside the membrane, otherwise it is outside.

4 Experimental Design

4.1 Datasets

We evaluated the prediction performance of the TOPTMH method on datasets used by the Phobius and MEMSAT3 methods and by participating on the static benchmark [13]. The datasets obtained from the Phobius study included a set of 247 transmembrane proteins and a set of 45 transmembrane proteins that contained signal peptide residues with transmembrane helix segments. We will denote the first dataset as TM-Only and the second as TM-SP. The dataset

obtained from MEMSAT3 consisted of a set of 184 non-homologous transmembrane proteins denoted as MÖLLER that also contained a few signal peptide proteins.

The static benchmark consists of a set of 2247 sequences whose true annotations are not given to the public. A method predicts the annotations for these sequences and uploads them to the evaluation server. The server assesses the quality of the predictions and compares them to that obtained by other methods. The 2247 sequences contain four distinct subsets. The first is the high-resolution subset which contains sequences of proteins whose high resolution structure is available, the second is the low-resolution subset that includes membrane proteins detected using low resolution structures, the third subset is the globular protein subset which includes globular protein sequences and the fourth is the signal peptide subset that includes proteins sequences with signal peptide residues. The sequences provided to the public is not grouped in the above mentioned subsets, but the results published on the evaluation server is presented accordingly.

4.2 Training and Testing Methodology

For each of the TM-SP and TM-ONLY datasets, the different methods were evaluated using a standard 10-fold cross validation protocol by splitting the proteins into 10 different parts. The percent sequence identity between the different folds were at most 30% and 35% for the TM-ONLY and TM-SP datasets, respectively. The ten folds were identical to that used by Phobius making it possible to directly compare our results with those obtained by Phobius.

The two-level HMM-SVM model was trained as follows. The training set was further split into 10 different folds $\{f_1, \ldots, f_{10}\}$. For each fold f_i, the other nine folds were used to train the SVM model and then used to predict the residues for the proteins in f_i. At the end of this step all the residues of the proteins in the training set have SVM predictions. These predictions are then used to train the HMM model for the training set. In addition, the entire training set is used to build an SVM residue prediction model. Note that the test set is not used anywhere during training. During testing, the residues of each test protein are first predicted using the SVM model built on the entire training set, and these predictions are provided as input to the HMM model to predict the TMH segments.

The predictions for the static benchmark were obtained by training the SVM and HMM models using all the sequences from TM-SP and TM-ONLY datasets.

4.3 Evaluation Metrics

The performance of TMH prediction is evaluated on a per-residue and on a per-segment basis using well-established metrics [3]. The per-residue evaluation measures the ability of a method to correctly annotate the different residues into helices or non-helices (two classes). We used three per-residue metrics denoted by $Q_{2T}^{\%obs}$, $Q_{2T}^{\%prd}$, and Q_2. $Q_{2T}^{\%obs}$ is the percentage of observed TMH residues that

are predicted correctly (helix recall), $Q_{2T}^{\%prd}$ is the percentage of predicted TMH residues that are predicted correctly (helix precision), and Q_2 is the percentage of correctly predicted residues (both helix and non-helix).

The per-segment evaluation measures the ability of a method to correctly identify the actual TMH segments. We used three per-segment metrics denoted by $Q_{htm}^{\%obs}$, $Q_{htm}^{\%prd}$, and Q_{ok}. $Q_{htm}^{\%obs}$ is the percentage of observed TMH segments that are predicted correctly (TMH segment recall), $Q_{htm}^{\%prd}$ is the percentage of predicted TMH segments that are predicted correctly (TMH segment precision), and Q_{ok} is the percentage of proteins for which all the TMH segments are predicted correctly. Note that Q_{ok} is a very strict metric as each protein contributes either a zero or an one. In the above metrics, a predicted TMH segment is considered to be correctly identified if there is an overlap of ten residues between the predicted and observed helix segments[2] In addition, a predicted helix segment is counted only once. This is illustrated by considering the following examples:

```
Obs1: TTTTTTTTTTTTTTT------TTTTTTTTTTTT
Pred1: -----TTTTTTTTTTTTTTTTTTTTTTTTT---

Obs2 : ---TTTTTTTTTTTTTTTTTTTTTTTTTTT--
Pred2: TTTTTTTTTTTTT------TTTTTTTTTTTTTT
```

In this example, Obs1 and Pred1 are the observed and predicted TMH segments for a particular protein sequence. During evaluation, the second segment of the Obs1 sequence will not be considered as correctly predicted, since the only segment predicted in Pred1 is already accounted for in the first segment of the Obs1 sequence. On the other hand, the second segment of the Pred2 sequence will be considered as incorrectly predicted as the first segment will be considered for the only segment in Obs2 sequence.

Although, the per-residue measures capture the accuracy of a method to predict the annotation label for a residue, it is not able to assess the ability of the method to identify the TMH segments separated by loop regions of different lengths. Hence, TMH prediction algorithms are mostly evaluated using per-segment metrics.

5 Results

5.1 Residue Annotation Performance

The performance achieved by the SVM-based residue annotation for different values of w is shown in Table 2. This table shows the per-residue performance metrics (Q_2, $Q_{2T}^{\%obs}$ and $Q_{2T}^{\%prd}$) for a subset of the TM-Only dataset. We observe that in terms of the various metrics, the performance achieved for different values of w is rather similar. The only exception is $w = 2$, where the performance

[2] Earlier techniques used an overlap of only three [3] or five [17] residues, which is too short and can artificially inflate the performance of a scheme.

Table 2. Residue Annotation Performance with varying w`mer` length

w`mer`	Q_2	$Q_{2T}^{\%obs}$	$Q_{2T}^{\%prd}$
2	86.6	78.1	76.9
5	88.2	85.3	75.5
7	**88.3**	84.7	**77.4**
11	88.3	**85.5**	76.6

The numbers in bold show the best w`mer` length performance as measured for that metric.

is substantially lower than the rest. Overall, the best performance was obtained using w`mer` of length seven. For this reason, all the remaining experiments presented in this study use $w = 7$.

5.2 Segment Identification Performance

Table 3 presents the per-residue and per-segment based results of different TMH segment identification approaches on the TM-Only and TM-SP datasets. For the SVM-HMM approach, Table 3 shows three different sets of results that were obtained by binning the SVM scores into 5, 7, and 12 bins (HMM-SVM-D5, HMM-SVM-D7, and HMM-SVM-D12). The row labeled "Raw-SVM" shows the results obtained by using as TMH segments the maximal contiguous segments that were predicted as TMHs by the SVM (i.e., the set of segments that form the input to the rule-based segment identification approach).

Comparing the per-residue performance achieved by the various approaches we see that Raw-SVM achieves very good per-residue two-state accuracy (Q_2). It has the highest Q_2 value for TM-Only and the second highest for TM-SP. However, focusing on this metric alone is misleading because most of the residues in transmembrane proteins are non-helix [19] and relatively high Q_2 values can be obtained by simply predicting most of the residues as being in a non-helix state. Consequently, high Q_2 values represent good performance only if they are accompanied with high helix recall ($Q_{2T}^{\%obs}$) values. In light of this discussion, we see that the HMM-based segment identification approaches tend to achieve considerably better recall values (especially for TM-SP) while their helix precision ($Q_{2T}^{\%prd}$) is in some cases better than that of the Raw-SVM approach. Among the different schemes, the rule-based approach achieves the best precision results, whereas the approach that combines the SVM- and HP-based HMMs (HMM-SVM-D7+HP) achieves the best recall. However, unlike the high precision achieved by the HMM-SVM-D7+HP approach, the rule-based scheme achieves the lowest recall leading to the worst Q_2 values.

Comparing the per-segment performance, we see that the Raw-SVM approach achieves Q_{ok} scores that range from 35%–40%, which are by far the lowest among the different approaches. These results indicate that even though Raw-SVM can correctly predict a large fraction of the helical residues, it fails to predict

Table 3. TMH Segment Identification Performance

	TM-SP			TM-ONLY		
Per-Residues Scores						
Methods	Q_2	$Q_{2T}^{\%obs}$	$Q_{2T}^{\%prd}$	Q_2	$Q_{2T}^{\%obs}$	$Q_{2T}^{\%prd}$
Raw-SVM	96.73	71.10	86.60	90.64	84.30	83.10
Rule	95.16	59.56	95.89	89.19	79.65	87.36
HMM-SVM-D5	96.28	76.39	84.87	89.40	85.54	82.25
HMM-SVM-D7	96.45	76.85	87.72	89.34	85.61	82.23
HMM-SVM-D12	96.24	77.56	84.45	89.31	86.13	81.35
HMM-SVM-D7+HP	97.08	84.80	88.50	89.46	86.21	82.04
Per-Segment Scores						
Methods	Q_{ok}	$Q_{htm}^{\%obs}$	$Q_{htm}^{\%prd}$	Q_{ok}	$Q_{htm}^{\%obs}$	$Q_{htm}^{\%prd}$
Raw SVM	35.55	85.23	70.09	38.86	94.34	74.33
Rule	64.44	75.00	100.00	70.85	92.88	94.96
HMM-SVM-D5	64.44	84.09	87.05	71.66	95.39	93.73
HMM-SVM-D7	71.11	85.23	92.59	72.06	95.63	93.52
HMM-SVM-D12	60.00	85.22	85.22	70.04	95.80	92.87
HMM-SVM-D7+HP	84.44	93.18	93.18	73.68	96.12	93.33

correctly large contiguous portions of each helical segment. On the other hand, the per-segment performance achieved by the other segment identification approaches are considerably higher. Both the rule- and HMM-based approaches are able to significantly improve over Raw-SVM for both the TM-SP and TM-ONLY datasets. Among them, the approaches based on HMM-SVM outperform the rule-based approach by 2%–12%, even though the latter achieved the highest $Q_{htm}^{\%prd}$ scores (100% and 96.44% for TM-SP and TM-ONLY, respectively).

The overall best Q_{ok} results were obtained by the HMM-SVM-D7+HP approach. In particular, the Q_{ok} values achieved by HMM-SVM-D7+HP are 19% and 3% better than the next best performing scheme (HMM-SVM-D7) on the TM-SP and TM-ONLY datasets, respectively. The large performance advantage of HMM-SVM-D7+HP over HMM-SVM-D7 on the TM-SP dataset are primarily due to increases in recall ($Q_{htm}^{\%obs}$). HMM-SVM-D7+HP achieves a $Q_{htm}^{\%obs}$ of 93.18% compared to the 85.23% achieved by HMM-SVM-D7. A possible explanation for the relatively poor performance of HMM-SVM-D7 is that due to the signal peptide segments present in some of the sequences in the TM-SP dataset, the SVM model fails to identify some of the TMH residues. However, these residues can be correctly identified when hydrophobicity scores are considered, and as such the combined HMM-SVM-D7+HP approach leads to better overall results.

Table 4. Performance Comparison with Phobius

	TM-SP	TM-ONLY
Method	Accuracy	Accuracy
TOPTMH	93.18	75.71
Phobius	91.10	63.60

Accuracy denotes the percentage of the correctly predicted proteins and a prediction was counted correct when all predicted TMH segments overlap all observed TMH segments over a five residue stretch and loops were located correctly. Prediction accuracy did not consider incorrect prediction of signal peptide segments to be consistent as [17].

Table 5. Performance Comparison with MEMSAT3 on the MÖLLER dataset

Method	# TM SEG	# TOPO	# TOPO+LOC	# TOPO+LOC(10)
TOPTMH	162 (88.04%)	149 (80.98%)	134 (72.83%)	131 (71.20%)
Phobius	152 (82.60%)	134 (72.80%)	126 (68.40%)	120 (65.20%)
MEMSAT3	156 (84.80%)	150 (81.50%)	147 (79.90%)	141 (76.60%)

TM SEG denotes the number of predicted proteins that had correct number of TMH segments irrespective of topology or location. # TOPO denotes the number of proteins for which the orientation of the protein (N-terminus is inside or outside of the cytoplasm) was predicted correctly. # TOPO+LOC denotes the number of proteins for which the topology and the TMH segment locations were predicted correctly. This score was calculated based on five residue segment overlap. # TOPO+LOC(10) shows the # TOPO+LOC scores for ten residue segment overlap.

5.3 Performance Comparison with Previous Methods

We compared the TOPTMH method (i.e., HMM-SVM-D7+HP) with Phobius and MEMSAT3, which are two of the best TMH prediction methods currently available. Phobius uses a sophisticated HMM to mark the TMH and signal peptide regions and MEMSAT3 uses a combination of neural network and dynamic programming to identify the TMH segments. The results of these comparisons are shown in Tables 4 and 5. To facilitate the comparisons between the different schemes, the performance metrics used in these tables are similar to the metrics used in Phobius and MEMSAT3 and allow us to directly compare TOPTMH performance with these systems.

Comparing TOPTMH's performance against Phobius (Table 4) we see that TOPTMH achieves accuracies that are 2% and 10% higher than those achieved by Phobius on the TM-SP and TM-ONLY datasets, respectively. The performance advantage of TOPTMH over Phobius also holds for the MÖLLER dataset (Table 5) as well. TOPTMH performed better in all three categories by correctly predicting 162, 149, and 134 proteins compared to the 152, 134, and 126 proteins predicted by Phobius, respectively.

Comparing TOPTMH's performance against MEMSAT3 (Table 5) we see that TOPTMH was able to predict the correct number of TMH segments for

Table 6. TMH Benchmark Results

	High Resolution Accuracy						Low Resolution Accuracy					
	Per-segment			Per-residue			Per-segment			Per-residue		
Method	Q_{ok}	$Q_{htm}^{\%obs}$	$Q_{htm}^{\%prd}$	Q_2	$Q_{2T}^{\%obs}$	$Q_{2T}^{\%prd}$	Q_{ok}	$Q_{htm}^{\%obs}$	$Q_{htm}^{\%prd}$	Q_2	$Q_{2T}^{\%obs}$	$Q_{2T}^{\%prd}$
TOPTMH	**86**	95	96	**84**	75	90	66	92	88	**90**	84	80
PHDpsihtm08	84	99	98	80	76	83	67	95	94	89	87	77
HMMTOP2	83	99	99	80	69	89	66	94	93	90	85	83
MEMSAT3	80	98	97	83	78	88	63	92	87	88	86	76
Phobius	80	92	93	80	69	84	65	90	88	90	81	79
DAS	79	99	96	72	48	94	39	93	81	86	65	85
TopPred2	75	90	90	77	64	83	48	84	79	88	74	71
TMHMM1	71	90	90	80	68	81	**72**	91	92	90	83	80
SOSUI	71	88	86	75	66	74	49	88	86	88	79	72
PHDhtm07	69	83	81	78	76	82	56	85	86	87	83	75

Results for TOPTMH and MEMSAT3 were obtained by collecting predictions for test set of the TMH static benchmark [13] and submitting the results to the benchmark server. Phobius [17] prediction were collected loading the benchmark test sequences to the Phobius web server [13] and submitting the output to the benchmark server. All the other results were provided by the TMH static benchmark evaluation web-site.

more proteins (162 *vs* 156) and predict the correct topology for a similar number of proteins (149 *vs* 150). However MEMSAT3 was able to predict more proteins with both correct topology and location than TOPTMH (147 *vs* 134). We believe that this is primarily due to the fact that due to the binary classification of the protein sequences in helix and non-helix residues, TOPTMH was not able to effectively differentiate between inside and outside loops and thus could not perform similar to MEMSAT3.

TOPTMH Performance on the Static Benchmark. The performance of TOPTMH on the static benchmark is shown on Table 6. The TOPTMH results shown in these tables correspond to the results obtained using the HMM-SVM-D7+HP topology prediction approach. From these results we see that TOPTMH achieved the highest Q_{ok} score of 86% for the high-resolution sequences and the highest Q_2 scores of 84% and 90% for the high- and low-resolution sequences, respectively. Moreover, TOPTMH has performed about 7% better in TMH prediction than both MEMSAT3 and Phobius. Note that even though HMMTOP2 achieved $Q_{htm}^{\%obs}$ and $Q_{htm}^{\%prd}$ scores that were higher than the corresponding scores achieved by TOPTMH, its Q_{ok} score of is lower than that achieved by TOPTMH. This is due to the fact that even though HMMTOP2 identified more TMH segments in total than TOPTMH, it was not as successful in predicting proteins for which all of the TMH segments were identified correctly.

6 Conclusions

In this paper we developed the TOPTMH method to predict the transmembrane α-helix topology using sequence information. TOPTMH uses PSI-BLAST constructed profiles and hydrophobicity information within a hybrid SVM- and HMM-based framework. This novel hybrid method captures the power of SVM-based models to discriminate between the helical and non-helical residues with the power of HMMs to identify length-dependent topological structures. Experiments on the Phobius and MÖLLER datasets showed that TOPTMH achieves high per-residue and per-segment accuracies and that on an independent static benchmark it outperforms existing state-of-the-art methods such as PHDpsihtm08 [25], HMMTOP2 [28], MEMSAT3 [10], Phobius [17], and TopPred2 [7].

References

1. Altschul, S.F., Madden, L.T., Schffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Research 25(17), 389–402 (1997)
2. Berman, H.M., Battistuz, T., Bhat, T.N., Bluhm, W.F., Bourne, P.E., Burkhardt, K., Feng, Z., Gilliland, G.L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J.D., Zardecki, C.: The protein data bank. Nucleic Acids Res. 28, 235–242 (2000)
3. Chen, C.P., Rost, B.: State-of-the-art in membrane protein prediction. Appl. Bioinformatics 1(1), 21–35 (2002)
4. The UniProt Consortium. The universal protein resource (uniprot). Nucleic Acids Res. 35, D193–D197 (2007)
5. Elofsson, A., von Heijne, G.: Membrane protein structure: Prediction versus reality. Annu. Rev. Biochem. 76, 125–140 (2007)
6. Engelman, D.M., Steitz, T.A., Goldman, A.: Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. Annual Review of Biophysics and Biophysical Chemistry 15, 321–353 (1986)
7. Engelman, D.M., Steitz, T.A., Goldman, A.: Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. Annu. Rev. Biophys. Chem. 15, 321–353 (1986)
8. Hessa, T., Kim, H., Bihlmaier, K., Lundin, C., Boekel, J., Andersson, H., Nilsson, I., White, S.H., von Heijne, G.: Recognition of transmembrane helices by the endoplasmic reticulum translocon. Nature 433(7024), 377–381 (2005)
9. Joachims, T.: Making large-Scale SVM Learning Practical. In: Joachims, T. (ed.) Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge (1999)
10. Jones, D.T.: Improving the accuracy of transmembrane protein topology prediction using evolutionary information. Bioinformatics 23(5), 538–544 (2007)
11. Kanungo, T.: UMDHMM: Hidden Markov Model Toolkit. Cambridge University Press, Cambridge (1999)
12. Karypis, G.: Yasspp: better kernels and coding schemes lead to improvements in protein secondary structure prediction. Proteins 64(3), 575–586 (2006)
13. Kernytsky, A., Rost, B.: Static benchmarking of membrane helix predictions. Nucl. Acids Res. 31(13), 3642–3644 (2003)

14. Kernytsky, A., Rost, B.: Static benchmarking of membrane helix predictions. Nucleic Acids Res. 31(13), 3642–3644 (2003)
15. Klabunde, T., Hessler, G.: Drug design strategies for targeting g-protein-coupled receptors. Chem. Bio. Chem. 3, 928–944 (2002)
16. Kyte, J., Doolittle, R.F.: A simple method for displaying the hydropathic character of a protein. Journal of Molecular Biology 157(1), 105–132 (1982)
17. Kll, L., Krogh, A., Sonnhammer, E.L.L.: A combined transmembrane topology and signal peptide prediction method. Journal of Molecular Biology 338, 1027–1036 (2004)
18. Kll, L., Sonnhammer, E.L.L.: Reliability of transmembrane predictions in whole-genome data. FEBS Lett. 532(3), 415–418 (2002)
19. Liu, J., Rost, B.: Comparing function and structure between entire proteomes. Protein Sci. 10, 1970–1979 (2001)
20. Lo, A., Chiu, H.-S., Sung, T.-Y., Lyu, P.-C., Hsu, W.-L.: Enhanced membrane protein topology prediction using a hierarchical classification method and a new scoring function. J. Proteome Res. 7(2), 487–496 (2008)
21. Oberai, A., Ihm, Y., Kim, S., Bowie, J.U.: A limited universe of membrane protein families and folds. Protein Sci. 15(7), 1723–1734 (2006)
22. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Proceedings of the IEEE, vol. 77, pp. 257–286 (1989)
23. Rangwala, H., Karypis, G.: frmsdpred: Predicting local rmsd between structural fragments using sequence information. Proteins (February 2008)
24. Rangwala, H., Kauffman, C., Karypis, G.: A generalized framework for protein sequence annotation. In: Proceedings of the NIPS Workshop on Machine Learning in Computational Biology (2007)
25. Rost, B., Fariselli, P., Casadio, R.: Topology prediction for helical transmembrane proteins at 86 accuracy. Protein Sci. 5(8), 1704–1718 (1996)
26. Sonnhammer, E.L.L., von Heijne, G., Krogh, A.: A hidden markov model for predicting transmembrane helices in protein sequences. In: Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology, pp. 175–182 (1998)
27. Tusndya, G.E., Simon, I.: Principles governing amino acid composition of integral membrane proteins: application to topology prediction. Journal of Molecular Biology 283(2), 489–506 (1998)
28. Tusndya, G.E., Simon, I.: The hmmtop transmembrane topology prediction server. Bioinformatics 17(9), 849–850 (2001)
29. von Heijne, G.: Membrane protein structure prediction hydrophobicity analysis and the positive-inside rule. Journal of Molecular Biology 225(2), 487–494 (1992)
30. von Heijne, G.: Formation of transmembrane helices in vivo–is hydrophobicity all that matters? The Journal of general physiology 129(5), 353–356 (2007)
31. Wallin, E., von Heijne, G.: Genome-wide analysis of integral membrane proteins from eubacterial, archaean, and eukaryotic organisms. Protein Sci. 7(4), 1029–1038 (1998)
32. Zhou, H., Zhou, Y.: Predicting the topology of transmembrane helical proteins using mean burial propensity and a hidden-markov-model-based method. Protein Sci. 12, 1547–1555 (2003)

Learning to Predict One or More Ranks in Ordinal Regression Tasks

Jaime Alonso, Juan José del Coz, Jorge Díez, Oscar Luaces, and Antonio Bahamonde

Artificial Intelligence Center. University of Oviedo at Gijón, Asturias, Spain
www.aic.uniovi.es

Abstract. We present nondeterministic hypotheses learned from an ordinal regression task. They try to predict the true rank for an entry, but when the classification is uncertain the hypotheses predict a set of consecutive ranks (an interval). The aim is to keep the set of ranks as small as possible, while still containing the true rank. The justification for learning such a hypothesis is based on a real world problem arisen in breeding beef cattle. After defining a family of loss functions inspired in Information Retrieval, we derive an algorithm for minimizing them. The algorithm is based on posterior probabilities of ranks given an entry. A couple of implementations are compared: one based on a multiclass *SVM* and other based on Gaussian processes designed to minimize the linear loss in ordinal regression tasks.

1 Introduction

In the last few years, ordinal regression has become an important issue in Machine Learning research. See [1] and [2] for a state of the art introduction. The aim of ordinal regression is to find hypotheses able to predict classes or *ranks* that belong to a finite ordered set. Applications include Information Retrieval [3], Natural Language Processing [4], collaborative filtering [5], finances [6], and user preferences [7].

The approach presented in this paper explores a new kind of predictions in ordinal regression. We shall build hypotheses that try to predict the true rank for an entry, but when the classification is uncertain the hypotheses predict an interval of ranks. The aim is to return a set of consecutive ranks, such that the set is as small as possible, while still containing the true rank. As we shall learn hypotheses for ordinal regression tasks with multiple outcomes, like nondeterministic automata, we shall call them *nondeterministic ordinal regressors*. From another point of view, these hypothesis could be called set-valued predictors.

Predictors of more than one class are not completely new. Given an error ϵ, the so called confidence machines, make *conformal predictions* [8]: they produce a set of labels containing the true class with probability greater than $1-\epsilon$. Other approaches arose in the context of hierarchical organization of biological objects: predicting gene functions [9], or mapping biological entities into ontologies [10].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 39–54, 2008.

In the next Section we shall show the usefulness of these nondeterministic hypotheses in a real world application context: the assessment of muscle proportion in carcasses of beef cattle. This is an important question in cattle breeding since this proportion determines, on the one hand, the prices to be obtained by carcasses, and on the other hand, the genetic value of animals to select studs for the next generation.

We formalize the problem of nondeterministic predictions as a special kind of *Information Retrieval*. Thus, we define a family of loss functions F_β and derive an algorithm for minimizing this loss. The algorithm needs the estimation of posterior probabilities of ranks given the entries. Then, we compare a couple of implementations built on the estimations provided by a *SVM* [11], and by the Gaussian approach of [1] devised for ordinal regression tasks.

The last Section of the paper presents an exhaustive set of experiments carried out in order to test the performance of the nondeterministic approach. Thus, in addition to the beef cattle learning task, we shall use 24 datasets publicly available that were previously used in ordinal regression tasks [1,12].

2 The Round Profile of Bovines

The problem that motivated the research reported in this paper arose when we were trying to make reliable predictions of the value of the carcasses of beef cattle. This learning task was proposed by ASEAVA, the Association of Breeders of a beef breed of the North of Spain, *Asturiana de los Valles*. This is a specialized breed with many double-muscled individuals; their carcass have dressing percentages over 60%, with muscle content over 75%, and with a low (8%) percentage of fat [13]. The market target of these carcasses is made up of those consumers that prefer lean meat without any marbling [7,14,15].

Even if the animals are not going to be slaughtered, the prediction of carcass value of a beef cattle is interesting since it can be considered as a kind of assessment that is useful for breeders to select the progenitors of the next generation. Thus, the ICAR (International Committee for Animal Recording) acknowledges as a good practice the recording of live animal assessments; since these assessments are a description of an animal's morphology that reveals part of its economic value.

The records so obtained can be used for the evaluation of programs of genetic selection of dairy, dual purpose and specialized beef breeds. The growth of the scores over years of selection for specific goals can be seen as a measure of the success of the selection policy. On the other hand, when the assessed traits are heritable, the scores can be directly used for selection purposes given that they are capturing part of animal's genetic value.

Traditionally, the assessment procedures were based on *visual* appreciations of well trained technicians that had to rank a number of morphological characteristics that include linear lengths of significant parts of animals' bodies. Although this process has been used successfully, it is clear that there is a problem with the repeatability of the assessments; not only between assessors, but

Fig. 1. The assessment of the round profile of a beef cattle is a measurement of the roundness of the lines drawn in the pictures. Thus, the leftmost cow in the top row is a paradigm of the animals which have rank 1, while the following are representative examples of ranks 2, 3 and 4 respectively.

even within assessors scoring the same animal in different times. In order to overcome these difficulties, we developed a new assessment method described in [16] that is almost completely repeatable and can be carried out using just 3 lengths (in centimeters) plus the appreciation of the curvature of the round profile (see the curves in Figure 1). For this learning task we used a kernel based method described in [17].

The aim of the assessment of round profiles is to rank the muscularity of animals. Therefore, it is a very important attribute for describing beef cattle. However, the curvature of the round profile is assessed by *visual* appreciations of experts. But visual appreciations is a source of problems. Thus, for instance, in [18], the authors describe an experiment in which a set of expert graders were asked to rank a collection of mushrooms into three major and eight subclasses of commercial quality. Grader consistency was assessed by repeated classification (four repetitions) of two 100-mushroom sets. Grader repeatability ranged from 6% to 15% misclassification.

Therefore, returning to beef cattle, to ensure the repeatability of the whole process, we should skip the subjective appreciation of the rank of round profiles. Thus, a new learning task arises: to estimate this rank from repeatable live animal descriptions. For this purpose, we built a dataset with 891 pairs of animal descriptions (6 lengths in centimeters of their bodies, live weight, and sex) and ranks (in a scale of 1-4). To ensure a uniform criterion, the first author of this paper measured and ranked the round profile of the 891 live animals.

But this is a difficult learning task. The classification accuracy achieved by a multiclass SVM was 77%; the implementation used was a probabilistic *libsvm* [11]. These results are not improved when using a learner specially devised for ordinal regression tasks. Thus, using the MAP approach of [1], the accuracy was 76%. The details about how we obtained these scores are included in the last section devoted to report a number of experimental results. On the other hand, we shall prove that a nondeterministic hypothesis contains the true class more than 84% of cases, while the average number of ranks predicted is just 1.21 or 1.30 depending of the learner used.

Moreover, the nondeterministic approach is more useful than the plain deterministic one in this problem for several reasons. First, the reliability of hypothesis predictions is higher than in the deterministic case. Therefore, when the hypothesis predicts only one rank, the estimation of the rank is very probably the true one. Second, when the prediction is an interval of more than one rank, we can appeal to a more expensive procedure to finally decide the true class. In this case, we may turn to an *actual* expert, or we can wait until the natural growth of the animal make the classification more clear.

However, sometimes even a nondeterministic prediction may be useful to discard an animal as stud for the next generation: a prediction of $[1, 2]$ must imply a poor genetic value as meat producer, provided that the hypothesis is sufficiently reliable.

3 Formal Framework

Let $\mathcal{X}$ be an input space, and $\mathcal{Y}$ a finite set of ordered ranks. Without any loss of generality, we can assume that $\mathcal{Y} = \{1, \ldots, k\}$ for a given k. We shall consider an ordinal regression task given by a training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ drawn from an unknown distribution $Pr(X, Y)$ from the product $\mathcal{X} \times \mathcal{Y}$. Within this context, we propose the following.

Definition 1. *A* nondeterministic hypothesis *is a function h from the input space to the set of non-empty intervals (subsets of consecutive ranks) of $\mathcal{Y}$; in symbols,*

$$h : \mathcal{X} \longrightarrow Intervals(\mathcal{Y}). \tag{1}$$

The aim of such a learning task is to find a nondeterministic hypothesis h from a space $\mathcal{H}$ that optimizes the *expected prediction performance (or risk)* on samples S' independently and identically distributed (i.i.d.) according to the distribution $Pr(X, Y)$:

$$R^{\Delta}(h) = \int \Delta(h(\boldsymbol{x}), y)\; d(Pr(\boldsymbol{x}, y)), \tag{2}$$

where $\Delta(h(\boldsymbol{x}), y)$ is a loss function that measures the penalty due to the prediction $h(\boldsymbol{x})$ when the true value is y.

In nondeterministic ordinal regression, we would like to favor those decisions of h that contain the true ranks, and a smaller rather than a larger number of

ranks. In other words, we interpret the output $h(\boldsymbol{x})$ as an imprecise answer to a query about the right rank of an entry $\boldsymbol{x} \in \mathcal{X}$. Thus, the nondeterministic ordinal regression can be seen as a kind of *Information Retrieval* task for each entry.

In Information Retrieval, performance is compared using different measures in order to consider different perspectives. The most frequently used are the *Recall* (proportion of all relevant documents that are found by a search) and *Precision* (proportion of retrieved documents that are relevant). The harmonic average of the two amounts is used to capture the goodness of a hypothesis in a single measure. In the weighted case, the measure is called F_β. The idea is to measure a tradeoff between *Recall* and *Precision*.

For further references, let us recall the formal definitions of these Information Retrieval measures. Thus, for a prediction of a nondeterministic hypothesis $h(\boldsymbol{x})$ with $\boldsymbol{x} \in \mathcal{X}$, and a rank $y \in \mathcal{Y}$, we can compute the following contingency matrix, where $z \in \mathcal{Y}$,

	$y = z$	$y \neq z$
$z \in h(\boldsymbol{x})$	a	b
$z \notin h(\boldsymbol{x})$	c	d

(3)

where each entry (a, b, c, d) is the number of times that happens the corresponding combination of memberships. Thus, notice that a can only be 1 or 0, depending on whether the rank y is in the prediction $h(\boldsymbol{x})$ or not; b is the number of ranks different from y included in $h(\boldsymbol{x})$; $c = 1 - a$; and d is the number of ranks different from y that are not in $h(\boldsymbol{x})$.

According to the matrix (Eq. 3), if h is a nondeterministic hypothesis, and $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$, we have the next definitions.

Definition 2. *The* Recall *in a* query *(i.e. an entry $\boldsymbol{x}$) is defined as the proportion of relevant ranks (y) included in $h(\boldsymbol{x})$:*

$$R(h(\boldsymbol{x}), y) = \frac{a}{a + c} = a = 1_{y \in h(\boldsymbol{x})}. \tag{4}$$

Definition 3. *The* Precision *is defined as the proportion of retrieved ranks in $h(\boldsymbol{x})$ that are relevant (y):*

$$P(h(\boldsymbol{x}), y) = \frac{a}{a + b} = \frac{1_{y \in h(\boldsymbol{x})}}{|h(\boldsymbol{x})|}. \tag{5}$$

In other words, given an hypothesis h, the *Precision* for an entry $\boldsymbol{x}$, that is $P(h(\boldsymbol{x}), y)$, is the probability of finding the true rank (y) of the entry ($\boldsymbol{x}$) by randomly choosing one of the ranks of $h(\boldsymbol{x})$.

Finally, the tradeoff is formalized by

Definition 4. *The F_β, in general is defined by*

$$F_\beta(h(\boldsymbol{x}), y) = \frac{(1 + \beta^2)a}{(1 + \beta^2)a + b + \beta^2 c}. \tag{6}$$

Table 1. For an entry $\boldsymbol{x}$ with rank 1, $(y = 1)$, *Precision*, *Recall*, F_1, and F_2 for different predictions of a nondeterministic classifier h

$h(\boldsymbol{x})$	*Precision*	*Recall*	F_1	F_2
$[1, 2, 3]$	0.33	1	0.50	0.71
$[1, 2]$	0.50	1	0.67	0.83
$[1]$	1	1	1	1
$[2, 3, 4]$	0	0	0	0

Thus, for a nondeterministic classifier h and a pair $(\boldsymbol{x}, y)$,

$$F_\beta(h(\boldsymbol{x}), y) = \begin{cases} \frac{1+\beta^2}{\beta^2+|h(\boldsymbol{x})|} & \textit{if} \quad y \in h(\boldsymbol{x}) \\ 0 & \textit{otherwise.} \end{cases} \tag{7}$$

The most frequently used F-measure is F_1. For ease of reference, let us state that

$$F_1(h(\boldsymbol{x}), y) = \frac{2_{y \in h(\boldsymbol{x})}}{1 + |h(\boldsymbol{x})|}. \tag{8}$$

To illustrate the use of F-measures of an entry, let us consider an example. If we assume that the true rank of an entry $\boldsymbol{x}$ is 1, $(y = 1)$, then, depending on the value of $h(\boldsymbol{x})$, Table 1 reports the *Recall*, *Precision*, F_1, and F_2. We observe that the reward attached to a prediction containing the true rank with another extra rank ranges from 0.667 for F_1 to 0.833 for F_2; while the amounts are lower when the prediction includes 2 extra ranks.

Once we have the definition of F_β for individual entries, it is straightforward to extend it to a test set. So, when S' is a test set of size n, the average loss on it will be computed by

$$\begin{aligned} R^{\Delta}(h, S') &= \frac{1}{n}\sum_{j=1}^{n} \Delta(h(\boldsymbol{x'_j}), y'_j) = \frac{1}{n}\sum_{j=1}^{n} \left(1 - F_\beta(h(\boldsymbol{x'_j}), y'_j)\right) \\ &= \frac{1}{n}\sum_{j=1}^{n} \left(1 - \frac{1+\beta^2}{\beta^2 + |h(\boldsymbol{x'_j})|} 1_{y'_j \in h(\boldsymbol{x'_j})}\right). \end{aligned} \tag{9}$$

It is important to realize that for a deterministic hypothesis h this amount is the average "0/1" loss, since all predictions are singletons, $|h(\boldsymbol{x})| = 1$. Thus, the nondeterministic loss used here is a generalization of the error rate of deterministic classifiers. Furthermore, the average *Recall* and *Precision* on test sets can be similarly defined. In this case, the *Recall* on a test set is the proportion of times that $h(\boldsymbol{x'})$ includes y', and is thus a generalization of the deterministic *accuracy*.

Algorithm 1. The nondeterministic ordinal regressor **nd_•**, an algorithm for computing the prediction with one or more ranks for an entry $\boldsymbol{x}$ provided that the posterior probabilities of ranks are given

Input: object description $\boldsymbol{x}$
Input: $\{Pr(j|\boldsymbol{x}) : j = 1, .., k\}$
for i =1 to k **do**
$[Start^+(i),\ Pr\_Inter^+(i)] = \max \left\{ \sum_{t=j}^{j+i-1} Pr(t|\boldsymbol{x}) : j = 1, \ldots, k-i+1 \right\}$
/* $Pr\_Inter^+(i)$ is the highest probability of the intervals of length i */
/* This interval starts at class $Start^+(i)$ */
end for
$Min = argmin \left\{ 1 - \frac{1+\beta^2}{\beta^2+i} Pr\_Inter^+(i) : i = 1, \ldots k \right\}$
return $\left[Start^+(Min), Start^+(Min) + Min - 1\right]$

4 How to Learn Intervals of Ranks with Posterior Probabilities

In the general ordinal regression setting presented in Section 3, let $\boldsymbol{x}$ be an entry of the input space $\mathcal{X}$, and let us now assume that we know the conditional probabilities of ranks given the entry, $Pr(rank = j|\boldsymbol{x})$ for $j \in \{1, \ldots, k\}$. In this context, we wish to define

$$h(\boldsymbol{x}) = Z \in Intervals\{1, \ldots, k\} \tag{10}$$

that minimizes the risk defined in (Eq. 2) when we use the nondeterministic loss given by F_β (Eqs. 6, 7, and 9). We shall prove that such $h(\boldsymbol{x})$ can be computed by Algorithm 1.

Proposition 1. *(Correctness) If the conditional probabilities $Pr(j|\boldsymbol{x})$ are known, Algorithm 1 returns the nondeterministic prediction $h(\boldsymbol{x})$ that minimizes the risk given by the loss $1 - F_\beta$.*

Proof. To minimize the risk (Eq. 2), it suffices to compute

$$\Delta_{\boldsymbol{x}}(Z) = \sum_{y \in \mathcal{Y}} \Delta(Z, y) Pr(y|x) = \sum_{y \in \mathcal{Y}} \left(1 - F_\beta(Z, y)\right) Pr(y|x), \tag{11}$$

with $Z \in Intervals\{1, \ldots, k\}$. Then, we only have to define

$$h(\boldsymbol{x}) = argmin\{\Delta_{\boldsymbol{x}}(Z) : Z \in Intervals\{1, \ldots, k\}\}. \tag{12}$$

First we shall prove that when Z is an interval of length i, say $Z = [s, s + i - 1]$, given $\boldsymbol{x}$, the value of Equation (11) can be expressed in function of i and the probability of the interval. In fact, with a probability of $1 - Pr(Z|\boldsymbol{x})$, we expect a loss of 1: the *true* rank will not be one of the interval Z. On the other hand,

with the probability of Z, the *true* rank will be in $h(\boldsymbol{x})$, and therefore the loss will be 1 minus the F_β of the prediction $h(\boldsymbol{x}) = Z = [s, s+i-1]$. In symbols,

$$\begin{aligned}\Delta_{\boldsymbol{x}}(Z) &= \Delta_{\boldsymbol{x}}\left([s, s+i-1]\right) \\ &= \left(1 - \sum_{j=s}^{s+i-1} Pr(j|\boldsymbol{x})\right) 1 + \left(\sum_{j=s}^{s+i-1} Pr(j|\boldsymbol{x})\right)\left(1 - \frac{1+\beta^2}{\beta^2+i}\right) \\ &= 1 - \frac{1+\beta^2}{\beta^2+i} \sum_{j=s}^{s+i-1} Pr(j|\boldsymbol{x}). \end{aligned} \tag{13}$$

Therefore, the interval of length i with lower loss starts at $Start^+(i)$ according to the Algorithm 1, moreover, its loss is

$$1 - \frac{1+\beta^2}{\beta^2+i} Pr\_Inter^+(i). \tag{14}$$

Thus if Min is the length that gives rise to the lowest loss, the output of the Algorithm is the value of Equation 12 as we wanted to prove.

In practice, posterior probabilities are not known: they are estimated by algorithms that frequently try to optimize the classification accuracy of a hypothesis that returns the class with the highest probability. In other words, probabilities are discriminant values instead of thorough descriptions of the distribution of classes in a learning task. Therefore, the actual role of β in Algorithm 1 is that of a parameter that fixes the thresholds to decide the number of ranks to predict. Hence, like other parameters, β should be tuned in order to achieve optimal results. Thus, depending of the learning task and the probabilistic learner, to reach the highest F_1 scores, it might be necessary to use in Algorithm 1 a value of β different from 1.

5 Experimental Results

In this section we report the results of a set of experiments designed to evaluate the nondeterministic learners proposed in this paper. The aim is to compare, on the one hand, the F_1 scores of well known deterministic learners and their nondeterministic counterparts. It may be argued that these comparisons are not completely fair since the F_1 score tolerates predictions of more than one rank, where it is easier to include the true one. In any case, we report these comparisons in order to test the capabilities of nondeterministic versions to achieve slightly better F_1 scores than their deterministic counterparts. On the other hand, we shall compare the *Recall* and size of predictions attained by nondeterministic learners.

Additionally, since we are dealing with ordinal regression tasks, we check the performance of nondeterministic algorithms in linear loss (sometimes called MAD, mean absolute deviation, or MAE, mean absolute error). For this purpose,

Table 2. Description of the datasets used in the experiments. The classes are real numbers, and they were discretized in 5 or 10 equal-frequency bins. The splits in train/test were suggested by the experiments reported in [1].

Dataset	#Attributes	#Train	#Test
pyrimidines	27	50	24
triazines	60	100	86
Wisconsin bc	32	150	44
machine cpu	6	150	59
auto mpg	7	200	192
stock	9	300	650
Boston	13	300	206
abalone	8	300	3877
bank	32	300	7892
computer	21	300	7892
California	8	300	20340
census	16	300	22484

we must assume singleton predictions; thus, we shall consider the center of each interval as the prediction attached to every interval $h(\boldsymbol{x})$. The idea is to consider that each rank r can be interpreted as the interval $[r - 0.5, r + 0.5]$ in the real line; thus, a prediction of, say, $[3, 4]$ represents the real interval $[2.5, 4.5]$, and the center point is 3.5.

We used two kinds of learning tasks. In addition to the dataset of beef cattle profiles explained in Section 2, we used a collection of 12 benchmarks (Table 2) that were originally used for metric regression learning tasks. They are publicly available at Luís Torgo's repository[1]. When they were used for ordinal regression in papers like [1,12], the continuous class values were discretized. We used versions with five and ten bins with the same frequency of training examples. The resulting rank values were ordered according to the original metric classes.

To compare the performance of different approaches, we randomly split each data set into training/test partitions. Table 2 reports the characteristics of these datasets and the sizes of splits. The partition was repeated 20 times independently.

Since the nondeterministic approach proposed in this paper is based on the estimation of posterior probabilities of ranks, we used two alternative methods for this stage. First, we used a multiclass *SVM* that estimates the probability of each class given an entry; the implementation used was *libsvm* [11]. The nondeterministic version built from it, following Algorithm 1, was called *nd_SVM*. Second, we used the *MAP* approach of [1] that was devised for ordinal regression tasks. It provides estimations of posterior probabilities using Gaussian processes. The nondeterministic counterpart was called *nd_MAP*. The use of *MAP* in the experiments required reduced sizes of training sets (Table 2) similar to those used in [1]. Nevertheless, the computational requirements of *SVM* would allow us to use *nd_SVM* in tasks of bigger sizes.

[1] http://www.liaad.up.pt/∼ltorgo/Regression/DataSets.html

Parameter setting. With the *SVM* we used a *rbf* kernel. To set the regularization parameter C and the rbf kernel parameter σ, we performed a grid search using a 2-fold cross validation repeated 5 times. The initial search was done with $C \in \{10^{-3}, \ldots, 10^{3}\}$ (respectively $\sigma \in \{10^{-3}, \ldots, 10^{2}\}$) varying the exponent in steps of 1. Let C' and σ' be the best parameters found; then followed a fine search from $C' - 0.8$ (respectively $\sigma' - 0.8$) to $C' + 0.8$ (respectively $\sigma' + 0.8$) with a step of 0.2. Additionally, for *nd_SVM* we searched within $\beta \in \{0.5, 1, 1.5\}$, while the fine search explored the best $\beta - 0.2$, and the best $\beta + 0.2$. We looked for a β, instead of simply using $\beta = 1$, since we wished to compensate any possible inaccuracy in the estimation of probabilities.

The *MAP* learner was used with its default parameters, and no additional tuning was required. The nondeterministic version *nd_MAP* used the search for β of the *nd_SVM*.

The scores achieved in F_1 are shown in Table 3. The nondeterministic learner based on *MAP* bears favorable comparison with the learner based on *SVM*. Thus, *nd_MAP* wins in 18 out of 24 datasets, while *nd_SVM* only wins 3 times out of 24; most of these victories are statistically significant using a Wilcoxon rank sum test of 1-tail over the 20 trials. Comparing the performance over the 24 datasets, we also appreciate significant differences (using a Wilcoxon test with $p < 0.01$) in favor of *nd_MAP*. Therefore, the nondeterministic version of *MAP* outperforms the version based on *SVM* in F_1 when we are using sizes of training sets similar to those showed in Table 2. In the comparison of deterministic versus nondeterministic, in all cases the nondeterministic version outperforms its deterministic counterpart; all but one cases are statistically significant with $p < 0.01$.

The scores in *Recall* are reported in Table 4. Again *nd_MAP* wins in 17 out of 24 datasets, while *nd_SVM* only wins 4 times out of 24; however, now the differences are not so frequently significant. To compare *Recall* scores with those achieved by the deterministic versions, let us remember that for deterministic algorithms, the proportion of successful predictions (accuracy) is also the F_1 and the *Recall*. Therefore, comparing the last two columns of Table 3 and the *Recall* columns of Table 4, we appreciate that the nondeterministic learners outperform the deterministic versions. Thus, in 5 bins datasets, the differences are about 0.24, while in 10 bins datasets the differences are even higher: about 0.31. These results are logical since nondeterministic predictions have more opportunities to include the true ranks.

The average sizes of predictions are shown in the last two columns of Table 4. Here we observe that in the learning tasks of 5 bins these sizes in average are below 2, while with 10 bins, the predictions used more than 3 ranks in average.

The explanation for these facts is straightforward. The nondeterministic algorithms tend to accumulate as many ranks as they are allowed by the F_1; thus, in tasks in which the deterministic learners have a poor performance, the corresponding nondeterministic learner may include more ranks in their predictions than in easier tasks. And it is clear that the learning tasks with 5 bins are easier than versions with 10 bins.

Table 3. F_1 scores of the two nondeterministic algorithms and their deterministic counterparts. The results are the averages over 20 trials. In bold face we emphasize the highest score of each dataset. Additionally we test the statistical significance of some interesting differences: between *nd_MAP* and *nd_SVM* (see the first column labeled by *si.*), *nd_MAP* versus *MAP* (second *si.* column), and *nd_SVM* versus *SVM* (last *si.* column). The symbols † (respectively ‡) show that differences are statistically *significant* using a threshold of 0.05 (respectively 0.01) in a Wilcoxon rank sum test.

# Bins	Dataset	*nd_MAP*	(si.)	*nd_SVM*	*MAP*	(si.)	*SVM*	(si.)
5	pyrimidines	**0.58**	‡	0.52	0.57		0.45	‡
	triazines	**0.40**		0.39	0.34	‡	0.30	‡
	Wisconsin bc	**0.38**	‡	0.34	0.29	‡	0.26	‡
	machine cpu	**0.66**	‡	0.64	0.60	‡	0.59	‡
	auto mpg	**0.73**	‡	0.69	0.72	‡	0.67	‡
	stock	0.86	†	**0.87**	0.86	‡	0.86	‡
	Boston	**0.71**	‡	0.68	0.68	‡	0.66	‡
	abalone	**0.53**		**0.53**	0.47	‡	0.47	‡
	bank	**0.50**	‡	0.47	0.44	‡	0.40	‡
	computer	**0.71**	‡	0.71	0.69	‡	0.68	‡
	California	**0.57**		**0.57**	0.52	‡	0.52	‡
	census	**0.53**	‡	0.51	0.48	‡	0.46	‡
	Average (5 b)	0.597		0.576	0.555		0.527	
10	pyrimidines	**0.35**	‡	0.27	0.28	‡	0.19	‡
	triazines	**0.23**		**0.23**	0.16	‡	0.16	‡
	Wisconsin bc	**0.21**	‡	0.19	0.15	‡	0.13	‡
	machine cpu	**0.46**		0.45	0.36	‡	0.37	‡
	auto mpg	**0.51**	‡	0.47	0.44	‡	0.35	‡
	stock	0.73	‡	**0.76**	0.70	‡	0.74	‡
	Boston	0.47	†	**0.48**	0.41	‡	0.42	‡
	abalone	**0.35**	‡	0.34	0.28	‡	0.27	‡
	bank	**0.31**	‡	0.28	0.24	‡	0.20	‡
	computer	**0.53**	‡	0.51	0.48	‡	0.45	‡
	California	**0.39**	‡	0.37	0.32	‡	0.30	‡
	census	**0.34**	‡	0.32	0.27	‡	0.25	‡
	Average (10 b)	0.407		0.388	0.342		0.320	
	Average all	0.502		0.482	0.449		0.423	

Considering the performance over all datasets, we can only find significant differences in *Recall* with $p < 0.06$; while the differences in size of predictions are definitively not significant.

Finally, Table 5 shows the scores achieved in linear loss. This is a relevant measure since we are dealing with ordinal regression learning tasks. Although the nondeterministic algorithms were not designed to improve the linear loss, we observe a good performance. Let us recall here that *MAP* is a state of the art learner for these tasks. In datasets of 5 bins, *MAP* wins *nd_MAP* 8 times out of 12, with only 2 times out of 12 victories for *nd_MAP*. While *nd_MAP* wins 7 times out of 12, against 4 wins out of 12 for *MAP*. The result is that differences over the 24 datasets are not statistically significant.

On the other hand, in linear loss, *nd_MAP* outperform *nd_SVM* in most of the datasets with differences statistically significant, see Table 5. The performance

Table 4. Scores of *Recall* and average ***size*** of predictions ($|h(\boldsymbol{x})|$) for nondeterministic algorithms. Notice that for deterministic algorithms, the proportion of successful predictions (accuracy) is also the F_1 and the *Recall* (see Table 3). The best scores for each dataset are in bold. When the differences are statistically significant in a Wilcoxon rank sum test, they are marked with † (threshold of 0.05) or ‡ (0.01).

		Recall			aver. $\lvert h(\boldsymbol{x})\rvert$		
# Bins	Dataset	*nd_MAP*	(si.)	*nd_SVM*	*nd_MAP*	(si.)	*nd_SVM*
5	pyrimidines	0.71		**0.73**	**1.53**	‡	1.95
	triazines	**0.78**	†	0.71	3.04		**2.75**
	Wisconsin bc	0.79	†	**0.83**	**3.21**	‡	3.84
	machine cpu	**0.84**	†	0.80	1.67		**1.61**
	auto mpg	**0.80**		**0.80**	**1.22**	‡	1.34
	stock	**0.92**	†	0.91	1.16	‡	**1.12**
	Boston	**0.80**		0.78	**1.30**		1.35
	abalone	**0.75**	‡	0.70	2.02	‡	**1.80**
	bank	**0.74**	†	0.77	**2.08**	‡	2.45
	computer	**0.82**		0.81	1.35		**1.34**
	California	**0.78**		0.75	1.83	†	**1.74**
	census	**0.74**		0.73	1.93		**1.89**
	Average (5 b)	**0.788**		0.777	**1.862**		1.931
10	pyrimidines	**0.73**		0.71	**3.32**	‡	4.74
	triazines	**0.73**		0.68	5.80		**5.17**
	Wisconsin bc	0.68	‡	**0.86**	**5.42**	‡	8.18
	machine cpu	**0.80**		0.77	2.75		**2.69**
	auto mpg	**0.76**		**0.76**	**2.04**	‡	2.29
	stock	**0.83**		0.82	1.33	‡	**1.21**
	Boston	0.66		**0.72**	**1.92**	‡	2.20
	abalone	**0.68**	†	0.64	3.35	‡	**3.06**
	bank	**0.70**		0.69	**3.85**	‡	4.33
	computer	**0.74**	†	0.72	**1.95**		1.97
	California	**0.69**		0.67	2.92		**2.84**
	census	**0.66**		**0.66**	**3.17**		3.23
	Average (10 b)	0.721		**0.725**	**3.151**		3.492
	Average all	**0.755**		0.751	**2.507**		2.712

over all datasets is again statistically significant with $p < 0.01$. Finally, let us point out that the nondeterministic *nd_SVM* outperforms (significantly with $p < 0.01$) *SVM*.

Profiles. Tables 6 summarize the scores achieved in the learning task described in Section 2. We used two datasets of sizes 300 and 500. The scores are quite similar for both sizes. Almost always, *nd_SVM* outperforms *nd_MAP* significantly ($p < 0.01$) in F_1 and linear loss, although the scores are quite similar. On the other hand, *nd_MAP* is superior to *nd_SVM* in *Recall*, but again the scores are similar and the significance is only achieved with $p < 0.1$ in one of the datasets. The differences are clearly significant ($p < 0.01$) in the size of the predictions; *nd_SVM* only requires an average of 1.21 or 1.22 ranks to reach a proportion of 85% of predictions that contain the true rank.

Table 5. Linear loss scores of the two nondeterministic algorithms and their deterministic counterparts. The lowest scores for each dataset are highlighted in bold. When the differences are statistically significant in a Wilcoxon rank sum test, they are marked with † (threshold of 0.05) or ‡ (0.01).

# Bins	Dataset	*nd_MAP*	(si.)	*nd_SVM*	*MAP*	(si.)	*SVM*	(si.)
5	pyrimidines	0.55	‡	0.65	**0.52**	†	0.77	‡
	triazines	**1.07**		1.10	1.18	‡	1.34	‡
	Wisconsin bc	**1.12**	‡	1.19	1.36	‡	1.44	‡
	machine cpu	0.48		0.48	**0.45**	‡	0.46	†
	auto mpg	0.32	‡	0.37	**0.30**	‡	0.35	‡
	stock	0.16	‡	0.15	**0.14**	‡	**0.14**	‡
	Boston	0.36	‡	0.41	**0.34**	‡	0.40	†
	abalone	**0.72**	†	0.73	**0.72**		0.75	‡
	bank	**0.76**	‡	0.83	**0.76**		0.90	‡
	computer	0.37		0.38	**0.34**	‡	0.36	‡
	California	0.63	†	0.62	**0.59**	‡	0.60	‡
	census	0.73	‡	0.77	**0.71**	‡	0.79	†
	Average (5 b)	**0.607**		0.640	0.618		0.692	
10	pyrimidines	**1.24**	‡	1.88	1.31	†	2.25	‡
	triazines	2.23		**2.21**	2.79	‡	2.58	‡
	Wisconsin bc	**2.44**		2.48	3.08	‡	3.14	‡
	machine cpu	**0.95**		0.96	**0.95**		1.04	‡
	auto mpg	0.74	‡	0.80	**0.69**	‡	1.38	†
	stock	0.34	‡	0.30	0.31	‡	**0.28**	‡
	Boston	1.06	‡	**0.87**	1.04	‡	0.89	‡
	abalone	**1.47**	‡	1.53	1.57	‡	1.71	‡
	bank	**1.55**	‡	1.76	1.65	‡	2.11	‡
	computer	0.75	‡	0.76	**0.72**	‡	0.80	‡
	California	**1.22**	‡	1.26	1.26	‡	1.36	‡
	census	**1.49**	‡	1.58	1.58	‡	1.71	‡
	Average (10 b)	**1.289**		1.365	1.413		1.603	
	Average all	**0.948**		1.002	1.015		1.147	

6 Conclusions

We have presented a new kind of ordinal regressors: they are able to predict a variable number of consecutive ranks (an interval of ranks) for each entry. We call such set-valued hypotheses nondeterministic regressors. Roughly speaking, the approach presented in this paper addresses the problem of deciding what to predict when it is possible to envision that the label returned by a learning algorithm is uncertain. The utility of these predictions was illustrated in the context of a real world application: the assessment of muscle proportion in beef cattle carcasses.

After presenting the formal framework as a kind of Information Retrieval, we proposed a family of loss functions for nondeterministic ordinal regression: the complementary of F_β measures. Next we derived an algorithm to minimize such loss functions provided we know the posterior probabilities of each rank given the entry to be ranked. To check the influence of the estimation of conditional

Table 6. Profiles (see Section 2). Dataset characterizations, and scores achieved by deterministic and nondeterministic algorithms. All differences are statically significant ($p < 0.01$) but those achieved in *Recall* ($p < 0.1$).

Dataset	#Attributes	#Train	#Test
Profiles 500	8	500	391
Profiles 300	8	300	591

	Dataset	*nd_MAP* (si.)		*nd_SVM*	*MAP* (si.)		*SVM* (si.)	
F_1	Profiles 500	0.78	‡	**0.79**	0.76	‡	**0.77**	‡
	Profiles 300	0.77	‡	**0.78**	0.76	‡	**0.77**	‡
Linear loss	Profiles 500	0.28	†	**0.28**	0.29	†	**0.27**	
	Profiles 300	0.29	‡	**0.28**	0.30	‡	**0.27**	‡

Dataset	*Recall*		aver. $\lvert h(\boldsymbol{x})\rvert$		
	nd_MAP (si)	*nd_SVM*	*nd_MAP* (si.)		*nd_SVM*
Profiles 500	**0.85**	0.84	1.28	‡	**1.21**
Profiles 300	**0.85**	0.84	1.30	‡	**1.22**

probabilities we compared two implementations. The first one, *nd_SVM* is based on a probabilistic *SVM*, while the second (*nd_MAP*) is built on *MAP*, a learner specialized in ordinal regression learning tasks that uses Gaussian processes to estimate posterior probabilities.

The experiments reported in the previous Section show that *nd_MAP* outperforms *nd_SVM* in almost all measures of performance. Therefore, it is clear the importance of having good probability estimations. However, *MAP* is slower than *SVM*, and it is not possible to handle datasets of medium or large size with the approach based on Gaussian processes.

We think that the main goal of nondeterministic ordinal regressors is not to achieve similar (in fact better) F_β than their deterministic counterpart. We would like to emphasize the dramatic improvement in the proportion of predictions that include the true rank, when the price to be paid for that increase is usually a tiny proportion of predictions with more than one rank.

Acknowledgements

The research reported here is supported in part under grant TIN2005-08288 from the MEC (Ministerio de Educación y Ciencia of Spain). The authors acknowledge the help and partial support of the Association of Breeders of *Asturiana de los Valles*, ASEAVA.

References

1. Chu, W., Ghahramani, Z.: Gaussian Processes for Ordinal Regression. The Journal of Machine Learning Research 6, 1019–1041 (2005)
2. Cardoso, J., da Costa, J.: Learning to Classify Ordinal Data: The Data Replication Method. The Journal of Machine Learning Research 8, 1393–1429 (2007)
3. Joachims, T.: Training linear SVMs in linear time. In: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD). ACM, New York (2006)
4. Shen, L., Joshi, A.: Ranking and Reranking with Perceptron. Machine Learning 60(1), 73–96 (2005)
5. Yu, S., Yu, K., Tresp, V., Kriegel, H.: Collaborative ordinal regression. In: Proceedings of the 23^{rd} International Conference on Machine Learning (ICML 2006), pp. 1089–1096 (2006)
6. Agarwal, A., Davis, J., Ward, T.: Supporting ordinal four-state classification decisions using neural networks. Information Technology and Management 2(3), 5–26 (2001)
7. del Coz, J.J., Bayón, G.F., Díez, J., Luaces, O., Bahamonde, A., Sañudo, C.: Trait selection for assessing beef meat quality using non-linear SVM. In: Saul, L.K., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 17 (NIPS 2004), pp. 321–328. MIT Press, Cambridge (2005)
8. Shafer, G., Vovk, V.: A Tutorial on Conformal Prediction. Journal of Machine Learning Research 9, 371–421 (2008)
9. Clare, A., King, R.: Predicting gene function in Saccharomyces cerevisiae. Bioinformatics 19(2), 42–49 (2003)
10. Kriegel, H., Kroger, P., Pryakhin, A., Schubert, M.: Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects. In: Proc. 4^{th} SIAM Int. Conf. on Data Mining, pp. 102–114 (2004)
11. Wu, T.F., Lin, C.J., Weng, R.C.: Probability estimates for multi-class classification by pairwise coupling. The Journal of Machine Learning Research 5, 975–1005 (2004)
12. Chu, W., Keerthi, S.S.: New approaches to support vector ordinal regression. In: Proceedings of the ICML 2005, Bonn, Germany, pp. 145–152 (2005)
13. Piedrafita, J., Quintanilla, R., Sañudo, C., Olleta, J., Campo, M., Panea, B., Renand, G., Turin, F., Jabet, S., Osoro, K., Oliván, M.C., Noval, G., García, P., García, M., Cruz-Sagredo, R., Oliver, M., Gispert, M., Serra, X., Espejo, M., García, S., López, M., Izquierdo, M.: Carcass quality of 10 beef cattle breeds of the Southwest of Europe in their typical production systems. Livestock Production Science 82(1), 1–13 (2003)
14. Luaces, O., Bayón, G.F., Quevedo, J.R., Díez, J., del Coz, J.J., Bahamonde, A.: Analyzing sensory data using non-linear preference learning with feature subset selection. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 286–297. Springer, Heidelberg (2004)
15. Díez, J., del Coz, J.J., Sañudo, C., Albertí, P., Bahamonde, A.: A kernel based method for discovering market segments in beef meat. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 462–469. Springer, Heidelberg (2005)
16. Alonso, J., Bahamonde, A., Villa, A., Castañón, Á.R.: Morphological assessment of beef catle according to carcass value. Livestock Science 107, 265–273 (2007)

17. Bahamonde, A., Bayón, G.F., Díez, J., Quevedo, J.R., Luaces, O., del Coz, J.J., Alonso, J., Goyache, F.: Feature subset selection for learning preferences: A case study. In: Greiner, R., Schuurmans, D. (eds.) Proceedings of the International Conference on Machine Learning (ICML 2004), pp. 49–56 (2004)
18. Kusabs, N., Bollen, F., Trigg, L., Holmes, G., Inglis, S.: Objective measurement of mushroom quality. In: Proc. New Zealand Institute of Agricultural Science and the New Zealand Society for Horticultural Science Annual Convention (1998)

Cascade RSVM in Peer-to-Peer Networks

Hock Hee Ang, Vivekanand Gopalkrishnan,
Steven C.H. Hoi, and Wee Keong Ng

Nanyang Technological University, Singapore

Abstract. The goal of distributed learning in P2P networks is to achieve results as close as possible to those from centralized approaches. Learning models of classification in a P2P network faces several challenges like scalability, peer dynamism, asynchronism and data privacy preservation. In this paper, we study the feasibility of building SVM classifiers in a P2P network. We show how cascading SVM can be mapped to a P2P network of data propagation. Our proposed P2P SVM provides a method for constructing classifiers in P2P networks with classification accuracy comparable to centralized classifiers and better than other distributed classifiers. The proposed algorithm also satisfies the characteristics of P2P computing and has an upper bound on the communication overhead. Extensive experimental results confirm the feasibility and attractiveness of this approach.

1 Introduction

Peer-to-peer (P2P) network is a large network of entities interconnected in a point-to-point manner. The Internet as a large point-to-point network of computers is a P2P network. P2P computing refers to computations performed in a P2P network of computers where there is no absolute centralized control. In recent years, data mining in P2P networks has attracted much attention as increasingly many applications have distributed data, from which useful knowledge may be mined. For instance, clustering and classification of peer data may reveal networks of cliques in social networks, and classification of network traffic could provide valuable information about network intrusions or usage behaviors.

The primary goal of learning in a P2P network is to achieve learning result that is as close as possible to that of a centralized approach. Learning models of classification (also clustering) is faced with several challenges [6]. In a P2P setting, learning algorithms need to take into account the scalability issue (Can the algorithm be computed when there are millions of peers?), peer dynamism (Can the algorithm deal with the availability and unavailability of data as peers connect and disconnect from the network?), asynchronism (Can the algorithm produce sufficiently accurate results without global synchronization?), and data privacy (Can the algorithm preserve the privacy of peer data when learning the global model?).

In this paper, we study the feasibility of learning in a P2P network in the context of learning classifiers. In particular, we are interested to know how Support

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 55–70, 2008.

Vector Machines (SVM) perform in a P2P network, as SVM is a class of powerful classification and regression algorithms. Some of the weaknesses of SVM is in its memory and computational requirements, which increase with the size of training data set. A proven approach for alleviating such requirements, while not degrading classification performance, is the cascade SVM approach, where the data set is partitioned into smaller chunks and small-scale SVM learning is performed on these chunks. Support vectors obtained from these small-scale SVM models are combined with other chunks to derive better support vectors and improve the final set of support vectors.

We show how the cascading of SVM learning can be mapped to a P2P network of data propagation. As network communication is a performance issue, we show how the sending of support vectors from peer to peer can be improved using the Reduced Support Vector Machine (RSVM) [13] approach. Our proposed P2P SVM provides a method for learning classifiers in a P2P network that has classification accuracy comparable to a centralized classifier, yet satisfies the characteristics of P2P computing and has an upper bound on the communication overhead. We have implemented P2P SVM and experimental results confirm the feasibility and attractiveness of using this approach.

We focus mainly on the classification accuracy, scalability in terms of computation and total bandwidth usage, effects of data distribution and imbalanced class distribution. The other issues affecting classification in P2P networks such as peer dynamism, data privacy, security and different types of P2P networks will be studied in future works.

Our contributions in this paper are as follows: (1) We demonstrate the feasibility of cascade SVM in a P2P network, which to the best of our knowledge, is the first such attempt. (2) Our proposed P2P SVM has classification accuracy comparable to a centralized solution and better than other classification approaches in a P2P network. (3) In order to reduce data propagation cost in a P2P setting, we show how an upper bound can be derived to control the network communication overhead.

The organization of this paper is as follows: Section 2 describes related work. Section 3 introduces our proposed approach to perform SVM in a P2P network. Section 4 describes our experiments and the last section concludes the paper.

2 Background and Related Work

Many well-known classifiers such as decision tree, nearest neighbor classifier, artificial neural networks, Bayes classifier and support vector machine (SVM) work well with small datasets, but fail to maintain reasonable time and cost benefits on large datasets common to many domains. Hence, researchers have developed alternative methods such as selective sampling [2, 13, 14], parallelized and distributed learning [3, 9, 12, 15, 19, 23], in order to learn from such large datasets.

Breiman [2] introduced the pasting of Ivotes (or Rvotes) that trains an ensemble of classifiers, each built from a subset of data that has been selectively

sampled using out-of-bag estimation (or randomly sampled for Rvotes). Lee and Mangasarian [13] presented the Reduced Support Vector Machines (RSVM) approach that solves the SVM optimization problem using a randomly selected smaller portion of the whole dataset. Lin and Lin [14] studied several implementations of RSVM and showed that for problems with dense support vectors, RSVM significantly reduces the training time that is required for a small drop in accuracy compared with the SVM solution.

Parallelized and distributed algorithms represent another paradigm to solve the large dataset problem. These algorithms can be broadly classified as ensemble and cascade approaches. In general, these approaches split a large problem into smaller easily solvable sub-problems, and then combine their results. A useful side-effect of approaches under this paradigm is that they can also be used on *naturally distributed* data, saving the cost of moving data to a single location for training using a centralized solution.

Distributed ensemble approaches can be further divided as voting [5, 12] and meta-learning [3, 7, 17] approaches. Voting approaches build an ensemble of classifiers and then perform final classification based on the votes of all classifiers in the ensemble. For instance, Lazarevic and Obradovic [12] provide a distributed boosting framework that exchanges training statistics and performs weighted majority voting to obtain the final prediction. On the other hand, Chawla *et al.* [5] present a distributed version of Ivotes (DIvotes) and Rvotes (DRvotes) that works by first splitting the data, then performing Ivotes on each subset, and obtaining the final hypothesis by majority voting. The advantage that DIvotes has over distributed boosting is that no communication is required among the distributed parities during the training phase, thus, significantly reducing the communication overhead.

Meta-learning is in essence the learning of meta-attributes generated from various learners (classifiers). Chan and Stoflo [3] present an arbiter tree approach that builds various levels of classifiers and combines the results using arbitration rules. More recently, Pfahringer *et al.* [17] proposed to landmark various learning algorithms in order to characterize the classification problems and find the relationship between classifiers, whereas Džeroski and Ženko [7] presented their approach of using the model tree induction to learn the meta-level features generated.

Cascade learning was proposed mainly for the purpose of speeding up computation. Tveit and Engum [19] pioneered the work on cascading SVM by providing a heap-based tree topology framework for parallelizing the computation of Proximal SVM. Since then, a number of works have focused on cascade SVM [9, 15, 23]. Lu *et al.* [15] presented and compared various ways of cascading SVM. Zhang *et al.* [23] further improved cascade SVM by examining various ways of performing feedback to obtain a global optimal solution. Graf *et al.* [9] also provided a cascade SVM algorithm with feedback and formally proved the convergence of the algorithm.

2.1 Learning in P2P Networks

In recent years, there has been increasing interest in classification problems in P2P networks. A P2P network consists of a set of k peers $P = \{p_1, \ldots, p_k\}$, where all peers function equally as both servers and clients. However, a P2P environment possesses unique characteristics that introduce challenges for the classification task. These characteristics include scalability, peer dynamism, data dynamism, asynchronism and privacy and security [6]. P2P networks can be considered as a massively distributed environment as the number of peers, k, in the network usually exceeds hundreds or thousands. In addition, these peers may leave and join the network anytime, and the data they possess may change frequently. Due to the size of the network, it is not feasible to perform synchronization considering the network latency and bandwidth. If data exchange among peers is involved, privacy and security may also pose concerns.

Based on data propagation, existing P2P classification approaches can be categorized as 1) model propagation [18] and 2) test data propagation [8, 16] approaches. Model propagation approaches build local classifiers on each peer and then propagate the model to other peers. The peers can then use the collected models for performing classification. In the latter approach, a peer only propagates test instances to other peers, which in turn classify these instances and return results to the requesting peer. The model propagation approaches generally incur more communication cost during the model construction phase, which exacerbates when the classification model frequently changes. However, under this approach, classification of test instances is faster and peers have more freedom on how the models can be used (e.g., perform meta-learning using the models).

Siersdorfer and Sizov [18] have proposed a framework for classifying web documents in a P2P environment. The algorithm trains a local classifier and propagates it to other peers. Each peer then uses the received models to construct a meta model for performing classification. Although the paper states that the propagated model should be a compressed representation of the local data set, it neither provides details on how this may be achieved, nor on how the models may evolve with the addition of new data. Furthermore, the tuning of the global model to improve accuracy requires synchronization among peers, which increases communication cost.

On the contrary, test instance propagation approaches are not affected by frequent changes of models and does not incur communication cost during construction of models. However, classification tasks are slower since requests have to be made to the P2P network, and if these tasks are frequent, the communication cost can be comparable to that of the model propagation.

Gorodetskiy *et al.* prototyped an agent-based, service-oriented P2P distributed classification approach [8]. However, the focus of the paper is not on the classification task, but to provide a proof-of-concept implementation and to explore the issues that may exist in the agent-based, service-oriented P2P network.

More recently, Luo *et al.* [16] proposed a P2P classification approach by pasting of small votes. In this approach, each peer pastes small bites to build local

classifiers until the error between subsequent models falls below a certain threshold. The final classification is then performed by sending classification requests to all peers based on an optimal communication protocol.

3 Approach

In this section, we present our proposed approach, illustrating the design process and finally providing a complexity analysis. Our approach based on the cascade SVM paradigm, is specifically designed for the P2P network, addressing the additional constraints not found in the general distributed and parallel computing environment. The three basic processes in cascade SVM are: 1)build an SVM for each of the local data, then iteratively 2) propagate and 3) merge the models to create an improved SVM until all subsets have been combined. Let us now examine cascade SVM and our proposed approach in detail.

3.1 Cascade SVM

In cascade SVM, the algorithm starts by building SVM using local data. The purpose of using SVM (as well as merging) is to filter out as many non support vectors as early as possible, to reduce the time and space complexity required to efficiently build the global solution. However, using standard SVM may generate quite a high number of support vectors. Since our approach requires propagation of models in the P2P network, these large number of support vectors result in a high communication cost. Hence, algorithms based on standard SVM are usually not viable. Therefore, our criteria for building local classifiers changes from being able to effectively filter out redundant data, to being able to extract a very small set of representative data.

3.2 P2P Cascade RSVM

Based on the above considerations, we employ an approximate SVM solution - RSVM, which reduces the number of support vectors, for the task. The disadvantage of using RSVM is that the resulting cascade SVM cannot produce a global optimal solution. By global optimal solution, we refer to the solution produced by SVM and cascade SVM with feedback/synchronization, which however, is infeasible to achieve, since the convergence to the global optimal solution requires synchronization among all peers (for the validation process). As the number of support vectors in a SVM has extensive influence on the memory and training time, being able to reduce the number of support vectors greatly improves the training speed and lowers the memory requirements. However, since the SVM decision hyperplane is constructed from these support vectors, reducing the number of support vectors may also reduce the classification accuracy. Despite this, it has been found that RSVM can use a very small subset to represent the whole data, with only a slight drop in classification accuracy compared to traditional SVM [16]. Hence, usage of RSVM does not cause any serious drawback.

Since peer data constantly change in a P2P network, a set of new training data is treated as a new peer's dataset, and goes through the same processes as the existing local data. This addresses the data dynamism issue, allowing incremental learning. Although, our approach allows incremental learning, decremental learning or removal of data is not addressed, as this concerns the issue of concept drift and is not within the scope of this paper.

After the model is generated, it is propagated to other peers. Despite the main disadvantage of high communication cost (effect reduced as stated above), model propagation provides a way to counter the peer dynamism constraint. With model propagation, even when peers go offline, their models still exist on other peers on the P2P network (provided they have successfully propagated to other peers before they went offline). This allows sharing of models between peers which were not present on the P2P network at the same time, which is an important factor for maintaining high classification accuracy within the P2P network. In addition, our approach ensures that models are only collected/merged once to prevent duplication. Besides these, model propagation guarantees achieving a local optimal solution with cascade SVM, since it becomes possible to validate using the peers' models, and the high duplication rate of models allows higher throughput for the transfer of models.

Similar to the automatic document organization approach, model propagation in our approach can be implemented separately from the building of the classifier. This allows our approach to be deployed in any type of P2P network increasing its flexibility. By viewing the models as files in a P2P network, we can map the problem of model propagation in P2P network to the file propagation problem in P2P network, which has been extensively studied. For our approach, we utilize the UPTReC [21] algorithm, because it provides a probabilistic guarantee in file consistency which helps to ensure that models can be properly propagated within the P2P network. Experiments [21] show that UPTReC can reduce up to 70% overhead messages compared with other existing techniques.

Models are collected as peers propagate them in the P2P network. In contrast to the cascade SVM, since we do not have control over how, when and how many of the peers' models will be collected, we perform the merging process as follows. All models collected within t duration are merged together in a single process and then merged with the peer's local optimal SVM. In the two extreme cases, given $t = 0$, this simply implies that each time a peer's model is collected, it is merged immediately with the last cascaded model, and given $t =$ the time required to collect models of all uncollected peers in the P2P network, all newly collected models are merged in a single process with the previously cascaded RSVM. For example, consider that peer j receives three new models from other peers before time t after startup, and two other new models between time t and $2t$. Therefore, at time t from startup, peer j will merge the three newly received models with the latest local model, and at time $2t$ from startup, it will merge the two newly received models with the latest cascaded model. This process is illustrated in Figure 1, and the training phase of the proposed approach is given in Algorithm 1.

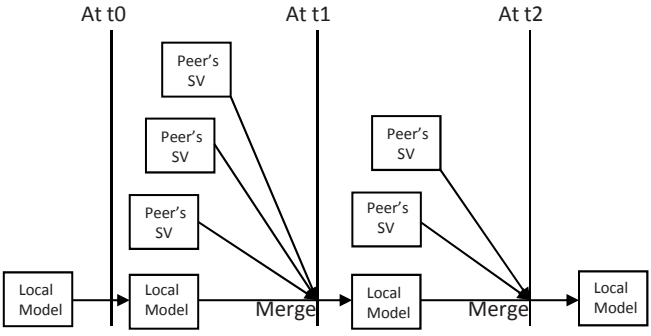


Fig. 1. Illustration of merging support vectors

To summarize, the main differences between existing cascading approaches and our proposed P2P Cascade RSVM lies in the use of RSVM, and in the ad-hoc merging of the collected models due to the high dynamism of the P2P networks. This greatly reduces the communication overhead for distributing data after distributed and parallelized construction of local models. These extensions of cascade SVM make it feasible to learn from the P2P environments and even achieve results comparable to centralized solution, while reducing computation and communication costs.

Algorithm 1. P2P Cascade RSVM algorithm for peer p_i

input: the percentage p of support vectors to use,
the duration t to wait before merging,
local training data D_i

$SSV_i = \{\}$
$PSV_i = \{\}$
training data $T = \emptyset$
Train local classifier model M_i using RSVM on D_i
Propagate the support vectors SV_i of M_i to other peers
while *true* **do**
  while *waiting time* $< t$ **do**
    foreach SV_j *of peer* p_j *received* **do**
      if $SV_j \notin SSV_i$ *and* $SV_j \notin PSV_i$ **then**
        $PSV_i = PSV_i \cup SV_j$
  if PSV_i *is not empty* **then**
    $T =$ support vectors of M_i
    forall $SV \in PSV_i$ **do**
      $T = T \cup SV$
    $M_i =$ SVM model trained using T
    $SSV_i = SSV_i \cup PSV_i$
    $PSV_i = \{\}$

3.3 Model Propagation Cost

In our approach, since the number of support vectors directly determines the size of the model to be propagated, the communication cost can also be greatly reduced. Furthermore, by specifying the size of the support vectors, either absolutely or as a percentage of the training data, we can give an upper bound on the communication cost of the construction of the cascade SVM as follows. Let N be the total number of peers in the P2P network, l be the total size (in terms of number of vectors) of the problem and $s, s < 1$ be the percentage of the problem to be used as support vectors. Then the upper bound of the total communication cost, c, required for all peers to obtain the global model is

$$c = N \cdot l \cdot s \tag{1}$$

for a two-class problem. For a multi class problem, where the number of classes is nc, and using the one-against-one strategy for SVM classification, the cost is as follows:

$$c = N \cdot l \cdot s \cdot (nc \cdot (nc - 1)/2) \tag{2}$$

3.4 Computation Cost

Considering the following SVM optimization problem [20]:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + C(\sum_{i=1}^{l} \xi_i^2) \\ \text{subject to} \quad & y_i(w^T w \phi(x_i) + b) \geq 1 - \xi_i \end{aligned} \tag{3}$$

given that x_i is a feature vector and y_i is the corresponding label of a training set, where $x_i \in R^n$ and $y_i \in \{1, -1\}$. As $\phi(x)$ maps x into a higher dimensional space, we can simply solve its dual, which is a quadratic programming problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T (Q + \frac{I}{2C}) \alpha - e^T \alpha \\ \text{subject to} \quad & y^T \alpha = 0, \\ & 0 \leq \alpha_i, i = 1, \ldots, l \end{aligned} \tag{4}$$

where the number of variables equals l, e is the unity vector, Q is an l by l positive semi-definite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel function. Computing the kernel function $K(x_i, x_j)$ for every training instance costs $O(l^2)$ and solving (4) costs $O(l^3)$.

However, for RSVM based on Least-Square SVM, we are only required to solve

$$\min_{\tilde{\alpha}} f(\tilde{\alpha}) = \frac{1}{2C} \tilde{\alpha}^T \tilde{\alpha} + \tilde{\alpha}^T (\tilde{Q}^T \tilde{Q}) \tilde{\alpha} - 2 e^T \tilde{Q} \tilde{\alpha} + e^T e \tag{5}$$

where f can minimized by finding the solution of $\frac{\partial f}{\partial \tilde{\alpha}_i} = 0, i = 1, \ldots, m$:

$$\frac{1}{C}\tilde{\alpha} + 2\tilde{Q}^T\tilde{Q}\tilde{\alpha} - 2\tilde{Q}Te = 0, \tag{6}$$

$$(\tilde{Q}^T\tilde{Q} + \frac{I}{2C})\tilde{\alpha} = \tilde{Q}^T e \tag{7}$$

a positive definite linear system of size m, where m is the size of the subset R used in RSVM, $\tilde{\alpha}$ are the coefficients of the separating hyperplane and $\tilde{Q} = [Q_{:,R} \quad y]$. Hence, the total time complexity for RSVM is $O(lm^2)$. For the complete formulation, refer to [14].

To analyze the time complexity of our approach, we have to examine the process of building the local model and merging of the collected models. Given a P2P network with N peers, and total training data of size l, let the size of local data for peer i be l_i, and the percentage of local data to be used for RSVM be $s, s < 1$. Then, the size of the subproblem to solve in RSVM is $m_i = l_i s$, and the time complexity for building a local model for peer i with RSVM is $O(l_i m_i^2)$. Since the size of the subproblem optimized by RSVM is already very small, and we have no prior knowledge of the amount of reduction that can be achieved by the optimization process, we assume that the size of the support vectors for the resulting models is the same as the size of the subproblem. Hence, after a peer constructs the local model (of size at most m_i), it propagates the model to other peers. The size of the support vectors collected from all peers is $m = \sum_1^N m_i$. If traditional SVM is used, the complexity of merging is $O(m^3)$. However, with other more efficient techniques such as SMO, cost of merging can be reduced, but in this case, we use the complexity of SVM to provide the upper bound. All in all, with $l_i m_i^2 << m^3$, the complexity of our proposed approach is $O(m^3)$.

Using centralized SVM and RSVM as comparison, we present a summary of the computation and communication costs of the various SVM based approaches in Table 1. It can be seen from Table 1 that our proposed approach has the least cost with respect to the centralized approaches.

Table 1. Summary of the training costs

Approach	Computation Cost	Communication Cost
SVM	$O(l^3)$	$O(l)$
RSVM	$O(lm^2)$	$O(l)$
P2P Cascade RSVM	$O(m^3)$	$O(m)$

4 Experiments and Result Analysis

Here, we present the experimental results on some large sized problems to simulate the problem size that may exist in a real P2P environment. First we describe the experimental setup. Then we compare the classification accuracy of centralized and existing P2P classification approaches, followed by a demonstration of

the effect of scalability, peers' data distribution and data class distribution on the various algorithms. Finally we illustrate the effect of the number of support vectors on the classification accuracy of our approach.

4.1 Experimental Setup

We used the covertype dataset and the waveform data generator available from the UCI repository [1]. For waveform, we generated 100,000 instances with 21 attributes. The covertype dataset was used further to generate a binary covertype dataset with class two versus all other classes. Summary of the datasets used is presented in Table 2. All attributes of the datasets were scaled to between -1 and 1.

Table 2. Summary of the datasets used in experiments

	Instances	Attributes	Classes
Binary Covertype	581,012	54	2
Covertype	581,012	54	7
Waveform	100,000	21	3

The experiments were conducted on a cluster of 16 machines, each with two Intel Dual Core Xeon 3.0GHz processors, 4 GB of Ram and connected by a gigabit ethernet.

The J48 algorithm (variant of the C4.5), from Weka [22] was used for the centralized classification and as the base classifier for the algorithm from [16]. In addition, we implemented the algorithm from [16], which we refer to as P2P Ivotes, in Java. We used the C-SVC algorithm from LIBSVM [4], in C++ as the centralized SVM solution, and used RSVM based on Least Square SVM algorithm from [14] in our approach which was implemented in C++.

In all P2P experiments, unless otherwise stated, we used 500 peers, and divided the data equally among them. We did not experiment with more peers since this would result in unrealistically small sizes for local peer data, which would adversely affect performance of the P2P approaches. For P2P Ivotes, bite size of 800 and λ of 0.02 were used. For SVM, SVM Ensemble and P2P Cascade RSVM, we used the RBF kernel, and for each dataset, the γ and C values were chosen using the model selection tool provided with LIBSVM on a 1 percent stratified sampled data of the whole dataset. For all datasets, we used 1 percent of the data as support vectors for our P2P cascade RSVM.

4.2 Classification Accuracy

In this experiment, we conducted a 10-fold cross validation using centralized RSVM, centralized J48, plurality voting on ensemble of J48, plurality voting on ensemble of SVM, P2P Ivotes and P2P Cascade RSVM on the binary covertype, covertype and waveform dataset. In order to train all peers on the same amount

Table 3. Tenfold cross-validation results

Dataset	Accuracy (%)					
	RSVM	J48	J48 Ensemble	SVM Ensemble	P2P Ivotes	P2P Cascade RSVM
Binary Covertype	71.97	**73.3**	58.26	54.16	58.72	72.93
Covertype	**68.16**	66.59	54.47	44.65	54.14	67.77
Waveform	99.8	**99.86**	99.62	99.79	99.78	99.61

Table 4. Average training time

Dataset	Time (secs)					
	RSVM	J48	J48 Ensemble	SVM Ensemble	P2P Ivotes	P2P Cascade RSVM
Binary Covertype	111.2	2357.9	159.04	48	326.56	**11.9**
Covertype	751.8	2501.64	238.12	**53.5**	378.38	126.4
Waveform	32	13.8	12	6.2	12	**0.4**

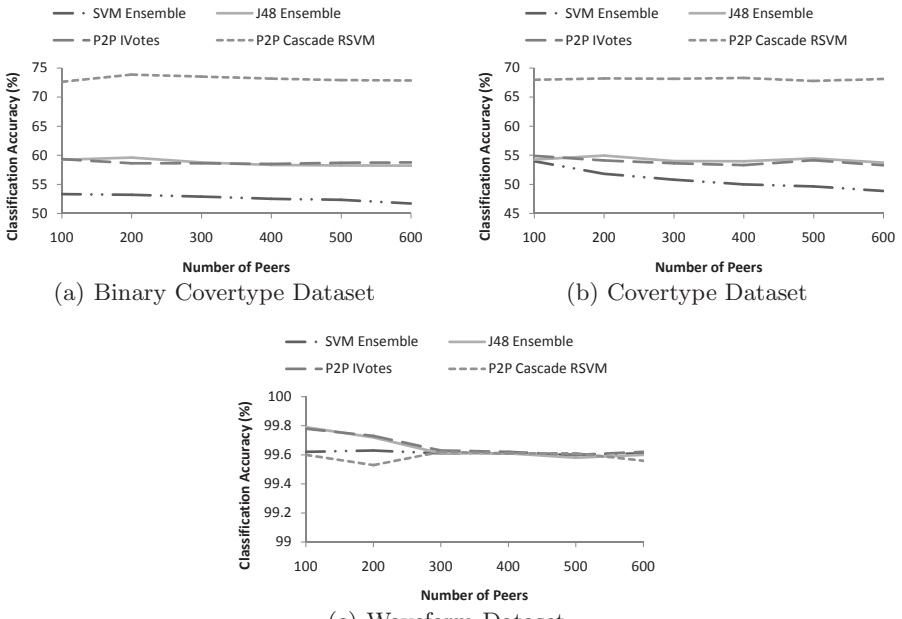
of data, we used 500 peers for binary covertype and covertype datasets and 100 peers for the waveform dataset. The classification accuracy and average training time taken are shown in Tables 3 and 4 respectively.

As shown in Table 3, our approach has accuracy comparable to the centralized solution on all datasets. Compared with other existing approaches, our approach exhibits similar accuracy on the waveform dataset, but has far better accuracy on the binary covertype and covertype datasets. In addition, our approach has the least training time for binary covertype and waveform dataset and second least for covertype dataset, which is probably due to the higher number of classes in the latter. We note that the P2P Ivotes results obtained by our experiments are dissimilar to those reported in [16], perhaps due to different methods of assigning peers' local training sets.

4.3 Scalability

To determine the scalability of the various P2P classification approaches, we varied the number of peers from 100 to 600 based on a 10-fold cross validation. For all approaches, the training data is divided equally among all peers with random class distribution.

As can be seen in Figure 2, our approach achieves significantly (based on student's t-test with p-value of 0.05) higher accuracy on the binary covertype and covertype data while producing similar accuracy on the waveform dataset. We observe in Figure 2(a) and 2(b), that the two covertype datasets show similar results, which is not surprising. For the Waveform dataset, none of the approaches seem to be affected by the number of peers that exist in the network. However, for both covertype datasets, all approaches except ours lose some accuracy when the number of peers increases. It is also noted that for all the datasets, the results of the J48 ensemble and the P2P Ivotes showed similar trends.



(a) Binary Covertype Dataset

(b) Covertype Dataset

(c) Waveform Dataset

Fig. 2. Effect of P2P network size on accuracy

We observed that some of the random data assignment resulted in a few peers not obtaining data from certain classes, which might explain the poor performance of the ensemble methods. This hypothesis will be verified in future work with further experiments.

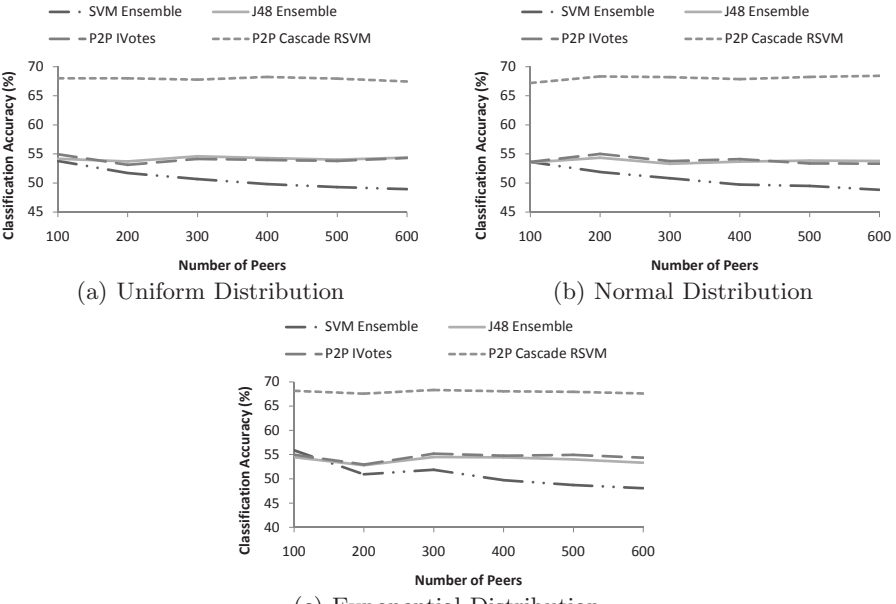
4.4 Peers' Data Distribution

Here we illustrate the effect of distribution of peers' data on the classification accuracy. From 100 to 600 peers, we randomly assign a subset of the data to each peer, where the size of the subset is based on exponential, uniform and normal distributions and test the accuracy using 10-fold cross validation. We have used the covertype dataset in this experiment.

As observed in Figure 3, the results for the different distributions do not seem to be very much different. Including the results from Figure 2(b), which is based on equal distribution, we conducted a student's t-test and found that there is actually no significant difference for each algorithm between the results of the different distributions. However, it would be interesting to see how peers dynamism can actually affect accuracy based on the different data size distribution.

4.5 Effect of Imbalanced Class Distribution

To see if the P2P classification approaches can deal with peers having data with imbalanced class distribution (natural class distribution of the whole dataset



(a) Uniform Distribution (b) Normal Distribution

(c) Exponential Distribution

Fig. 3. Effect of peers' data distribution on accuracy (covertype dataset)

remains unchanged), we purposely vary the class distribution of the data subset assigned to each peer. Using the binary covertype data, we modify the class distribution such that the class distribution has d percentage of skew compared to the natural class distribution. For example, if the natural class distribution is 60/40, a skew of $d = 10\%$ generates a modified class distribution of 65/35 for half of the peers and 55/45 for the other half of the peers. Although we modified the class distribution of the local training data, we still ensured that every peer received the same amount of data.

The results in Figure 4 show that our approach achieves better accuracy in the presence of imbalanced class distribution. Performing a student's t-test shows that the difference in accuracy between the other existing P2P approaches and our approach is significant with p-value of 0.05. Note that with the increase in percentage of skewness, the accuracy of the J48 ensemble and P2P Ivotes gradually decreases. However, the accuracies of our approach and SVM ensemble are not affected. Our approach is unaffected by the class imbalance perhaps due to the merging of support vectors that may have a rebalancing effect on the class distribution.

4.6 Size of Support Vectors

By restricting the number of support vectors used to build the SVM, we can limit the communication, computation and memory cost, albeit possibly at the

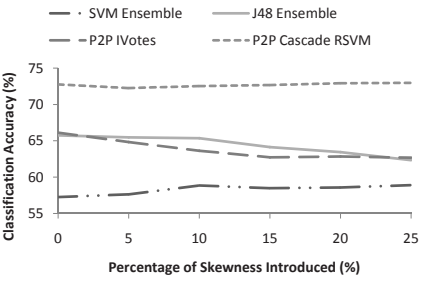


Fig. 4. Effect of imbalance class distribution on accuracy (binary covertype dataset)

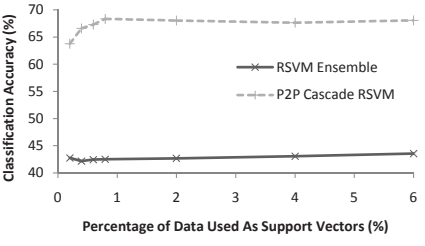


Fig. 5. Effect of support vector size constraint on accuracy (covertype dataset)

expense of classification accuracy. Here, we demonstrate the effect of the number of support vectors used on the accuracy. This experiment was conducted on the covertype data using an ensemble of RSVM and P2P cascade RSVM with 500 peers, by varying the percentage of support vectors used (subproblem size) from 0.2 to 6 percent.

From Figure 5, we note that when the percentage of support vectors used is too small (i.e. less than 1 percent), the classification accuracy is not stable and of unacceptable level. However, when the percentage of support vectors increases to above 1 percent, the increase in accuracy starts to plateau. Another point to note is that time and memory complexity of SVM is quadratic with respect to the number of support vectors. Therefore a low percentage of support vectors would be preferred but care must be taken to ensure that there are enough support vectors to represent the peers' local training data (which is dependent on the size of local training dataset).

5 Conclusion

In this paper, we study the problem of learning models of classification in a P2P network. We have proposed a combination of the cascade SVM and Reduced Support Vector approaches to learn classifiers in a P2P setting. Experimental results show that our proposed approach can learn classifiers with accuracies close to those of centralized approaches. Moreover, our approach also outperforms other

distributed models of classifier learning. The proposed approach scales with the size of the network, and accuracy is not affected by the number of peers. Also, we provide an upper bound on the massive communication overhead in P2P classification using the Reduced Support Vector approach to cap the number of support vectors computed. Overall, experimental results confirm the feasibility and attractiveness of using our approach. As part of future work, we will be exploring in detail, the effects of peer dynamism, cliques, and data privacy on the problem of learning in P2P networks. In addition, we will investigate unified kernel machines [11] and distributed active learning [10] techniques for enhancing classification performance.

References

[1] Asuncion, A., Newman, D.: UCI machine learning repository (2007)
[2] Breiman, L.: Pasting small votes for classification in large databases and on-line. Machine Learning 36(1-2), 85–103 (1999)
[3] Chan, P., Stolfo, S.: Toward parallel and distributed learning by meta-learning. In: AAAI Workshop in Knowledge Discovery in Databases, pp. 227–240 (1993)
[4] Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
[5] Chawla, N.V., Hall, L.O., Bowyer, K.W., Moore, T.E., Kegelmeyer, W.P.: Distributed pasting of small votes. In: Multiple Classifier Systems, pp. 52–61 (2002)
[6] Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed data mining in peer-to-peer networks. IEEE Internet Computing, Special issue on Distributed Data Mining 10(4), 18–26 (2006)
[7] Džeroski, S., Ženko, B.: Is combining classifiers with stacking better than selecting the best one? Machine Learning 54(3), 255–273 (2004)
[8] Gorodetskiy, V., Karsaev, O., Samoilov, V., Serebryakov, S.: Agent-based service-oriented intelligent P2P networks for distributed classification. In: International Conference on Hybrid Information Technology, pp. 224–233 (2006)
[9] Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: The cascade SVM. In: NIPS (2004)
[10] Hoi, S.C.H., Jin, R., Zhu, J., Lyu, M.R.: Batch mode active learning and its application to medical image classification. In: ICML, pp. 417–424 (2006)
[11] Hoi, S.C.H., Lyu, M.R., Chang, E.Y.: Learning the unified kernel machines for classification. In: SIGKDD, pp. 187–196 (2006)
[12] Lazarevic, A., Obradovic, Z.: Boosting algorithms for parallel and distributed learning. Distributed and Parallel Databases 11(2), 203–229 (2002)
[13] Lee, Y., Mangasarian, O.: RSVM: Reduced support vector machines. In: SIAM International Conference on Data Mining, pp. 00–07 (2001)
[14] Lin, K., Lin, C.: A study on reduced support vector machines. IEEE Transactions on Neural Networks 14(6), 1449–1459 (2003)
[15] Lu, B., Wang, K., Wen, Y.: Comparison of parallel and cascade methods for training support vector machines on large-scale problems. In: International Conference on Machine Learning and Cybernetics, pp. 3056–3061 (2004)
[16] Luo, P., Xiong, H., Lü, K., Shi, Z.: Distributed classification in peer-to-peer networks. In: SIGKDD, pp. 968–976 (2007)
[17] Pfahringer, B., Bensusan, H., Giraud-Carrier, C.G.: Meta-learning by landmarking various learning algorithms. In: ICML, pp. 743–750 (2000)

[18] Siersdorfer, S., Sizov, S.: Automatic document organization in a P2P environment. In: European Conference on IR Research, pp. 265–276 (2006)
[19] Tveit, A., Engum, H.: Parallelization of the incremental proximal support vector machine classifier using a heap-based tree topology. Technical report, IDI, NTNU, Trondheim, Norway (2003)
[20] Vapnik, V.N.: The nature of statistical learning theory. Springer, New York (1995)
[21] Wang, Z., Das, S.K., Kumar, M., Shen, H.: An efficient update propagation algorithm for P2P systems. Computer Communications 30(5), 1106–1115 (2007)
[22] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
[23] Zhang, J., Li, Z., Yang, J.: A parallel SVM training algorithm on large-scale classification problems. In: International Conference on Machine Learning and Cybernetics, pp. 1637–1641 (2005)

An Algorithm for Transfer Learning in a Heterogeneous Environment

Andreas Argyriou[1], Andreas Maurer[2], and Massimiliano Pontil[1]

[1] Department of Computer Science
University College London
Malet Place, WC1E London, UK
{a.argyriou,m.pontil}@cs.ucl.ac.uk
[2] Adalbertstrasse 55
D-80799 München, Germany
andreasmaurer@compuserve.com

Abstract. We consider the problem of learning in an environment of classification tasks. Tasks sampled from the environment are used to improve classification performance on future tasks. We consider situations in which the tasks can be divided into groups. Tasks within each group are related by sharing a low dimensional representation, which differs across the groups. We present an algorithm which divides the sampled tasks into groups and computes a common representation for each group. We report experiments on a synthetic and two image data sets, which show the advantage of the approach over single-task learning and a previous transfer learning method.

Keywords: Learning to learn, multi-task learning, transfer learning.

1 Introduction

Transfer learning uses the experience gathered from previous learning tasks in order to improve learning a new task. In the context of machine learning, past experience is provided by a collection of training sets, each sampled from a specific task. The underlying assumption is that the tasks belong to the same environment and share common properties. Uncovering these properties should thus enhance learning future tasks in the environment.

An important approach to transfer learning relies on the assumption that *all* the tasks are mutually related, in the sense that they share the same underlying representation, see [1,2,6,7,10,13] and references therein. This requirement may be too strong for heterogeneous environments. As an illustrative example, consider object recognition involving different geometric invariances, such as rotation, scaling, illumination etc., where only one invariance is relevant for any given task.

The main contribution of this paper is a method to learn and represent the structure of such a heterogeneous environment. Our method naturally extends a previous method for multi-task and transfer learning with linear representations

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 71–85, 2008.

[2,10]. Furthermore, we connect this approach to previous work in the context of spectral regularization [3] and collaborative filtering [11].

Previous work on task clustering [5,8,12,14] considers tasks to be related if the corresponding weight vectors are close to each other. In contrast, our approach assumes that tasks within the *same group* are related if their weight vectors span a low dimensional subspace. For example, in a binary classification task a target vector and its negative are far from each other in distance, but – lying in a one-dimensional subspace – closely related according to our assumption.

The paper is organized as follows. In Section 2, we introduce the transfer learning problem. In Section 3, we present our model for transfer learning over a heterogeneous environment. In Section 4 we describe the algorithmic implementation of the model. Next, in Section 5 we present numerical experiments with the algorithm. Finally, in Section 6 we present our conclusions.

2 Transfer Learning

We are interested in learning classification tasks, as they occur in a prescribed *environment*. For simplicity, we restrict ourselves to linear classification functions. However, our considerations apply to kernel methods as well as to regression or other learning problems.

Following [6], we regard an environment as a probability measure ρ on a set of learning tasks and, since the tasks we consider are described by weight vectors, we regard ρ as a probability measure on $\mathbb{R}^d$.

Information on the environment may be obtained by the following two-step procedure

- draw a weight vector $w \in \mathbb{R}^d$ from ρ
- generate a sample $\mathbf{z} \in \left(\mathbb{R}^d \times \{-1, 1\}\right)^m$ using w.

The vector w above corresponds to a classification function

$$f(x) = \operatorname{sign}\left(\langle w, x\rangle\right),$$

$x \in \mathbb{R}^d$. The sample (training set) $\mathbf{z} = ((x_1, y_1), \ldots, (x_m, y_m))$ is obtained by sampling the function f at m random locations $x_1, \ldots, x_m$ with labelling noise. The above procedure is then repeated n times, to yield a collection of n training sets

$$\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n).$$

Each of the $\mathbf{z}_t$ corresponds to a different classification task in the environment, for $t = 1, \ldots, n$. We have assumed, for simplicity, that the samples $\mathbf{z}_1, \ldots, \mathbf{z}_n$ have the same size, m, but what follows applies also when the sample sizes are different.

Transfer learning extracts structural knowledge from $\mathbf{Z}$, so as to enhance learning a future task drawn from the environment. This is particularly important when the number of examples for each task is relatively small in comparison to

the number of parameters. In this case, single-task learning – learning each task in isolation – leads to poor performance. However, if the tasks in the environment are related, transfer learning may work well [6,10]. Thus, the problem is ultimately that of uncovering and exploiting relationships between the tasks.

As a very simple example, suppose that all the tasks are equal. In this case, a good transfer learning algorithm would combine all training sets to learn a single task. A more realistic situation is one in which all the tasks' weight vectors are a linear combination of a few feature vectors. A good transfer learning algorithm would learn these feature vectors from the data $\mathbf{Z}$. A classical approach, which goes back to work on learning to learn and multi-task learning [6,7], is to search for such a low dimensional representation shared by all the tasks. Knowledge of the relevant features can reduce the burden of high dimensionality to the estimation of a small number of coefficients.

We consider a linear representation described by a matrix $T \in \mathbb{R}^{d\times d}$, which maps the raw representation x to the feature vector Tx. Our first step is to design a quantity which measures the performance of T relative to a task. If T is fixed, we learn a weight vector from a sample $\mathbf{z}$ by regularization, that is, we solve the problem

$$r(T,\mathbf{z}) = \min_{v\in\mathbb{R}^d} \left\{ \frac{1}{m}\sum_{i=1}^{m} \ell\left(\langle v, Tx_i\rangle, y_i\right) + \lambda\|v\|^2 + \|T\|_2^2 \right\} \tag{1}$$

where λ is a positive parameter, $\|v\|^2 = \langle v, v\rangle$, ℓ a loss function and $\|T\|_2$ denotes the Frobenius norm of matrix T. The minimizing vector in (1) – let us call it $v(T,\mathbf{z})$ – is then used in the classification of future inputs preprocessed by T. Note that, with this notation, the target vector w mentioned previously corresponds to $T^\top v$.

The term $\|T\|_2^2$ plays no role in finding $v(T,\mathbf{z})$, but it allows one to regard $r(T,\mathbf{z})$ as a measure of the learning performance of the representation T on the sample $\mathbf{z}$. Indeed, if the term $\|T\|_2^2$ were omitted, the quantity $r(\gamma T,\mathbf{z})$ would decrease in γ and would converge as $\gamma\to\infty$ to the minimal empirical error of a linear function on the data $\mathbf{z}$.

We now consider a set of tasks as represented by the multi-sample $\mathbf{Z}$. A good representation T is one which gives good learning performance, on average, over the tasks. Hence, we may define the quantity

$$R(T,\mathbf{Z}) = \frac{1}{n}\sum_{t=1}^{n} r(T,\mathbf{z}_t)$$

and learn T by solving the problem

$$\underset{T\in\mathbb{R}^{d\times d}}{\text{minimize}}\, R(T,\mathbf{Z}). \tag{2}$$

We note that this problem is conceptually equivalent to the multi-task feature learning algorithm in [2]. Let us denote by $H(\mathbf{Z})$ the minimum in (2). Intuitively, this quantity is a measure of heterogeneity of the training sets. That is, the smaller $H(\mathbf{Z})$ the less heterogeneous (more related) the tasks, in that it is possible to find a common representation which fits the training sets $\mathbf{Z}$ well.

3 Heterogeneous Environment

The approach outlined above relies on the assumption that *all* the tasks are mutually related in the sense that they share the same representation. This requirement may be too strong when the environment is heterogeneous.

3.1 Method

Assume that there are several groups of tasks, so that the tasks within each group are related but tasks from different groups have little in common. If $I \subseteq \{1, \ldots, n\}$ is the set of indices of the tasks within such a group, we expect the training data $\mathbf{Z}(I) := (\mathbf{z}_t)_{t \in I}$ to be highly related, that is, the heterogeneity measure $H(\mathbf{Z}(I))$ to be small.

Our goal therefore is to partition the training set $\mathbf{Z}$ into K groups that minimize average heterogeneity. For this purpose, we let $\mathcal{P}$ be the set of all partitions of size K of the set $\{1, \ldots, n\}$ and solve the problem

$$\underset{\{I_1,\ldots,I_K\} \in \mathcal{P}}{\text{minimize}} \sum_{k=1}^{K} \frac{|I_k|}{n} H(\mathbf{Z}(I_k)). \tag{3}$$

The factors $\frac{|I_k|}{n}$ weight each group in proportion to its size.

This approach can be equivalently seen as that of learning a library of K feature maps $\mathbf{T} = (T_1, \ldots, T_K)$, where each T_k is a $d \times d$ matrix representing the k-th group of tasks in the environment. To see this, we rewrite problem (3) as

$$\underset{\{I_1,\ldots,I_K\} \in \mathcal{P}}{\text{minimize}} \; \underset{T_1,\ldots,T_K}{\text{minimize}} \sum_{k=1}^{K} \frac{|I_k|}{n} R(T_k, \mathbf{Z}(I_k)). \tag{4}$$

Interchanging the minimization over the partitions and the matrices T_k, we obtain

$$\underset{T_1,\ldots,T_K}{\text{minimize}} \left\{ \frac{1}{n} \sum_{t=1}^{n} \min_{k=1}^{K} r(T_k, \mathbf{z}_t) \right\}. \tag{5}$$

This observation reformulates the combinatorial optimization problem (3) as a continuous optimization problem and is analogous to the passage from the assignment problem to the objective function of k-means clustering.

When $K = 1$ problem (5) reduces to problem (2). For $K > 1$, the minimization over k effects an assignment of tasks to groups. The h-th group consists of those tasks t for which the regularization error $r(T_k, \mathbf{z}_t)$ is minimal for $k = h$. Therefore, the matrix T_k is a common representation for the tasks of the corresponding group.

We now describe how the library $\mathbf{T}$ is used to learn a new task from a given training set $\mathbf{z}$. First, we compute the weight vectors $v(T_k, \mathbf{z})$ for $k = 1, \ldots, K$ and the associated minimal values $r(T_k, \mathbf{z})$ in (1). Second, we assign the task to the group

$$h = \arg \min_{k=1}^{K} r(T_k, \mathbf{z}).$$

The corresponding weight vector, $v(T_h, \mathbf{z})$, is then used for the classification of future data from the same task, preprocessed by T_h.

The two norms appearing in the definition (1) of $r\,(T, \mathbf{z})$ have an effect of complexity regularization, which we briefly sketch. The term $\|T\|_2^2$ controls the complexity of candidate libraries. If a large number of tasks have been observed in the past, the quantity minimized in (5) is, with high probability in $\mathbf{Z}$ and uniformly in all libraries $\mathbf{T}$, a good upper estimate for the quantity

$$\mathbb{E}_{\mathbf{z}} \min_{k=1}^{K} r\,(T_k, \mathbf{z})\,,$$

where the expectation is over a training set generated from a random task drawn from the environment. Similarly, the term $\|v\|^2$ regularizes the complexity of candidate weight vectors and has the effect that $r\,(T, \mathbf{z})$ is, for sufficient sample size, a good upper estimate for the expected classification error incurred by the use of $v(T, \mathbf{z})$. Combining these observations, one finds that the quantity minimized by our method is close to a high-probability upper bound on the error incurred by the above algorithm using $\mathbf{T}$ on future tasks.

These arguments, which give a statistical justification for our method, are made rigorous in [4] for a closely related model.

3.2 Connection to Spectral Regularization

We now explain why the learned representations $T_1, \ldots, T_K$ in (5) are encouraged to be low dimensional. We first analyse the case $K = 1$. Let us use the notation $W = [w_1, \ldots, w_n]$ if $w_1, \ldots, w_n \in \mathbb{R}^d$.

Lemma 1. *Problem* (2) *is equivalent to*

$$\underset{W \in \mathbb{R}^{d \times n}}{\text{minimize}} \left\{ \frac{1}{mn} \sum_{t=1}^{n} \sum_{i=1}^{m} \ell(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \|W\|_1 \right\} \tag{6}$$

where $\gamma = 2\sqrt{\lambda}$ *and* $\|W\|_1$ *is the* ℓ_1 *norm of the singular values of* W*. Moreover if* $\hat{W}$ *solves* (6) *and* $\hat{T}$ *solves* (2)*, then*

$$\hat{T}^\top \hat{T} = (\lambda \hat{W} \hat{W}^\top)^{\frac{1}{2}}.$$

Proof. We define the matrix $D = T^\top T$. If T is full rank, we have, for every training set $\mathbf{z}$, that

$$r(T, \mathbf{z}) = \min_{w \in \mathbb{R}^d} \left\{ \hat{\ell}(w, \mathbf{z}) + \lambda \langle w, D^{-1} w \rangle + \mathrm{tr} D \right\},$$

where $\hat{\ell}(w, \mathbf{z}) := \frac{1}{m} \sum_{i=1}^{m} \ell(\langle w, x_i \rangle, y_i)$. Thus, problem (2) becomes

$$\inf_{D \succ 0} \min_{W \in \mathbb{R}^{d \times n}} \left\{ \frac{1}{n} \sum_{t=1}^{n} \hat{\ell}(w_t, \mathbf{z}_t) + \lambda \mathrm{tr}(D^{-1} W W^\top) + \mathrm{tr} D \right\}.$$

Interchanging the infimum with the minimum and following [2], the infimum over D is realized by $(\lambda W W^\top)^{\frac{1}{2}}$. The result then follows. □

We note that regularization with $\|W\|_1$, the trace norm, has originally been considered by [11] in the context of collaborative filtering. As shown in [9], the trace norm is the convex envelope of the rank function in the unit ball of matrices. This provides some intuition as to why the optimal matrix $\hat{W}$ (and, by Lemma 1, $\hat{T}$) has low rank.

We now consider the general case $K \geq 1$. If W is a $d \times n$ matrix and $I \subseteq \{1, \ldots, n\}$, we let $W(I) = [w_t : t \in I]$. The proof of the following result is established along similar lines as in Lemma 1.

Theorem 1. *Problem* (3) *is equivalent to the problem*

$$\underset{W \in \mathbb{R}^{d \times n}}{\text{minimize}} \left\{ \frac{1}{nm} \sum_{t=1}^{n} \sum_{i=1}^{m} \ell(\langle w_t, x_{ti} \rangle, y_{ti}) + \gamma \min_{\{I_1, \ldots, I_K\} \in \mathcal{P}} \sum_{k=1}^{K} \|W(I_k)\|_1 \right\}.$$

The above theorem states that problem (3) is equivalent to a regularization problem in which the tasks are partitioned into groups, so that the associated weight vectors have small trace norm on average. Regarding the trace norm as an approximation of the rank, we interpret this regularization as favoring groups of tasks which lie in low dimensional subspaces.

4 Learning Algorithm

We now describe an algorithm for solving problem (3). The algorithm performs stochastic gradient descent on the objective function (5). At each iteration, it selects a task index $t \in \{1, \ldots, n\}$ at random and computes the gradient[1] of the function

$$\min_{k=1}^{K} r(T_k, \mathbf{z}_t).$$

Only the matrix which realizes this minimum needs to be updated. This step requires the computation of the K vectors $v(T_k, \mathbf{z}_t)$ for the current values of the matrices $T_1, \ldots, T_K$, that is, the solution of K standard regularization problems. Thus, if the time complexity for solving each of these problems is $C(m, d)$ then the total time complexity per iteration is $O(KC(m, d) + md^2)$. This linear dependence on K is appealing in practice since it allows for more complex models. We also note that, since the optimal matrices in the library will be low dimensional, we may further accelerate the algorithm by using rectangular matrices T_k with a small but sufficient number of rows.

Although this algorithm is guaranteed to converge, the objective function (5) is non-convex and hence the limiting point is not necessarily a global solution. We note, in passing, that we are not aware of a single multi-task method on heterogeneous task environments that is convex. However, for the purposes of finding a good local minimum, the following initialization heuristic has been observed to lead to good empirical results. First we train a single feature map

[1] We ignore the issue of non-differentiability, because the objective function is almost everywhere differentiable.

Algorithm 1

Inputs: number of groups K, regularization parameter $\lambda > 0$, learning rate $\eta > 0$, training sets $\mathbf{z}_t = \{(x_{t1}, y_{t1}), \ldots, (x_{tm}, y_{tm})\},\ t = 1, \ldots, n$.
Initialization: Randomly choose $d \times d$ real matrices $T_1, \ldots, T_K$.
Repeat until convergence of the objective
  Draw t at random from $\{1, \ldots, n\}$
  for $k = 1, \ldots, K$ **do**
    Compute a solution $v(T_k, \mathbf{z}_t)$ of (1)
  end for
  Set $h = \arg\min_{k=1}^{K} r(T_k, \mathbf{z}_t)$, $\bar{v} = v(T_h, \mathbf{z}_t)$
  Set $T_h = T_h - \frac{\eta}{m} \sum_{i=1}^{m} \ell' \left(\langle \bar{v}, T_h x_{ti} \rangle, y_{ti} \right) \bar{v} x_{ti}^\top - 2\eta T_h$

($K = 1$) until convergence. Then from this map two slightly mutated matrices are created to initialize the library with $K = 2$ and the process is repeated up to the actual K we aim for. As we shall see in Section 5, in our experiments the algorithm has always converged to a good local minimum, in that the great majority of the tasks were assigned to correct groups.

5 Experiments

In classification experiments performed with synthetic and real data, we have verified the following hypotheses.

- The learning algorithm correctly distinguishes the heterogeneous groups of tasks and determines the appropriate subspaces within each group.
- The algorithm improves significantly over that in [2] (case $K = 1$) in terms of the transfer error on new tasks.[2] Moreover, when allowing more complex models (that is, when K is larger than the actual number of groups in the data) the transfer error does not improve.
- The performance improves monotonically with the number n of tasks available for training and deteriorates with the number G of underlying groups.

In all the experiments, we have used the SVM hinge loss and tuned the regularization parameter λ using cross validation. Choosing K was also done by cross validation, as we discuss below.

5.1 Synthetic Data

Environment. In the first experiment, we assume that the environment distribution ρ is a uniform mixture of a number G of group-specific measures ρ_k on the unit sphere S^{d-1} in $\mathbb{R}^d$. In other words, $\rho = \frac{1}{G} \sum_{k=1}^{G} \rho_k$. A task is chosen

[2] Average misclassification error on new tasks.

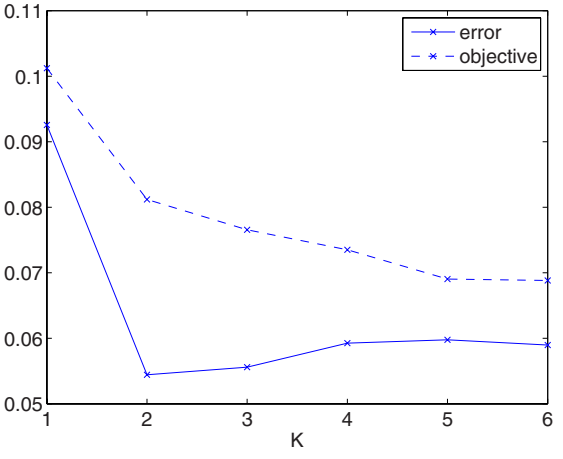


Fig. 1. Synthetic data (with $G = 2$). Plot of transfer error and objective function (5) versus K for $n = 1000$ tasks. The values of the objective have been rescaled to fit the plot. The error obtained using the identity matrix (training the tasks independently) was 0.35.

by selecting the k-th group with probability $\frac{1}{G}$ and then drawing a task weight vector w from ρ_k.

We further assume that each measure ρ_k is concentrated and uniform on the intersection of a low dimensional subspace of $\mathbb{R}^d$ with S^{d-1} and that these subspaces are mutually orthogonal. Each task is a binary classification represented by a vector $w \in S^{d-1}$. An input x for each task is sampled uniformly on S^{d-1} and the outputs are obtained by taking the sign of $\langle w, x \rangle$.

Our experiment consists in using the training set sampled from the environment to identify the subspace on which each group of tasks lies and subsequently to reliably predict membership in such a subspace for new tasks drawn from the distribution ρ.

Results. We first consider an environment with two groups of tasks (that is, $G = 2$), each of which lies on a 2-dimensional space of $\mathbb{R}^{100}$. We use $m = 50$ examples for training on each task and compute the transfer error over 200 new tasks sampled from the same environment, training on 50 examples and testing on 150 examples. In experiments with the synthetic data, the algorithm typically converged in less than 100,000 iterations.

As shown in Figure 1, using $K \geq 2$ yields a large improvement over grouping all tasks in the same group. Moreover, adding more than two groups in the model has a negative effect, the reason being that all matrices T_k are used in the obtained solution. This means that for $K > 2$ the objective function continues to decrease, as demontrated in the figure. Thus, K can be selected using cross validation on the training tasks but cannot be selected using the objective (5). Note that cross validation is meaningful when there is a sufficient number of related tasks, even if the sample per task is small.

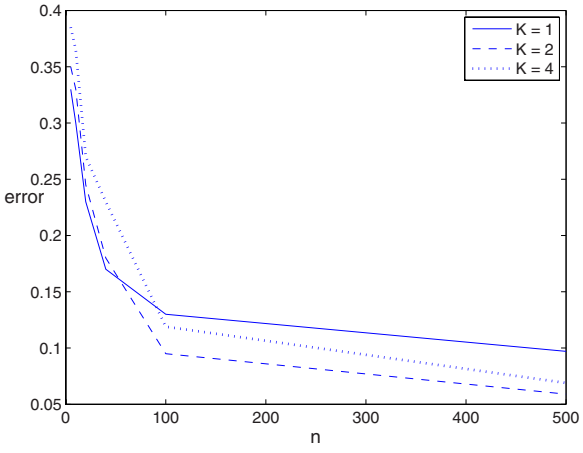


Fig. 2. Synthetic data (with $G = 2$). Plot of transfer error versus n using $K = 1, 2, 4$.

In the case $K = 2$, the resulting matrices T_k reflect the structure of the environment as each of them projects on a 2-dimensional subspace of $\mathbb{R}^{100}$. Specifically, we found that the *two* largest singular values of T_1 and T_2 account for 97% of the spectrum, whereas for $K = 1$ the *four* largest singular values are needed. In addition, we have verified that the tasks are assigned to the correct groups with 99% accuracy.

An interesting observation is that even for $K \neq 2$ the 4-dimensional support of the whole environment is correctly identified. Thus, for $K = 1$ the obtained matrix projects on a 4-dimensional subspace, whereas for $K = 4$ the matrices project on parts of this subspace.

Other issues relate to the effects of the dimensions of the problem on transfer error. In Figure 2, we verify the known result – see e.g. [6] – that the error decreases with n. Note that for small values of n, the method performs better with $K = 1$ than with $K = 4$ and for even smaller values, it outperforms $K = 2$. The reason is that with a small number of tasks the data may be insufficient for learning each of the actual two subspaces but sufficient for learning the joint subspace.

Another effect is that of G, the actual number of underlying subspaces. Our method performs well with values of G much larger than 2 but as G increases the problem inevitably becomes harder. In Figure 3, we plot the error obtained by training our method and tuning K for different values of G.

5.2 Character Recognition Experiments

We now describe two experiments with images of handwritten characters[3], which further illustrate how our method works. The character images were obtained

[3] Available at `http://www.andreas-maurer.eu/similarity.htm`

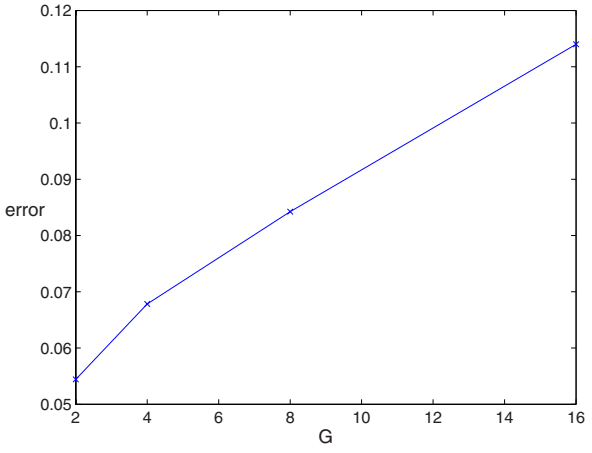


Fig. 3. Synthetic data. Plot of transfer error versus G using $n = 1000$.

with a real camera. We also note that rotation/scaling (see below) of the character shown in an image was done mechanically during image acquisition. In the experiments, we have treated images simply as vectors of pixels and no use of image preprocessing techniques or special properties of image data was made.

Projection on Image Parts. We consider the problem of invariant classification in the presence of noise. Every task is a pairwise classification of 28×56 images of characters. One half of the image contains the relevant character under an arbitrary rotation, whereas the other half contains a randomly chosen character also under an arbitrary rotation. There are two groups of tasks occurring with equal probability: tasks in which the relevant character appears on the left half of the image and tasks in which it appears on the right half. For example, the images of Figure 4 are examples of a task (classifying 6 versus 1) of the "right" group.

To train a library we selected pairs of alphabetic characters. Since we wish to obtain a library of features that represent the broader structural properties of the environment, namely rotation invariance on one half of the image with simultaneous irrelevance on the other half, we tested the learned library on samples generated from the digit character set (after removing digit 9). In this way we directly measured how well the representation transferred to novel but structurally similar problems. We trained our algorithm until convergence, which was achieved in 400,000 iterations.

To measure the transfer error, we generated 500 tasks of pairwise classification from the digits data set. For each of these tasks, a sample of size 10 was chosen to train a classifier using the library found during the training phase. The performance of this classifier was then tested on 40 examples chosen from the same task. The high dimensionality of the input space (1568) and the small sample size (10) explain the high error (0.27) of learning each task independently, as Table 1 shows. At the same time, the large number of tasks observed for the

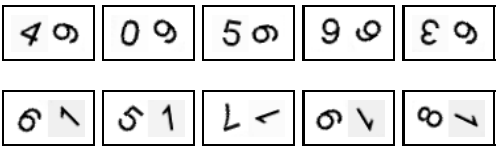


Fig. 4. Character recognition (left-right data set). Example images from one classification task (6 versus 1) in the group that focuses on the right part of the image.

Table 1. Character recognition (left-right data set). Transfer error for different methods.

Independent	$K = 1$	$K = 2$
0.27	0.036	0.013

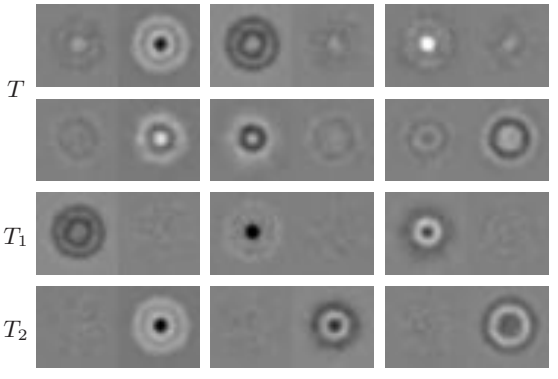


Fig. 5. Character recognition (left-right data set). Dominant right singular vectors of: T learned with $K = 1$ (top); T_1 and T_2 learned with $K = 2$ (bottom). See text for description.

training of the feature maps (1000) accounts for the spectacular improvement obtained by transfer learning ($K = 1$).

A convenient way to visualize the effect of the resulting maps on the image vector is to display their right singular vectors as 28×56 images. In Figure 5, we present these after normalization to the range $[-1, 1]$ and mapping of zero to gray. We show only the singular vectors corresponding to the 6 largest singular values for T and the 3 largest singular values for T_1 and T_2. The concentric features in both halves clearly reflect the rotational invariance properties present in the training data. In the single map case, the map does not distinguish the relevance of each part of the image relative to the tasks, whereas in the two-map case, matrix T_1 represents the invariance properties of the "left" group of tasks and matrix T_2 those of the "right" group. In addition, the singular values of T, T_1 and T_2 (Figure 6) show that this specialization of T_1 and T_2 results in more concise representations within each group of tasks. That is, the effective ranks of

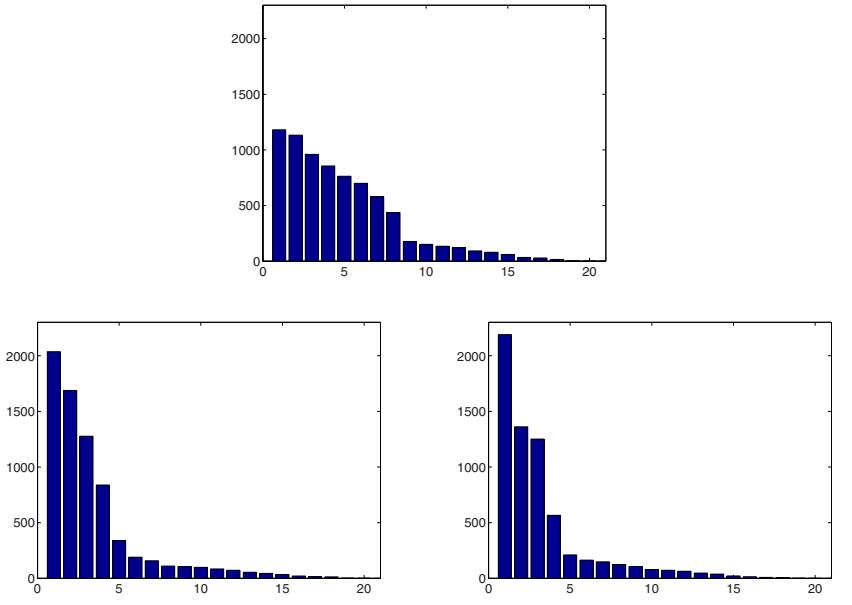


Fig. 6. Character recognition (left-right data set). Spectrum of T learned with $K = 1$ (top), spectra of T_1 (bottom-left) and T_2 (bottom-right), learned with $K = 2$.

Table 2. Character recognition (left-right data set). Assignment of tasks in groups, when training with $K = 2$. The first three rows show percentage of assignments for the transfer tasks, the last row for the training tasks.

	T_1	T_2
All digits (left & right)	48.2%	51.8%
Left	99.2%	0.8%
Right	1.4%	98.6%
Training data	50.7%	49.3%

T_1 and T_2 equal 4, whereas that of T equals 8 and hence, setting $K = 2$ allows us to learn subspaces with lower dimensionalities.

Finally, in Table 2 we verify that the assignment of tasks into groups reflects the left-right structure of the data set. Here, the "Left" ("Right") percentage was measured over the sample that corresponds to classification on the left (right) half of the image.

Rotation and Scale Invariance. Our final experiment is set in an environment which contains a mixture of tasks involving rotation invariant and scale invariant character recognition, with scale factors ranging from 2/3 to 3/2 (see Figure 7). As in the previous experiment, our libraries were trained using pairwise classification of alphabetic characters and were transferred on pairwise classifi-

Table 3. Character recognition (rotation-scaling data set). Transfer error for different methods.

Independent	K=1	K=2
0.17	0.018	0.015

cation of digits. Because of the difficulties caused by scale invariance, the images were preprocessed using a Gaussian kernel, with 1000 centers chosen randomly from the training data and Gaussian kernel width of $1/\sqrt{8}$. Again, 400,000 iterations were needed for training the algorithm.

Unlike the previous experiments, it is now unclear what the features relevant to each group of tasks may be, or how the feature space relevant to rotation invariant recognition relates to that of scale invariant recognition. To investigate the potential of our method we ask two questions:

1. Is the grouping method (with $K = 2$) able to improve the transfer performance of the more standard transfer learning algorithm with $K = 1$?
2. To what extent does the grouping reflect our intuition of rotation and scale invariant problems as corresponding to two distinct groups of tasks?

For training our method, we generated 2000 samples of size 10 (5 per class) from the alphabetic set, selecting rotation and scale invariant tasks with equal probability. The transfer data was generated as in the previous experiment: 500 tasks drawn from the digits, 10 training examples per task, 40 examples for testing.

As shown in Table 3, we observe again a dramatic improvement from independent learning to transfer learning. There is also an observable advantage of the grouping method but not as pronounced as in the previous experiments. The most likely explanation is that the two feature subspaces may now be far from orthogonal, unlike the previous experiments.

To answer the second of the above questions we counted the assignments to T_1, T_2 within exclusively rotation or scale invariant tasks (Table 4). We see a certain specialization of T_2 to rotation invariant tasks and of T_1 to scale invariant tasks, but it is not as clearly defined as in the previous experiment. Here it

Table 4. Character recognition (rotation-scaling data set). Assignment of tasks in groups, when training with $K = 2$. The first three rows show percentage of assignments for the transfer tasks, the last row for the training tasks.

	T_1	T_2
All digits (rotation & scale)	44.8%	55.2%
Rotation invariance	10%	90%
Scale invariance	72%	28%
Training data	46.5%	53.5%

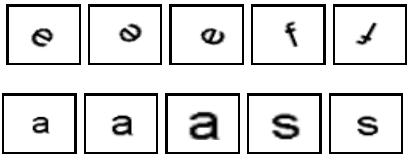


Fig. 7. Character recognition (rotation-scaling data set). Example images from a classification task in the rotationally invariant group (top) and one in the scale invariant group (bottom).

is important to realize that grouping into rotation and scale invariant tasks agrees with a certain human intuition, but there may well be other task-grouping criteria (for example rounded characters versus characters with sharp corners) which may be necessary for further improving performance.

6 Summary

We have presented a method for transfer learning over environments in which tasks are concentrated on a number of low dimensional subspaces (*heterogeneous environments*). Our approach, which is justified theoretically by a generalization bound on the transfer error [4], uses gradient descent to learn a library of feature maps that describe such subspaces. The method naturally extends previous work on multi-task learning which considered only one common group of tasks [2].

We have reported experiments with synthetic and real data. These experiments are illustrative examples of the many real problems in which multiple heterogeneous tasks occur. They clearly demonstrate that our method is effective in identifying both *the groups of tasks* and *the underlying common feature maps*. Moreover, the method outperforms both single-task learning and the precursor method [2], which corresponds to the case that $K = 1$. We believe the work is a significant improvement over [2] (which in turn has shown state-of-the-art results in a number of benchmark data sets), since any algorithm using $K = 1$ cannot distinguish the different subspaces on which the tasks may lie.

We have also briefly sketched an interpretation of our approach in terms of spectral regularization. We speculate that following this observation our method can be easily applied in the context of collaborative filtering, see e.g. [11]. Another interesting question that can be the topic of future research is to study conditions on the environment which ensure convergence of the algorithm to a good local minimum.

References

1. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research 6, 1817–1853 (2005)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning (2008), http://www.springerlink.com/content/161105027v344n03

3. Argyriou, A., Micchelli, C.A., Pontil, M., Ying, Y.: A spectral regularization framework for multi-task structure learning. Advances in Neural Information Processing Systems (2007)
4. Argyriou, A., Maurer, A., Pontil, M.: Generalization Bounds for Task Grouping (Technical Report), University College London (February 2008)
5. Bakker, B., Heskes, T.: Task clustering and gating for Bayesian multi–task learning. Journal of Machine Learning Research 4, 83–99 (2003)
6. Baxter, J.: A model for inductive bias learning. Journal of Artificial Intelligence Research 12, 149–198 (2000)
7. Caruana, R.: Multi–task learning. Machine Learning 28, 41–75 (1997)
8. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research 6, 615–637 (2005)
9. Fazel, M., Hindi, H., Boyd, S.P.: A rank minimization heuristic with application to minimum order system approximation. In: Proceedings, American Control Conference, pp. 4734–4739 (2001)
10. Maurer, A.: Bounds for linear multi-task learning. Journal of Machine Learning Research 7, 117–139 (2006)
11. Srebro, N., Rennie, J.D.M., Jaakkola, T.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems, vol. 17, pp. 1329–1336. MIT Press, Cambridge (2005)
12. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: The TC algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 489–497 (1996)
13. Wilson, A., Fern, A., Ray, S., Tadepalli, P.: Multi-task reinforcement learning: a hierarchical Bayesian approach. In: Proceedings of the Twenty-Fourth International Conference on Machine Learning, pp. 1015–1022 (2007)
14. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. Journal of Machine Learning Research 8, 35–63 (2007)

Minimum-Size Bases of Association Rules

José L. Balcázar

Departament de Llenguatges i Sistemes Informàtics
Laboratori d'Algorísmica Relacional, Complexitat i Aprenentatge
Universitat Politècnica de Catalunya, Barcelona
balqui@lsi.upc.edu

Abstract. We focus on confidence-bounded association rules; we model a rather practical situation in which the confidence threshold is fixed by the user, as usually happens in applications. Within this model, we study notions of redundancy among association rules from a fundamental perspective: we discuss several existing alternative definitions and provide new characterizations and relationships between them. We show that these alternatives correspond actually to just two variants, which differ in the special treatment of full-confidence implications. For each of these two notions of redundancy, we show how to construct complete bases of absolutely minimum size.

Keywords: association rules, redundancy, optimum bases.

1 Motivation and Related Work

Few, if any, data mining tasks have the relative importance within that field of research as association rule mining. Whereas practitioners provide some success stories in various fields, researchers have provided a wealth of algorithmic ideas related to the task. Since the publication of the first proposal of confidence- and support-bound-based association mining [2], many algorithms have been designed. The interesting FIMI competition tested a wide family of these algorithms (http://fimi.cs.helsinki.fi). Currently, the amount of knowledge related to association rules has grown to the extent that the tasks of creating complete surveys and websites that maintain pointers to literature on association rules become daunting (a recent survey is [8] but additional materials appear in http://wwwai.wu-wien.ac.at/~hahsler/research/association_rules/, for instance, at the time of writing); see also [3], [26], [32], [33], and the references and discussions in their introductory sections.

A close relative of the notion of association rule, namely, that of exact implication, that is, an association rule that holds in 100% of the cases, had been studied before in the research area of closure spaces, where a number of methods have been found to construct, for every binary dataset, sets of implications (often called "bases") that are complete in the sense that all other implications can be derived from them; some of these bases enjoy minimality properties depending on the notion of derivation at hand [10], [12], [27], [31], in fact, such implications can

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 86–101, 2008.

be seen also as conjunctions of definite Horn clauses, and the closure under intersection that characterizes closures spaces corresponds to the fact, well-known in logic and knowledge representation, that Horn theories are exactly those closed under bitwise intersection of propositional models (see e.g. [17]). Thus, as a form of knowledge gathered from a dataset, implications have several advantages: explicit or implicit correspondence with Horn logic, therefore a tight parallel with functional dependencies and a simple and well-known calculus through the Armstrong axioms (explained below), whence a clear notion of redundancy.

However, the fact has been long acknowledged (e.g. already in [23]) that, often, it is inappropriate to search only for absolute implications in the analysis of real world datasets. There may be many reasons to consider interesting a co-occurrence pattern, even if the perceived implication does not hold in all the cases. Already in [23], partial rules are defined in relation to their so-called-there "precision", that is, the notion of intensity of implication now widely called "confidence": for a given rule $X \to Y$, the ratio of how often X and Y are seen together to how often X is seen. Many other alternative measures of intensity of implication exist, and several sources describe them (see [13], [14]); we keep our focus on confidence, which is among the most common ones, certainly the first one proposed, and has a natural interpretation for educated users because it corresponds to a lower bound to the observed conditional probability.

The first attempts at mining partial rules were also proposed in [23]; yet, the process of searching for implications or for partial rules was not used on really large datasets until the introduction of the support bound: a threshold on how often the itemsets under analysis appear in the dataset. The idea of restricting the exploration for association rules to frequent itemsets, with respect to a support threshold, gave rise to the most widely discussed and applied algorithm, Apriori [3], and to an intense research activity. Unfortunately, if the combinatorial properties of implications are already nontrivial to handle, those of partial rules are even harder. Already with full-confidence implications, the output of an association mining process often consists of large sets of rules, and a well-known difficulty in applied association rule mining lies in that, on large datasets, and for sensible settings of the confidence and support thresholds, huge amounts of association rules are often obtained, much beyond what any user of the data mining process may be expected to look at; and the difficulty of studying the formal properties of partial rules makes it very difficult to select in a principled, provably optimal way, a subset of the rules without losing information.

Therefore, besides the interesting progress in the topic of how to organize and query the rules discovered (see [21], [22], [28]), one research topic that has been worthy of attention is the identification of patterns that indicate redundancy of rules, and ways to avoid that redundancy [1], [9], [18], [19], [23], [26], [32] (see also section 6 of [8] and the references therein). A major problem, open since [23], would be to give a general method for constructing bases of minimum size: a basis for a given dataset would be a subset of the rules that hold in the dataset, that is complete, in the sense that it makes all the remaining rules redundant. Therefore, restricting ourselves to the basis does not incur loss of information.

But the very notion of completeness of a basis depends on the concrete ways specified to construct "redundant" rules out of the basis. Therefore, we discuss this point briefly now and propose one specific standpoint. Imagine that a standard association rule miner has been run on a given dataset, with user-specified thresholds of support and confidence — a situation that fully matches most cases of application; its output $\mathcal{R}$ is now available, in the form of a (probably large) set of rules, each labeled with its confidence, all of these above the threshold. We want to select a basis $\mathcal{B} \subseteq \mathcal{R}$, aiming at choosing it as small as possible and, simultaneously, making sure that we do not lose information in doing so. (In fact, we are after better algorithmics that obtain directly $\mathcal{B}$ instead of mining for the whole of $\mathcal{R}$ and postprocessing it, but, for the sake of the properties of the basis, the discussion is clearer if we assume $\mathcal{R}$ known.) Thus, we are to find a subset of rules $\mathcal{B} \subseteq \mathcal{R}$ such that all the rules in $\mathcal{R}$ become redundant; and, of course, the crux now is how to define formally "redundant".

For the case of exact implications, "redundancy" has several equivalent natural, robust logical formalizations, such as entailment among definite Horn clauses. Alternatively, it also corresponds to derivability under the so-called Armstrong axiom schemes [30]: Reflexivity ($X \to X$), Augmentation (if $X \to Z$ and $Y \to W$ then $XY \to ZW$) and Transitivity (if $X \to Y$ and $Y \to Z$ then $X \to Z$).

But, in our context of partial rules with a hard confidence threshold in place, Augmentation and Transitivity, and also other natural inference schemes, are not valid anymore: for instance, if most of the times X appears it comes with Z, but it only comes with Y when Z is not present, then the confidence of $X \to Z$ may be high whereas the confidence of $XY \to Z$ may be null; that is, Augmentation (with $W = \emptyset$ here) is not valid. Neither is Transitivity: knowing that $A \to B$ and $B \to C$ (or even $AB \to C$) hold with confidence γ does not inform us about whether $A \to C$ holds with confidence γ. Additionally, a rule with several items in the consequent is *not* equivalent to the conjunction of the Horn-style rules with the same antecedent and each item of the consequent separately, and, if we look only into rules with singletons as consequents, we are almost certain to lose information. Indeed, if the confidence of $X \to YZ$ is high, it means that Y and Z appear together in most of the transactions having X; whereas the fact that both Y and Z appear in fractions at least γ of the transactions having X does not inform us that they show up *together* at a similar ratio of these transactions. This is also a failing form of Augmentation, with $X = Y$ this time.

Thus, we lack characterizations of derivability, and are left with the task of identifying, little by little, specific cases of redundancy, working them out, and seeing whether they give us bases and with which properties. This task indeed has been performed, and with great results already, but there is some progress to achieve yet. Most notably for our work here, we find that [1], [19] [26], and [32] have all proposed interesting notions of redundancy and methods to construct nonredundant bases; some of them work (as does [23]) in a setting where all the partial rules, plus their confidences, are to be inferred from the basis; and still the size of their basis can be suboptimal. The basis of [1] and the representative rules of [19] are of

minimum size in a well-defined sense, but this fact is a contribution of the present paper. The "basic association rules" of [20] suffer a limitation that consequents must be singletons, which loses information as indicated above (the same limitation applies to the useful `apriori` implementation of Borgelt available on the web [6]). Along a different avenue, some works are set up in a context of "using all the information available": namely, combining the supports of some sets in various ways, one can determine, through short computations, the supports of many other sets and the confidence of many rules. The "nonderivable" itemsets and rules [7], [15], [25] and the "covering" scheme of [9] all refer to the possibility of deriving rules of confidence above the threshold from information about the supports of specific sets. This seems a very effective approach, employing information about actual supports, that depend on the dataset at hand. These works were inspiring, and crucial to our research; however, we wish to study here a non-comparable approach: as discussed above, in our setting we are after a notion of redundancy based only on little information, essentially as in [1] and [19]. Instead of considering a rule redundant when its confidence can be somehow inferred from others, we take the slightly different view that it is redundant when "the fact that its confidence is above γ can be inferred from others", where these other rules are known to have confidence above γ but the inference process does not use their actual confidence values. We believe that this approach will be a good complement to the existing works, and expect that it would be particularly useful in cases where the user of the data mining system is not familiar with inclusion-exclusion principles and similar facts used in the "nonderivable itemsets" approach: only the rules (or almost only the rules, as explained next) are brought to bear in the derivation of other rules. We will see that this is possible, and indeed can be achieved through definitions that are already in the literature [1], [18], of which we establish new, important properties; and we will push this approach provably to the limit.

Then, we follow the proposals of [26], [32], and several other works, pushing through beyond that limit by assuming that we are allowed one single additional bit per rule: we will handle separately, to a given extent, full-confidence implications from lower-than-1-confidence rules, in order to profit from their very different combinatorics. We discuss adequate notions of redundancy and completeness, prove new properties, and refine the existing basis constructions up to a point where we can prove again that we attain the limit of the redundancy notion. We close the paper with some empirical data regarding our proposals (the author is grateful to Bart Goethals and Christian Borgelt for making their respective implementations of `fpgrowth` and `apriori` so easily accessible) and a short section of conclusions. Due to the space limits, here we defer the proofs of the theorems and additional examples and discussions to an extended version available from the author.

2 Preliminaries

A dataset $\mathcal{D}$ is given; it consists of transactions, each of which is an itemset labeled by a unique transaction identifier. The identifiers allow for many

transactions sharing the same itemset. Upper-case, often subscripted letters from the end of the alphabet, like X_1 or Y_0, denote itemsets. Juxtaposition denotes union of itemsets, as in XY; and $Z \subset X$ denotes proper subsets. For a transaction t, we denote $t \models X$ the fact that X is a subset of the itemset corresponding to t.

From the given dataset we obtain a notion of support of an itemset: $s_{\mathcal{D}}(X)$ is the cardinality of the set of transactions that include it, $\{t \in \mathcal{D} \mid t \models X\}$, sometimes, abusing language, we also refer to that set of transactions itself as support. Whenever $\mathcal{D}$ is clear, we drop the subindex: $s(X)$.

We immediately obtain by standard means (see, for instance, [12] or [32]) a notion of closed itemsets, namely, those that cannot be enlarged while maintaining the same support. The function that maps each itemset to the smallest closed set that contains it is known to be monotonic, extensive, and idempotent, that is, it is a closure operator. This notion will be reviewed in more detail later on.

Association rules are pairs of itemsets, denoted as $X \to Y$ for itemsets X and Y. Intuitively, they express that Y occurs particularly often among the transactions in which X occurs. More precisely, each such rule has a confidence associated: the confidence $c_{\mathcal{D}}(X \to Y)$ of an association rule $X \to Y$ in a dataset $\mathcal{D}$ is $\frac{s(XY)}{s(X)}$, that is, the ratio by which transactions having X have also Y, or, again, the observed empirical approximation to a conditional probability of Y given X. As with support, often we drop the subindex $\mathcal{D}$. This view suggests a form of correlation that, in many applications, is interpreted implicitly as a form of causality (which, however, is not guaranteed in any formal way; see the interesting discussion in [11]).

We denote as $\mathcal{R}$ the set of all the rules of confidence at least γ for the given dataset $\mathcal{D}$, again we should label $\mathcal{R}$ with $\mathcal{D}$ and γ as subscripts but these will be always clear from the context. Always $\gamma > 0$. We resort to the convention that, if $s(X) = 0$ (which implies $s(XY) = 0$) we redefine the undefined confidence as 1, since the intuitive expression "all transactions having X do have also Y" becomes vacuously true. Additionally, note that $c_{\mathcal{D}}(X \to Y) = c_{\mathcal{D}}(X \to XY)$, and we will switch rather freely between right hand sides that include the left hand side and right hand sides that don't: when two rules have the same left hand side, and the union of left and right hand sides also coincide, we say that they are *equivalent by reflexivity*. Clearly their confidences will always coincide.

3 Redundancy Notions

We start our analysis from one of the notions of redundancy proposed in [1], and from a variation thereof, seemingly less restrictive.

Definition 1. *1. $X_0 \to Y_0$ is AY-redundant with respect to $X_1 \to Y_1$ if the confidence and support of the former are always larger than or equal to those of the latter, whatever the dataset [1].*

2. $X_0 \to Y_0$ is plainly redundant with respect to $X_1 \to Y_1$ if the confidence of $X_0 \to Y_0$ is larger than or equal to the confidence of the latter, whatever the dataset.

Thus, plain redundancy is like AY-redundancy, but forgetting the condition regarding support. It turns out that the condition about confidence is already rather strong, due to the "whatever the dataset" clause, to the point that our first new result is that the simplified version is as powerful as the original one:

Theorem 1. *Consider any two rules $X_0 \to Y_0$ and $X_1 \to Y_1$ where $Y_0 \not\subseteq X_0$. Then $X_0 \to Y_0$ is AY-redundant with respect to $X_1 \to Y_1$ if and only if $X_0 \to Y_0$ is plainly redundant with respect to $X_1 \to Y_1$.*

This will allow us to concentrate on confidence bounds at the time of discussing complete bases, since support bounds will follow essentially from that result. The reference indicated [1] also provides two simpler definitions of redundancy:

Definition 2. *(From [1].)*

1. *If $Z_0 \neq \emptyset$, rule $X_0Z_0 \to Y_0$ is simply redundant with respect to $X_0 \to Y_0Z_0$;*
2. *if $X_1 \subseteq X_0$ and $X_0Y_0 \subset X_1Y_1$, rule $X_0 \to Y_0$ is strictly redundant with respect to $X_1 \to Y_1$.*

It is rather easy to check that moving attributes from the right hand side into the left hand side can only increase the confidence and leave equal the support: this fact corresponds to simple redundancy, and relates rules obtained from the same frequent set $X_0Y_0Z_0$. Strict redundancy focuses, instead, on rules extracted from two different (frequent) itemsets, say X_0Y_0 where X_0 will be considered as antecedent, versus X_1Y_1, where X_1 will be antecedent, and under the conditions that $X_1 \subseteq X_0$ and $X_0Y_0 \subset X_1Y_1$ (the case $X_0Y_0 = X_1Y_1$ is already covered by simple redundancy). Both simple and strict redundancies imply AY-redundancy; this is argued in [1], which discusses most of the results just in terms of these two simplified notions. Note that, in principle, there could possibly be many other ways of being redundant beyond simple and strict redundancies: we show next, however, that, in essence, this is not the case, as we can state the following new, far from obvious characterization:

Theorem 2. *Consider any two rules $X_0 \to Y_0$ and $X_1 \to Y_1$ where $Y_0 \not\subseteq X_0$. The following are equivalent:*

1. *$X_1 \subseteq X_0$ and $X_0Y_0 \subseteq X_1Y_1$;*
2. *rule $X_0 \to Y_0$ is either simply redundant or strictly redundant with respect to $X_1 \to Y_1$, or they are equivalent by reflexivity;*
3. *rule $X_0 \to Y_0$ is plainly redundant with respect to $X_1 \to Y_1$.*

The two inclusions in the first statement form a condition that characterizes exactly the cover operator of [18] (Property 3.3 there): hence that operator is also fully equivalent to plain redundancy.

3.1 Optimum-Size Basis for Plain Redundancy

The main property of a basis, namely completeness, or not losing information upon deletion of the remaining rules, corresponds now to the following formalization:

Definition 3. *Given the set of rules $\mathcal{R}$ that hold in a given dataset $\mathcal{D}$ at confidence at least γ, $\mathcal{B} \subseteq \mathcal{R}$ is a complete basis if every rule of $\mathcal{R}$ is plainly redundant with respect to some rule of $\mathcal{B}$.*

We describe now, briefly, the construction of a basis as proposed, almost simultaneously, in [1] and in [19] (called there "representative rules").

Definition 4. *Given itemsets Y and $X \subseteq Y$, X is a* γ-antecedent *for Y if $c(X \rightarrow Y) \geq \gamma$, that is, $s(Y) \geq \gamma s(X)$.*

This is the same as organizing all the rules of $\mathcal{R}$ according to the itemset resulting from union of antecedent and consequent. For each itemset Z, we will keep some rules $X \rightarrow Y$ with $XY = Z$, or equivalently, we will keep some antecedents X for Z. Keeping all γ-antecedents of all sets yields, essentially, the whole of $\mathcal{R}$. We will keep only a part of them, as few as possible, but losing no information.

Definition 5. *Given itemsets Y and $X \subseteq Y$, X is a* valid γ-antecedent *for Y if the following holds:*

1. *X is a γ-antecedent of Y,*
2. *X fulfills this preceding property* minimally*: no proper subset of X is a γ-antecedent of Y, and*
3. *X is not a minimal γ-antecedent of an itemset strictly larger than Y.*

Now, the basis we are studying consists of all rules $X \rightarrow Z - X$ for all itemsets Z and for all valid antecedents X of Z. We refer to this basis as $\mathcal{B}_0$. It is immediate to see that all the rules in the basis $\mathcal{B}_0$ actually hold with confidence at least γ since, for $X \subseteq Z$, $c(X \rightarrow Z - X) = c(X \rightarrow Z) \geq \gamma$ which is explicitly required for X to be an antecedent of Z. It is proved in [1] that this basis is irredundant with respect to simple and strict redundancies. Completeness can be stated as follows (see also [19]):

Proposition 1. *$\mathcal{B}_0$ is a complete basis: all the rules in $\mathcal{R}$ are plainly redundant with respect to $\mathcal{B}_0$.*

Hence $\mathcal{B}_0$ contains rules that hold and which imply all the rules that hold, that is, it is indeed a basis. Efficient algorithms to construct $\mathcal{B}_0$ are provided in [1] and [19], where small examples can be found as well; we have developed alternative algorithmics that we will describe elsewhere. Our contribution here, rather than algorithmic, is foundational: now we can state and prove the most interesting novel property of this basis. It was known [1] that it is irredundant with respect to simple or strict redundancy, that is, none of the rules in it is simply redundant, nor strictly redundant. Our characterization in Theorem 2 then applies, so that none of the rules in $\mathcal{B}_0$ is, in fact, plainly redundant. But this irredundancy does not rule out the possibility that some other basis, constructed in an altogether different form, could have less many rules. We state and prove now that this is not so: there is absolutely no other way of constructing a basis smaller than this one, while preserving completeness with respect to plain redundancy, because it

has absolutely minimum size among all complete bases. Therefore, in order to find smaller ways of listing association rules from $\mathcal{R}$, and not losing information, a notion of redundancy more powerful than plain redundancy is unavoidably necessary.

Theorem 3. *Let $\mathcal{B}' \subseteq \mathcal{R}$ be any set of rules such that all the rules of $\mathcal{R}$ are plainly redundant with respect to $\mathcal{B}'$. Then, $\mathcal{B}'$ has at least as many rules as $\mathcal{B}_0$.*

Implementation and test of this approach reveals interesting reductions of the sizes of the rules, but these are still somewhat large. We concentrate efforts henceforth on the study of alternative existing notions of redundancy, as an attempt at getting bases smaller than $\mathcal{B}_0$.

4 Closure-Based Redundancy

The theorems in the previous section tell us that, for plain redundancy, the absolute limit of a basis, thus without losing information, is reached by the construction of [1]. Several studies, prominently [32], have put forward a different notion of redundancy; namely, they give a separate role to the full-confidence implications. Along this way, one gets a stronger notion of redundancy and, therefore, smaller bases can be constructed.

Indeed, implications can be summarized better, because they allow for transitivity and augmentation to apply in order to find redundancies; moreover, they can be combined in a certain form of transitivity with a partial rule of confidence at least γ to give rules that also have confidence at least γ. The best way to handle them is through a closure operator ([10], [12], [32], [26], [31]).

Given a dataset $\mathcal{D}$, the closure operator associated to $\mathcal{D}$ maps each itemset X to the largest itemset $\overline{X}$ that has the same support as X in $\mathcal{D}$; it can be defined in several alternative ways. A set is closed if it coincides with its closure. When $\overline{X} = Y$ we also say that X is a generator of Y. Our definition gives directly that always $s(X) = s(\overline{X})$. We will make liberal use of this fact, which is easy to check also with other definitions of the closure operator, as stated in [26], [32], and others.

Implications, or association rules of confidence 1, are closely related to this closure operator: $c(X \to Y) = 1$ if and only if $Y \subseteq \overline{X}$. Several quite good algorithms exist to find the closed sets and their supports. There are proposals of basis constructions from closed sets in the literature. In the min-max basis of [26], antecedents are minimal generators, that is, as small as possible, whereas consequents are closures, that is, as large as possible. We will follow the approach of [32] where both antecedents and consequents are chosen as small as possible; but, if we consider only number of rules, and not their sizes, both approaches share many similarities.

Redundancy based on closures is a natural generalization of equivalence by reflexivity; it works as follows ([32], see also section 4 in [26]): given a dataset and a closure operator corresponding to implications that have confidence 1 in the dataset, two partial rules $X_0 \to Y_0$ and $X_1 \to Y_1$ such that $\overline{X_0} = \overline{X_1}$ and

$\overline{X_0Y_0} = \overline{X_1Y_1}$ turn out to be equivalent in terms of support and confidence; the reason is that $s(X_0) = s(\overline{X_0}) = s(\overline{X_1}) = s(X_1)$, and $s(X_0Y_0) = s(\overline{X_0Y_0}) = s(\overline{X_1Y_1}) = s(X_1Y_1)$. Therefore, groups of rules sharing the same closure of the antecedent, and the same closure of the union of antecedent and consequent, give cases of redundancy. The notion of redundancy in [32] leads to selecting as irredundant rules from each such group that have inclusion-minimal antecedents and consequents. The size of the basis obtained in this way is analyzed empirically in [32], where it is also combined with the strategy from [23] of using only neighbor closures. This basis was found to be smaller than the set of all the rules in all cases, many times exhibiting a huge reduction factor. We will provide additional improvements by refining the closure analysis and by combining the idea of closure-based redundancy with the notion of valid antecedents of the previous section. Most interestingly, we provide again a proof that, with our variant, we reach the limit of closure-based redundancy: our basis will be shown again to have the smallest possible size with respect to closure-based completeness.

4.1 Characterizing Closure-Based Redundancy

Let $\mathcal{B}$ be the set of implications, of confidence 1, in the dataset $\mathcal{D}$; alternatively, $\mathcal{B}$ can be any of the bases already known for implications in a dataset. In our experiments later on we will use as $\mathcal{B}$ the Guigues-Duquenne basis, that has been proved to be of minimum size [10], [31]. From here on, we require $0 < \gamma < 1$, leaving the rules of confidence 1 to be handled from $\mathcal{B}$.

Definition 6. *Let $\mathcal{B}$ be a set of implications. Rule $X_2 \to Y_2$ has closure-based redundancy relative to $\mathcal{B}$ with respect to rule $X_1 \to Y_1$, which we denote by $\mathcal{B} \cup \{X_1 \to Y_1\} \models X_2 \to Y_2$, if any dataset $\mathcal{D}$ in which all the rules in $\mathcal{B}$ hold with confidence 1 gives $c_{\mathcal{D}}(X_2 \to Y_2) \geq c_{\mathcal{D}}(X_1 \to Y_1)$.*

We continue our study by showing a necessary and sufficient condition for closure-based redundancy, along the same lines as the one in the previous section.

Theorem 4. *Let $\mathcal{B}$ be a set of exact rules, with associated closure operator mapping each itemset Z to its closure $\overline{Z}$. Let $X_2 \to Y_2$ be a rule not implied by $\mathcal{B}$, that is, where $Y_2 \not\subseteq \overline{X_2}$. Then, the following are equivalent:*

1. *$X_1 \subseteq \overline{X_2}$ and $X_2Y_2 \subseteq \overline{X_1Y_1}$*
2. *$\mathcal{B} \cup \{X_1 \to Y_1\} \models X_2 \to Y_2$*

4.2 Optimum-Size Basis for Closure-Based Redundancy

In a similar way as in the previous section, we give here a basis, similar to the one proposed in [32] but smaller, by combining closure-based redundancy with the conditions of Definition 5. As in [32], the rules of confidence 1 are handled separately: we focus on the partial rules. We show first that our construction indeed gives a basis, that is, consists of rules that hold and make redundant all other rules that hold, in the following sense:

Definition 7. *Given the set of rules $\mathcal{R}$ that hold in a given dataset $\mathcal{D}$ at confidence at least γ, and given in it the rules $\mathcal{B}$ that hold with confidence 1,* closure-based completeness *of a set of partial rules $\mathcal{B}' \subseteq \mathcal{R}$ holds if every partial rule of $\mathcal{R}$ has closure-based redundancy relative to $\mathcal{B}$ with respect to some rule of $\mathcal{B}'$.*

Conceptually, our new basis departs only slightly from the bases of [26] and [32], but is nonetheless different in most cases (and therefore smaller, in some cases by an important factor, as shown below). It is constructed as follows. For each closed set X, we will consider a number of closed sets Y properly included in X to act as antecedents. They follow a similar pattern to the one of valid antecedents; but, instead of minimal antecedents, we will pick just minimal closed antecedents. That is:

Definition 8. *For each closed set X, a closed set $Y \subset X$ (proper inclusion) is a* basic γ-antecedent *if the following holds:*

1. *Y is a γ-antecedent of X: $s(X) \geq \gamma s(Y)$;*
2. *Y is* not *a γ-antecedent of X' for any larger closed set $X' \supset X$: $s(X') < \gamma s(Y)$;*
3. *Y is minimal among the closed proper subsets of X for which the previous two conditions hold.*

Then we use these antecedents for our basis:

Definition 9. *The basis $B^\star_\gamma$ consists of all the rules $Y \to X - Y$ for all closed sets X and all basic γ-antecedents Y of X.*

This set of rules entails exactly the rules that hold:

Theorem 5. *Let $\mathcal{B}$ be any basis for implications that hold with confidence 1.*

1. *All the rules in $B^\star_\gamma$ hold with confidence at least γ.*
2. *$B^\star_\gamma$ is a complete basis for closure-based redundancy: if the rule $Y \to Z$ holds with confidence at least γ, then, taken together with the full-confidence implications, $\mathcal{B} \cup B^\star_\gamma \models Y \to Z$.*

Now we can move to the main result of this section, and in fact of the whole paper: this basis is of absolutely minimum size.

Theorem 6. *Let $\mathcal{B}' \subseteq \mathcal{R}$ be an arbitrary basis having closure-based completeness for $\mathcal{R}$. Then, for each implication $Y \to Z \in B^\star_\gamma$, there is in $\mathcal{B}'$ an implication of the form $Y' \to Z'$ with $\overline{Y'Z'} = \overline{YZ}$ and $\overline{Y'} = Y$.*

That is, for each $Y \to X - Y \in B^\star_\gamma$, there is a corresponding rule in $Y' \to Z' \in \mathcal{B}'$; this rule in $\mathcal{B}'$ provides us with $X = \overline{Y'Z'}$ and $Y = \overline{Y'}$. Thus, both X and Y are univocally determined by $Y' \to Z'$ and, hence, the same rule in $\mathcal{B}'$ cannot yield but one of the rules in $B^\star_\gamma$, so that $\mathcal{B}'$ must have at least as many rules as $B^\star_\gamma$. Therefore, $B^\star_\gamma$ has a minimum number of rules, in an absolute sense, among all bases that are complete according to closure-based redundancy.

4.3 Support Bounds

Now we discuss, briefly, what happens if we work under a support threshold. In fact, for most datasets, if we do not impose a support bound then even the lattice including just closed sets reaches easily dozens or hundreds of thousands of nodes, or indeed even millions.

Assume that we do not mine the rule basis from a lattice including all closed sets but only those above a support threshold. Is there any risk of obtaining a wrong basis? The answer is negative:

Proposition 2. *For any fixed confidence threshold γ, mining the $B^\star_\gamma$ basis only on closed sets of support at least τ, for $\tau \leq \gamma$, provides a basis of the whole set of rules that hold with confidence at least γ and support at least τ.*

This proposition is proved by combining Theorem 1 with an easy observation: if the rule $X \to Y$ has support at least τ, both $\overline{X}$ and $\overline{XY}$ have also support at least τ, so that we can argue as in the proof of completeness above. We are therefore safe if we apply the basis construction for $B^\star_\gamma$ to a lattice of frequent closed sets above support τ, instead of the whole lattice of closed sets. However, this proposition does not ensure us that the basis obtained under a support bound is minimal anymore. There is a strategy (to be described in a forthcoming paper) that provides us with a correct and provably minimum basis also under a support bound, at the price of somewhat longer computations. For our development here, we just consider basis $B^\star_\gamma$ as computed from the lattice of closed sets above the support bound.

5 Empirical Validation

As indicated, this paper focuses rather on the foundational properties of redundancy and bases, and its algorithmic content is not the main contribution. However, we present some algorithmics and empirical results for the sake of completeness. We have implemented a known construction of a minimal basis for the full-confidence implications [10] to compute the closure operator, and an algorithm that constructs our proposed basis. For this computation, we consider an algorithm that conveniently uses as a black-box a separate closed itemsets miner. It is explained in Table 1: it scans the lattice of closed sets repeatedly to construct the basic γ-antecedents. That implementation has provided us with all the figures in Table 3. The initialization of the lists scan the whole lattice to pick up closed proper predecessors: a natural alternative would preprocess the lattice as a graph in order to find the predecessors of a node directly; however, in practice, with this alternative, whenever the graph requires too much space the computation slows down unacceptably, probably due to a worse fit to virtual memory caching. The search of the optimal algorithmic compromise between avoiding repeated computations while efficiently handling virtual memory will be the topic of further work; the current implementation gives answers in just seconds in most cases, on a mid-range Windows XP laptop, taking a few minutes when the closure space reaches a couple dozen thousand itemsets.

Table 1. Algorithmic approach to $B^{\star}_{\gamma}$

```
Algorithm B*_γ-1(closed sets,γ):
    for each of the closed sets:
        construct a list of closed proper subsets
        filter it to leave only γ-antecedents
        filter again to leave only minimal γ-antecedents
    for each of the closed sets:
        filter out from the list minimal γ-antecedents of larger closed sets
    for each of the closed sets:
        for each antecedent in its list:
            output as rule:
                left hand side: a minimum-size generator of the antecedent
                right hand side: a minimum-size generator of the closed set,
                    removing from it items appearing in the left hand side
```

Table 2. Parameters of the datasets

Dataset	Num. Items	Size in Transactions
Chess	76	3196
Connect	130	67557
Mushroom	120	8124
Pumsb	7117	49046
Pumsb_star	7117	49046
T10I4D100K	1000	100000

An important property of our approach, shared with all the closure-based works, is that the key parameter is neither the size nor the dimensionality of the dataset, rather the size of the closures lattice. If that structure is of moderate size, our proposal works very well; the average degree of the corresponding Hasse graph is the next crucial value. If this degree is sublinear, which tends to be the case, then the computation of the rules is quadratic on the number of closures.

We have run our implementation exactly on the same real datasets (downloaded from `http://fimi.cs.helsinki.fi`) as the main table in [32], and with the same values of the parameters. Of course further comparisons are desirable, but in this way it is clear that we did not pick our experiments specifically to favor our approach. We have also included one of the synthetic datasets treated there. Some parameters of the datasets are indicated in Table 2.

Numbers of rules appear in Table 3. The values of "Supp/Conf" is the value of the confidence and support parameters; "Traditional" is the number of rules obtained via the original definition; "Closure-based" is the number of rules of the closure-based basis of [32] (which is already an impressive improvement); these columns are taken literally from [32] (there the value of the confidence parameter is made to coincide with the support parameter, so we do the same). These numbers are to be compared with the number of rules for our approach, that is, the sum of the columns "GDbasis" (number of rules in the Guigues-Duquenne minimum-size basis for full-confidence implications) and $B^{\star}_{\gamma}$. That

Table 3. Sizes of sets of rules for some datasets

Dataset	Supp/Conf	Traditional	Closure-based	GD basis	$B^\star_\gamma$ basis	Total
Chess	80	552564	27711	5	226	231
Chess	70	8171198	152074	10	891	901
Connect	97	8092	1116	4	41	45
Connect	90	3640704	18848	14	222	236
Mushroom	40	7020	475	24	41	65
Mushroom	20	19191656	5741	170	158	328
Pumsb	95	1170	267	2	32	34
Pumsb	85	1408950	44483	9	1080	1089
Pumsb_star	60	2358	192	5	6	11
Pumsb_star	40	5659536	13479	47	82	129
T10I4D100K	0.5	2216	1231	0	585	585
T10I4D100K	0.1	431838	86902	214	4054	4268

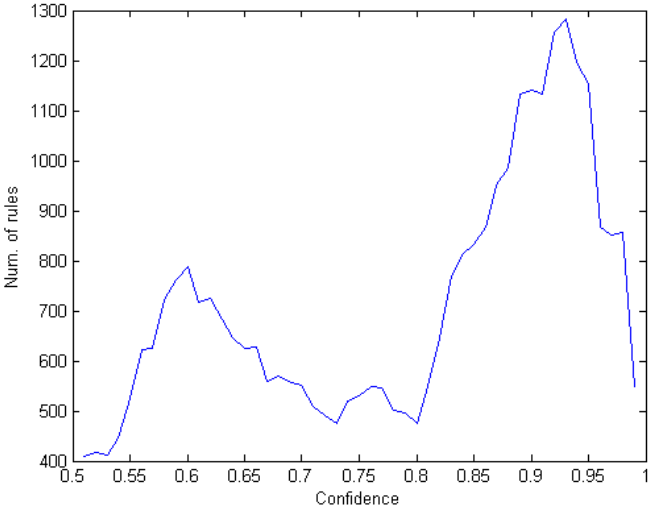


Fig. 1. Number of rules in the basis $B^\star_\gamma$ for `pumsb-star` at 20% support

total is provided in the last column. The improvements are apparent: however, they are just consequences of our main theorems.

One very interesting outcome of the experiments was the following. Some of us are used to a monotonicity intuition, by which lower confidence thresholds allow for more rules, so that the size of the output grows (sometimes enormously) as the confidence drops. However, in the case of the basis $B^\star_\gamma$, some datasets exhibit a nonmonotonic evolution: at lesser confidence thresholds, sometimes less many rules are mined. Inspecting the actual rules, we can find the reason: sometimes there are several rules at, say, 90% confidence that become simultaneously redundant due to a single rule of smaller confidence, say 85%, which does not appear at 90% confidence. This may reduce the set of rules upon lowering the

confidence threshold. An example illustrating this point is given by the dataset `pumsb-star`, mined for our basis $B^{\star}_{\gamma}$ at 20% support threshold with confidence ranging from 99% to 51%. The number of full-confidence implications in the Guigues-Duquenne basis at this support threshold is 47. The number of partial rules varies between 476 (at 80% confidence) and 1282 (at 93%), except near 50% confidence where the number of rules drops a bit more. The graphic in Figure 1 indicates the number of rules for each confidence level between 0.51 and 0.99, computed at a granularity of 0.01.

6 Conclusions

Our main contribution, at a glance, is a way of thinking about confidence-bounded association rules in terms of notions of redundancy. We have provided characterizations of existing redundancy notions, from which we have been able to exactly pinpoint the limitation of an existing basis proposal, for a plain notion of redundancy, and also to improve the constructions of bases for closure-based redundancy. As a consequence, analysis of specific datasets is now more feasible, from the perspective of the human analyst who must read through the output of the rule miner.

Several questions are worth further study. Mainly, it is not difficult to propose stronger notions of redundancy, and, in fact, we could see these contributions as progress towards a complete logical approach, where redundancy would play the role of entailment: in a forthcoming paper [4], will describe logical calculi that exactly correspond to plain redundancy and to closure-based redundancy. Then, the following natural question arises: our notions of redundancy only relate one partial rule to another partial rule, possibly in presence of full-confidence implications, and always with respect to a fixed confidence threshold: is it indeed possible that a partial rule is entailed by two partial rules, but not by a single one? The failures of Transitivity and Augmentation strongly suggest the intuition of a negative answer; in a forthcoming paper [4] we will see that this is so for certain confidence thresholds, but that there are confidence thresholds where this intuition is wrong: for instance, the reader may enjoy analyzing the case of rules $A \to BC$ and $A \to BD$, assuming that they hold with 65% confidence, and trying to make rule $ACD \to B$ fall below the same confidence threshold in the same dataset. This is, in fact, impossible, and our next paper [4] will characterize exactly when two partial rules entail a third one, either in presence or in absence of a closure operator for the exact rules. This could be a way towards stronger redundancy notions and even smaller bases.

We are studying as well ways of computing bases of provably minimum size under a support threshold; as discussed above, our basis is correct, and small, but, for this particular case, full optimality is not guaranteed yet. We will discuss an alternative in future works. Additional comparatives with other approaches to redundancy, based on additional information beyond the list of rules mined, is also necessary, including, for instance, the nonderivable itemsets of [7] or the cover operator of [9].

We plan also to extend this approach to the mining of more complex dependencies [29] or of dependencies among structured objects; however, extending the development to sequences, partial orders, and trees, is not fully trivial, because, as demonstrated in [5], the combinatorial structures may make redundant rules that would not be redundant in a propositional (item-based) framework; additionally, an intriguing question is: what part of all this discussion remains true if implication intensity measures different from confidence are used?

References

1. Aggarwal, C.C., Yu, P.S.: A new approach to online generation of association rules. IEEE Transactions on Knowledge and Data Engineering 13, 527–540 (2001)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in very large databases. In: ACM SIGMOD, pp. 207–216 (1993)
3. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Fayyad, U., et al. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI Press, Menlo Park
4. Balcázar, J.L.: Deduction Schemes for Association Rules. Discovery Science (to appear, 2008)
5. Balcázar, J.L., Bifet, A., Lozano, A.: Mining implications from lattices of closed trees. Extraction et Gestion des Connaissances (2008)
6. Borgelt, C.: Efficient Implementations of Apriori and Eclat Workshop on Frequent Itemset Mining Implementations (2003)
7. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 74–85. Springer, Heidelberg (2002)
8. Ceglar, A., Roddick, J.F.: Association mining. ACM Computing Surveys 38 (2006)
9. Cristofor, L., Simovici, D.: Generating an informative cover for association rules. In: ICDM 2002, pp. 597–613 (2002)
10. Guigues, J.-L., Duquenne, V.: Famille minimale d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines 24, 5–18 (1986)
11. Freitas, A.: Understanding the crucial differences between classification and discovery of association rules. SIGKDD Explorations 2, 65–69 (2000)
12. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Heidelberg (1999)
13. Garriga, G.C.: Statistical Strategies for Pruning All the Uninteresting Association Rules. In: ECAI 2004, pp. 430–434 (2004)
14. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: a survey. ACM Computing Surveys 38 (2006)
15. Goethals, B., Muhonen, J., Toivonen, H.: Mining non-derivable association rules. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674. Springer, Heidelberg (2005)
16. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharma, R.S.: Discovering all most specific sentences. ACM Trans. Database Syst. 28, 140–174 (2003)
17. Khardon, R., Roth, D.: Reasoning with models. Artificial Intelligence 87, 187–213 (1996)
18. Kryszkiewicz, M.: Representative Association Rules. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 198–209. Springer, Heidelberg (1998)

19. Kryszkiewicz, M.: Representative Association Rules and Minimum Condition Maximum Consequence Association Rules. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 361–369. Springer, Heidelberg (1998)
20. Li, G., Hamilton, H.: In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178. Springer, Heidelberg (2004)
21. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: KDD 1999, pp. 125–134 (1999)
22. Liu, B., Hu, M., Hsu, W.: Multi-level organization and summarization of the discovered rules. In: KDD 2000 (2000)
23. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques et Sciences Humaines 29, 35–55 (1991)
24. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1, 241–258 (1997)
25. Muhonen, J., Toivonen, H.: Closed non-derivable itemsets. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 601–608. Springer, Heidelberg (2006)
26. Pasquier, N., Taouil, R., Bastide, Y., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. Journal of Intelligent Information Systems 24, 29–60 (2005)
27. Pfaltz, J.L., Taylor, C.M.: Closed set mining of biological data. In: BIOKDD 2002, pp. 43–48 (2002)
28. Tuzhilin, A., Liu, B.: Querying multiple sets of discovered rules. In: KDD 2002, pp. 52–60 (2002)
29. Simovici, D.A., Cristofor, D., Cristofor, L.: Mining purity dependencies in databases. In: Extraction et Gestion des Connaissances EGC 2002, pp. 257–268 (2002)
30. Ullman, J., Widom, J.: A First Course in Database Systems (1997)
31. Wild, M.: A theory of finite closure spaces based on implications. Advances in Mathematics 108, 118–139 (1994)
32. Zaki, M.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9, 223–248 (2004)
33. Zaki, M., Ogihara, M.: Theoretical foundations of association rules. In: DMKD Workshop on research issues in DMKD (1998)

Combining Classifiers through Triplet-Based Belief Functions

Yaxin Bi[1], Shengli Wu[1], Xuhui Shen[2], and Pan Xiong[2]

[1] School of Computing and Mathematics, University of Ulster
Newtownabbey, Co. Antrim, BT37 0QB, UK
{y.bi, s.wu1}@ulster.ac.uk
[2] Institute of Earthquake Science, China Earthquake Administration
Beijing, 100036 China

Abstract. Classifier outputs in the form of continuous values have often been combined using linear sum or stacking, but little is generally known about evidential reasoning methods for combining truncated lists of ordered decisions. In this paper we introduce a novel class-indifferent method for combining such a kind of classifier decisions. Specifically we model each output given by classifiers on new instances as a list of ranked decisions that is divided into 2 subsets of decisions, which are represented by *triplet-based belief functions* and then are combined using Dempster's rule of combination. We present a formalism for triplet-based belief functions and establish a range of general formulae for combining these beliefs in order to arrive at a consensus decision. In addition we carry out a comparative analysis with an alternative representation *dichotomous belief functions* on the UCI benchmark data. We also compare our combination method with the popular methods of stacking, boosting, linear sum and majority voting over the same benchmark data to demonstrate the advantage of our approach.

1 Introduction

The idea of ensemble learning or a committee approach is to learn and retain multiple classifiers and combine their decisions in some way to classify new instances [8]. Thus the key to the success of ensemble learning relies not only on a learning algorithm, but also on a combination function. In this work we focus on the latter task − developing an effective combination method. The design of a method for combining classifier decisions is a challenging task in constructing an effective ensemble and various methods have been developed in the past decades. Kuncheva in [12] roughly characterizes combination methods, based on the forms of classifier outputs, into two categories. In the first category, the combination of decisions is performed on single classes, such as majority voting and Bayesian probability, which have extensively been examined in the ensemble literature [9], [11] and [23].

The second category is concerned with the utilization of numeric scores (probabilities) corresponding to classes. One typical method, often called a

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 102–116, 2008.

class-aligned method, is based on the same classes from different classifiers in calculating the support for classes, regardless of what the support for the other classes is. This method includes meta-learning − stacking where combination functions are learnt from numeric values of classes [10] and [17], linear sum and order statistics [11], [21] and [24] . An alternative group of methods called class-indifferent methods makes use of as much information as possible obtained from sets of classes in calculating the support for each class [12].

A class-aligned method and a class-indifferent method are both based on continuous values of classes in calculating the support for classes, but the latter takes impact from different classes into account in determining the support for classes that permits the presence of uncertainty information − as happens when an instance is classified into different classes by different classifiers. Several work related to class-indifferent methods utilizes single classes and sets of classes as described in [1] and [23]. However class-indifferent methods for combining decisions in the form of lists of ranked decisions have not been intensively studied and are poorly understood. In particular, little is known about evidential reasoning methods for combining truncated lists of ordered decisions.

In this study we propose a novel approach to modeling the process of combining classifiers in the Dempster-Shafer theory framework, which is built on our previous study for text categorization [4] and [5]. We model each output given by classifiers on new instances as a list of ranked decisions (classes) that is divided into 2 subsets of decisions along with the whole set of decisions which are represented by the structure called a triplet. In each triplet, the first subset contains a decision corresponding to the largest numeric score, the second subset corresponds to the second largest numeric value and the whole set of decisions represents the uncertainty in making the former two decisions in terms of *ignorance*. In particular we extend our previous work by establishing the general formulae for combining triplets by Dempster's rule of combination and empirically and analytically examine the effect of different sizes of decision list on their accuracy for combining classifiers over the UCI benchmark data sets [7].

To evaluate the superiority of our method we have conducted a comparative analysis on the three representations of classifier decisions in the forms of triplet-based belief functions, dichotomous-based belief functions [3] and full list of probabilities, we also compare them with two state of the art methods: Stacking (multi-response linear regression) [17] and AdaBoost.M1 [16] along with majority voting and linear sum methods in combining individual (base) classifiers. By comparing the *t*-test results drawn from the various combinations of classifiers and the ranking statistics *win/draw/loss*, we demonstrate the properties and relative advantage of our method in combining classifiers.

2 Modelling Classifier Outputs and Combination Methods

In supervised machine learning, a learning algorithm is provided with training instances of the form $\{\langle d_1, c_1\rangle, \cdots, \langle d_{|D|}, c_q\rangle\}$ $(d_i \in D, c_i \in C, 1 \leq q \leq |C|)$ for

inducing some unknown function f such that $f(d) = c$. D is the space of attribute vectors and each vector d_i is in the form $(w_{i_1}, \cdots, w_{i_n})$ whose components are symbolic or numeric values; C is a set of categorical classes and each class c_i is in the form of class label. Given a set of training data, a learning algorithm is aimed at learning a function φ − a classifier from the training data, where classifier φ is an approximation to the unknown function f.

Given a new instance d, a classification task is to make the decision for d using φ about whether instance d belongs to class c_i. Instead of single-class assignment, we denote such a process as a mapping:

$$\varphi : D \rightarrow C \times [0,\ 1] \tag{1}$$

where $C \times [0,\ 1] = \{(c_i,\ s_i) \mid c_i \in C,\ 0 \leq s_i \leq 1,\ 1 \leq i \leq |C|\}$, s_i is a numeric values that can be in different forms, such as a similarity score, a class-conditional probability (prior posterior probability) or other measures, depending on the types of learning algorithms. It represents the degree of support or confidence about the proposition of that instance d is assigned to class c_i. The greater the value of class c_i, the greater the possibility of the instance belonging to that class. Without loss of generality, we denote the classifier output by $\varphi(d) = \{s_1, \cdots, s_{|C|}\}$ − a general representation of classifier outputs. Given an ensemble of classifiers, $\varphi_1, \varphi_2, \cdots, \varphi_M$, all classifier outputs can be organized into a matrix called a decision profile as depicted in Fig. 1, the combination of classifier outputs can be carried out in different ways. One of the most commonly used combination

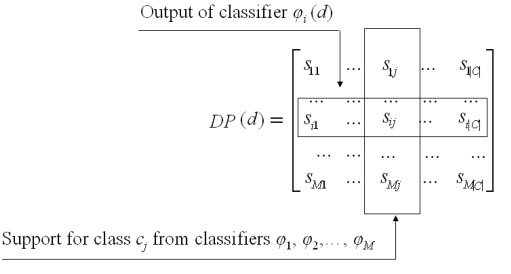


Fig. 1. A decision profile for instance d generated by $\varphi_1(d), \varphi_2(d), \cdots, \varphi_M(d)$

methods is to calculate the support for class c_j using only the DP's jth column, i.e. $s_{1j}, s_{2j}, \cdots, s_{Mj}$, regardless of what the support for the other classes is. We call such a methods a class-aligned method. Alternatively, the combination of classifier outputs can be performed on an entire decision profile or the selected information to constrain a class decision. We refer to this alternative group of methods as class-indifferent methods [12].

In our work, the concept of the class-indifferent methods is slightly different from the ones aforementioned in [1], [12] and [23] . We neither generate decision templates nor use an entire decision profile to compute the degrees of support for every class. Instead we select 2 classes from each $\varphi(d)$ according to their

numeric values and restructure it into a new list composed of three subsets of C which are represented by the novel evidence structure of triplet. In this way, a decision profile as illustrated in Fig. 1 will be restructured into a triplet decision profile where each column no longer corresponds to the same class. The degree of support for each class is computed through combining all triplets in a decision profile. We will detail our method in later sections.

3 Dempster-Shafer (DS) Theory of Evidence

In any exercise where decisions are to be combined, quantitative and qualitative pertinent information and knowledge often originate from different evidence sources, and they are often pervaded with uncertainty. In this study we seek a way to formalize such process in the context of ensembles for decision making with uncertainty. We briefly describe the DS theory of evidence below [19].

Definition 1. Let Θ be a finite nonempty set, and call it the *frame of discernment*. Let $[0,1]$ denote the interval of real numbers from zero to one, i.e. $[0,1] = \{x|0 \leq x \leq 1\}$. A function $m : 2^{\Theta} \rightarrow [0,1]$ is called a *belief mass function* if it satisfies:

$$1) \;\; m(\emptyset) = 0; \qquad 2) \sum_{X \subseteq \Theta} m(X) = 1.$$

A belief mass function is a basic probability assignment (*bpa*) to all subsets X of Θ. A subset A of a frame Θ is called a *focal element* of a belief mass function m over Θ if $m(A) > 0$ and it is called a singleton if it contains only one element.

Definition 2. Let m_1 and m_2 be belief mass functions on the same frame Θ, and for any subsets $A \subseteq \Theta$, the *orthogonal sum* $\oplus$ of two belief mass functions on A is defined as

$$m(A) = (1/N) \sum_{X \cap Y = A} m_1(X) m_2(Y) \tag{2}$$

where $N = 1 - \sum_{X \cap Y = \emptyset} m_1(X) m_2(Y)$ and $K = 1/N$ is called the normalization constant of the orthogonal sum $m_1 \oplus m_2$. The orthogonal sum is a fundamental operation of evidential reasoning and it is often called Dempster's rule of combination. There are two conditions to ensure the orthogonal sum exists: 1) $N \neq 0$; 2) two belief mass functions must be independent of each other. We often allocate some mass to undeterministic status by means of *ignorance*.

It should be noted that the second condition above is a theoretical restriction on applying the orthogonal sum. In the case of classifier combination, it could be argued that the pieces of evidence derived from classifiers may not be entirely independent, but in this study we simply make an independence assumption on this. A recent discussion on this issue can be found in [2].

4 Triplet-Based Belief Mass Function

In this section we describe the development of a novel evidence structure − the triplet and its formulation.

Starting by analyzing the computational complexity of combining multiple pieces of evidence, we consider how a more efficient method for combining evidence can be established. Given M pieces of evidence represented by Fig. 1, the computational complexity of combining these pieces of evidence using equation (2) is dominated by the number of elements in C and the number of classifiers M. In the worst case, the time complexity of combining M pieces of evidence is $O(|C|^{M-1})$. One way of reducing the computational complexity is to reduce the number of pieces of evidence being combined, so that the combination of evidence is carried by a partition of the frame of discernment C, with less focal elements than C, but including possible answers to the proposition of interest. The partition can thus be used in place of C when the computations of the orthogonal sum are carried out [18]. For example, a dichotomous structure can be used to partition the frame of discernment C into two subsets ϑ_1 and ϑ_2, where there are a number of mass functions that represent evidence in favor of ϑ_1 and against ϑ_2, along with the lack of evidence − *ignorance*. It has been shown that Dempster's rule can be implemented in such a way that the number of computations increases only linearly with the number of elements in C if the mass functions being combined are focused on the subsets where ϑ_1 is singleton and ϑ_2 is the complement of ϑ_1, i.e., $O(|C|)$ [3].

The partitioning technique enables a large problem to be broken up into several smaller and more tractable problems. However, a fundamental issue in applying this technique is how to select elements that contain the possibly correct answers to the propositions corresponding to C.

An intuitive way is to select the element with the highest degree of confidence. Indeed, since the classifier outputs approximate class posteriori probabilities, selecting the maximum probability reduces to selecting the output that is the most 'certain' of the decisions. This could be justified from two perspectives. First, the probability assignments given in formula (1) give quantitative representation of judgments made by classifiers on the propositions; the greater their values, the more likely these decisions are correct. Thus selecting the maximum distinguishes the trivial decisions from the important ones. Second, the combination of decisions with the lower degrees of confidence may not contribute to the performance increase of combined classifiers, but only make the combination of classifiers decisions more complicated [21]. The drawback of selecting the maximum, however, is that the combined performance can be reduced by a single dominant classifier that repeatedly provides high confidence values. Contenders with the higher values are always chosen as the final classification decisions, but some of these may not be correct.

To cope with the deficiency resulting from the maximal selection, we propose to take the second maximum decision into account in combining classifiers. Its inclusion not only provides valuable information contained in the discarded classes by the maximal selection for combining classifiers, but this also to some

extent avoids the deterioration of the combined performance caused by the errors resulting from a single dominant classifier that repeatedly produces high confidence values. We propose a novel structure − a triplet − partitioning a list of decisions $\varphi(d)$ into three subsets below.

Definition 3. Let C be a frame of discernment and let $\varphi(d) = \{s_1, \cdots, s_{|C|}\}$ be a list of scores, an application-specific belief mass function is defined as a mapping function, $m : 2^C \rightarrow [0, 1]$, i.e. a bpa to $c_i \in C$ for $1 \leq i \leq |C|$ as follows:

$$m(\{c_i\}) = \frac{s_i}{\sum_{j=1}^{|C|} s_j} \tag{3}$$

where $1 \leq i \leq |C|$.
This mass function expresses the degrees of belief with regard to the choices of classes to which a given instance could belong. By equation (3), we can rewrite $\varphi(d)$ as $\varphi(d) = m(\{c_1\}), m(\{c_2\}), \cdots, m(\{c_{|C|}\})$.

Definition 4. Let Θ be a frame of discernment and $\varphi(d) = \{m(\{x_1\}), m(\{x_2\}), ..., m(\{x_n\})\}$, where $|n| \geq 2$, an expression of the form $Y = \langle\{u\}, \{v\}, C\rangle$ is defined as a *triplet*, where $\{u\}, \{v\}$ are singletons, and C is the whole set Θ, and they satisfy

$$m(\{u\}) + m(\{v\}) + m(C) = 1$$

To obtain triplet belief mass functions, we define a focusing operator and denote it by m^σ as follows:

$$\{u\} = arg\max\{m(\{x_1\}), m(\{x_2\}), ..., m(\{x_n\})\} \tag{4}$$

$$\{v\} = arg\max\{m(\{x\}) \mid x \in \{x_1, ..., x_n\} - \{u\})\} \tag{5}$$

$$C = \Theta, m^\sigma(\Theta) = 1 - m^\sigma(\{u\}) + m^\sigma(\{v\}) \tag{6}$$

and each row in Fig. 1 is simply rewritten as formula (7) below.

$$\varphi_i(d) = \{m^\sigma(\{u\}), m^\sigma(\{v\}), m^\sigma(C)\}\ 1 \leq i \leq M \tag{7}$$

We refer to m^σ as a *triplet function* or as a *two-point mass function*. For the sake of simplicity, $\varphi_i(d) = \{m(\{u\}), m(\{v\}), m(C)\}$. Following the same way, we divide $\varphi_i(d)$ into four subsets which is represented by *three-point focuses* called a quartet and define associated quartet belief mass functions. More details about quartets can be found in [6].

Suppose we are given two triplets $\langle\{x_1\}, \{y_1\}, C\rangle$ and $\langle\{x_2\}, \{y_2\}, C\rangle$ where $x_i, y_i \in C$ $(i = 1, 2)$, and the associated triplet mass functions m_1 and m_2. The enumerative relations between any two pairs of focal elements $\{x_1\}, \{y_1\}$ and $\{x_2\}, \{y_2\}$ include: two focal points equal; one focal point equal; and totally different focal points. The general formulae for combining any number of triplet functions based on the three different cases are given below.

4.1 Two Focal Point Equal

Considering a collection of triplet belief mass functions $m_1, ..., m_l$ that are defined on $\{x, y, \Theta\}, ..., \{x, y, \Theta\}$, by formula (2) − the orthogonal sum − we can derive the general formulae to combine these triplet functions:

$$m(\{x\}) = K[\prod_{i=1}^{l}(m_i(\{x\}) + m_i(\Theta)) - \prod_{i=1}^{l} m_i(\Theta)] \tag{8}$$

$$m(\{y\}) = K[\prod_{i=1}^{l}(m_i(\{y\}) + m_i(\Theta)) - \prod_{i=1}^{l} m_i(\Theta)] \tag{9}$$

$$m(\Theta) = K \prod_{i=1}^{l} m_i(\Theta) \tag{10}$$

$$K^{-1} = \prod_{i=1}^{l}(m_i(\{x\}) + m_i(\Theta)) + \prod_{i=1}^{l}(m_i(\{y\}) + m_i(\Theta)) - \prod_{i=1}^{l} m_i(\Theta) \tag{11}$$

4.2 One Focal Point Equal

Similarly we consider the case of any number of triplet belief mass functions $m_1, ..., m_l$ defined on $\{x, y_1, \Theta\}, ..., \{x, y_l, \Theta\}$, where $y_1 \neq ... \neq y_l$, by repeatedly using formula (2) we then have

$$m(\{x\}) = K(\prod_{i=1}^{l}(m_i(\{x_i\}) + m_i(\Theta)) - \prod_{i=1}^{l} m_i(\Theta)) \tag{12}$$

$$m(\{y_i\}) = K(m_i(\{y_i\}) \prod_{j=1, j\neq i}^{l} m_j(\Theta));\ i \in \{1, \cdots, l\} \tag{13}$$

$$m(\Theta) = K \prod_{i=1}^{l} m_i(\Theta) \tag{14}$$

$$K^{-1} = \prod_{i=1}^{l}(m_i(\{x_i\}) + m_i(\Theta)) + m_i(\{y_i\}) \prod_{j=1, j\neq i}^{l} m_j(\Theta);\ i \in \{1, \cdots, l\} \tag{15}$$

4.3 Completely Different Focal Points

Given a collection of triplet mass functions be $m_1, ..., m_l$ defined on $\{x_1, y_1, \Theta\}$, $..., \{x_l, y_l, \Theta\}$, where $x_i \neq y_j, 1 \leq i, j \leq l$, by repeatedly using formula (2) to perform pairwise combination of any triplet functions, we then have

$$m(\{x_i\}) = K(\prod_{i \neq j}^{l}(m_i(\{x_i\})m_j(\Theta)) \tag{16}$$

$$m(\{y_i\}) = K(\prod_{i \neq j}^{l}(m_i(\{y_i\})m_j(\Theta)) \tag{17}$$

$$m(\Theta) = K(\prod_{i=1}^{l}(m_i(\Theta)) \tag{18}$$

$$K^{-1} = \sum_{i=1}^{l}\prod_{i \neq j}^{l}(m_i(\{x_i\})m_j(\Theta) + \sum_{i=1}^{l}\prod_{i \neq j}^{l}(m_i(\{y_i\})m_j(\Theta) + \prod_{i=1}^{l}(m_i(\Theta) \tag{19}$$

With a collection of triplet functions $m_1, m_2, ..., m_n$, simply it can be reorganized on the basis of *one* focus being equal, *two* focuses being equal, and *none* of focuses being equal as follows:

$$m = \underbrace{m_{11} \oplus ... \oplus m_{1l_1}}_{l_1 \text{ items}} ... \oplus \underbrace{m_{k1} \oplus ... \oplus m_{kl_k}}_{l_k \text{ items}} \tag{20}$$

where $1 \leq k \leq 3; 0 \leq l1, ..., l_k$ and $l_1 + ... + l_k = n$, and n is the number of items to be summed. For each l_i, we can use the formulae (8)−(19) to calculate the combinations of n triplet functions. In calculating Equation (20), it is assumed that at each step of pairwise calculation, the focusing operator will be applied to obtain a new triplet function, and a final decision to be made on a class assignment for an instance is based on the maximum selection of the belief given by Equation (20).

5 Experimental Evaluation

5.1 Experimental Settings

In our experiments, we used thirteen data sets downloaded from the UCI machine learning repository [7]. All the selected data sets have at least three or more classes as required by the evidential structures. The details about these data sets can be found in Table 1.

Table 1. The general description about the datasets

Dataset	Instance No	Classes	Attribute
anneal	798	6	38
audiology	200	23	69
balance	625	3	4
car	1728	4	6
glass	214	7	9
autos	205	6	25
iris	150	3	4
letter	20000	26	16
heart	303	5	13
segment	1500	7	19
soybean	683	19	35
wine	178	3	13
Zoo	101	7	17

For base (individual) classifiers, we used thirteen learning algorithms which all are taken from the Waikato Environment for Knowledge Analysis (Weka) version 3.4 (see Table 2). These algorithms were simply chosen on the basis of their performance in three randomly picked data sets. For meta classifiers – stacking, we chose the multi-response linear regression (MLR) and we also chose AdaBoostM1 to compare with our method. Parameters used for each algorithm were at the Weka default settings. The details about these algorithms can be found in [22].

Table 2. The general description about thirteen learning algorithms

No	Classifier	No	Classifier
0	AOD	1	NaiveBayes
2	SOM	3	IB1
4	IBk	5	KStar
6	DecisionStump	7	J48
8	RandomForest	9	DecisionTable
10	JRip	11	NNge
12	PART		

The experiments were performed on a three partition scheme using a ten-fold cross validation to avoid overfitting to some extent[1]. We divided the data sets into 10 mutually exclusive sets. For each fold, after the test set removal, the training set were further subdivided into 70% for a new training set and a 30% validation set. Apart from the evaluation of the performance of individual classifiers, the validation set is used to select the best combination of classifiers. The performance of the combinations of selected classifiers (the best ensembles of

[1] Two partition with a ten-fold cross validation was used for MLR and AdaBoostM1.

classifiers) using DS, majority voting (MV) and linear sum (SUM) combination methods is evaluated on the testing set.

Eight groups of experiments are reported here, which were done individually and in combination across all the thirteen data sets. These include 1) assessing all the algorithms as shown in Table 2; 2) combining the individual classifiers using DS, in which the outputs are represented by triplet functions and full list of probabilities (prior to transforming them to triplets); 3) combining the individual classifiers represented by the dichotomous functions using DS, where belief mass functions were defined on the basis of the performance of classifiers in terms of recognition, substitution and rejection rates [23]; 4) combining the individual classifiers to construct the ensembles using MV, in which the individual outputs were in single classes; 5) combining the individual classifiers using SUM, in which the classifier outputs were in the form of full list of probabilities; 6) combining J48, NaiveBayes, MLR and KStar by MLR as used in [17]; 7) combining the best, the second best and the third best individual classifiers (SMO, IBk and NNge) by MLR; and 8) experimenting on AdaBoostM1 with SMO − the best individual classifier across the thirteen data sets.

To compare the classification accuracies between the individual classifiers and the combined classifiers across all the data sets, we employed the ranking statistics in terms of *win/draw/loss* record [14]. The win/draw/loss record presents three values, the number of data sets for which classifier A obtained better, equal, or worse than classifier B with respect to a classification accuracy. All collected classification accuracies were measured by the averaged F-measure [22]. A paired t-test across all these domains were also carried out to determine whether the differences between the base classifiers and combined classifiers are statistically significant at the 0.05 level.

5.2 Experimental Results

The ten experimental results are summarized in Tables 3 and 4, respectively, including all the classification accuracies of the best individual classifiers and the best combined classifiers. Table 3 presents the classification accuracies of the best individual classifiers and the best combined classifiers using DS under the four different structures. Table 4 gives the best combined classifiers using MV, SUM, boosting and stacking. In these tables, each cell represents either the accuracy of a best ensemble or the best individual classifier on the corresponding data set. If the difference between the best combined classifier and the best individual on the same data set is statistically significant, then the larger of the two is shown in *bold*.

The bottoms of Tables 3 and 4 provide summary statistics of comparing the performance of the best base classifiers with the best combined classifiers across the data sets. From these summaries, it can be observed that the accuracy of the combined classifiers based on the triplet structure using DS is better than the eight others on average. It has more wins to losses over the dichotomous, Fullist and the best combined classifiers using MV, SUM, boosting and stacking compared with the best individual classifiers. This finding is further supported

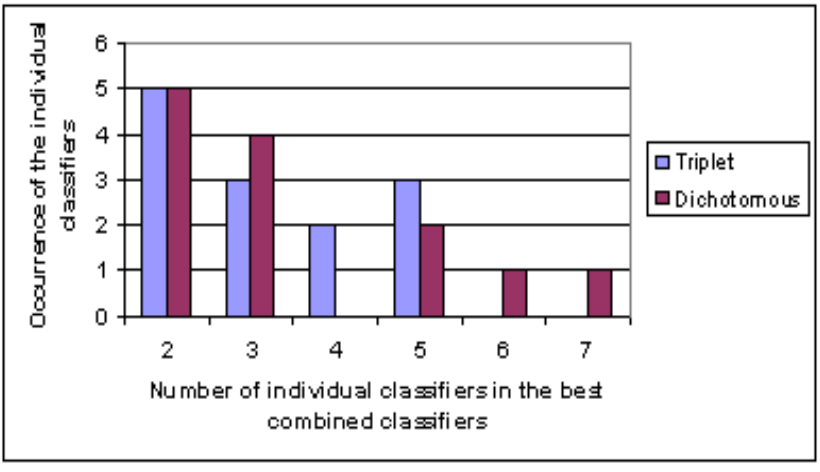


Fig. 2. Ensemble size: number of individual classifiers involved in the best combined classifiers across all the data sets with the evidential structures of triplet and dichotomous

Table 3. The classification accuracies of the best INDIVIDUAL classifier, best combined classifiers based on the different structures of TRIPLET, DICHOTOMOUS and FULLIST using DS over the thirteen data sets

Dataset	Individual	Triplet	Dichotoms	Fullist
Anneal	80.23	**81.57**	80.68	69.88
Audiology	48.67	**57.44**	51.97	49.32
Balance	65.67	63.17	65.67	21.68
Car	89.62	**94.29**	**91.92**	90.03
Glass	65.36	66.81	66.26	62.75
Autos	77.59	79.28	78.78	66.80
Iris	95.33	**96.67**	**96.67**	60.71
Letter	92.05	**92.91**	**93.41**	68.38
Cleveland	35.48	37.09	35.37	34.26
Segment	96.69	97.35	**97.74**	88.40
Soybean	95.89	96.88	96.85	91.11
Wine	98.90	**100.00**	98.90	96.70
Zoo	90.62	**93.61**	**93.61**	64.39
Average	79.39	81.31	80.60	66.49
Win/Draw/Loss		12/0/1	10/2/1	2/0/10
Significant win		7	5	0

by the statistically significant wins in which the triplet has two more wins than the dichotomous, three more wins than MV, four more wins than AdaBoost.M1, and seven more wins than Fullist and MLR.

Fig. 2 presents the sizes of the best combined classifiers across all the data sets. With the different structures, the construction of these ensembles involves

Table 4. The classification accuracies of the best combined classifiers using MV, SUM, AdaBoost.M1 (BOOSTING corresponds to setting (8) in Section 5.1), and MLRs (STACK1,2 correspond to the settings (6) and (7) in Section 5.1) over the thirteen data sets

Dataset	MV	Sum	Boosting	Stack1	Stack2
Anneal	**81.14**	80.51	77.35	72.77	75.34
Audiology	**54.30**	53.72	45.16	32.89	32.19
Balance	62.72	63.17	**93.17**	62.73	**68.49**
Car	**91.75**	91.41	**92.60**	86.18	90.03
Glass	66.69	66.40	65.97	58.41	57.77
Autos	77.94	78.28	77.32	75.34	77.32
Iris	**96.67**	95.33	**98.00**	94.67	94.00
Letter	92.77	92.89	92.53	92.03	92.53
Cleveland	34.37	**37.64**	31.91	35.13	31.87
Segment	96.55	**97.68**	96.57	96.59	95.85
Soybean	96.17	96.60	95.50	95.25	95.20
Wine	98.97	99.42	98.38	98.90	98.32
Zoo	**93.61**	**93.61**	89.43	82.57	83.64
Average	80.28	80.51	81.07	75.65	76.35
Win/Draw/Loss	10/0/3	11/1/1	5/0/8	0/1/12	3/0/10
Significant win	4	3	3	0	0

2-7 combinations of classifiers, and among them most of these combinations only involves two classifiers. This result is consistent with previous studies conducted in [4] and [20], and different from ones presented in [15] and [14] where their experiments showed that ensemble accuracy increased with ensemble size and the performance levels out with ensemble sizes of 10-25. Our experimental results uncover that due to the different way of constructing ensembles, the sizes of ensembles constructed by different learning algorithms working on a single data set is not necessarily the same as that constructed by single learning methods which manipulate different portion of features or instances of the data set.

6 Discussion

Perhaps one of the most important difference between the triplet-based class-indifferent method from the single class label methods of AdaBoost.M1 and MV lies in the fact that it is motivated or built around the appealing intuition that the support contained in discarded classes −the second best decision − could help improve the combined performance of classifiers. More precisely the combined effect of two triplet classifiers will be affected by the first and second elements along with ignorance. Therefore the use of support from the other classes may play an important role in overcoming the single error produced by a classifier that repeatedly provides the high confidence values of classes as occurred in AdaBoost.M1 and MV. The empirical results show this property is appealing.

Now we look at a theoretical justification on our claim. We state formally the conditions for the first or second decision in either of two triplets to become the best supported decision. Assume that two triplet functions m_1 and m_2 fall into the category where a pair of singletons $\{x_1\}, \{y_1\}$ is equal to a pair of $\{x_2\}, \{y_2\}$, i.e., $x = x_1 = x_2$ and $y = y_1 = y_2$ (see Section 4.1). By using formulae (2), we have the following inequality when x is the best choice:

$$\begin{aligned} m_1(\{x\})m_2(\{x\}) + m_1(\{x\})m_2(\Theta) + m_1(\Theta)m_2(\{x\}) > \\ m_1(\{y\})m_2(\{y\}) + m_1(\{y\})m_2(\Theta) + m_1(\Theta)m_2(\{y\}) \end{aligned} \tag{21}$$

Substituting for Θ in formula (21) and rearranging it, we have the condition for the best support of x below:

$$m_1(\{x\}) > 1 - \frac{[1 - m_1(\{y\})][1 - m_2(\{y\})]}{[1 - m_2(\{x\})]} \tag{22}$$

Likewise we can derive the condition for y being the best supported decision:

$$m_2(\{y\}) > 1 - \frac{[1 - m_1(\{x\})][1 - m_2(\{x\})]}{[1 - m_1(\{y\})]} \tag{23}$$

We can obtain the conditions for the other two cases in two triplets in a similar manner.

The idea of MLR is to learn a single combination model for all the classes from the outputs of original classifiers using linear regression. It can be regarded to as a class-indifferent method. Our experimental results show that the performance of DS is significantly better than that of MLR and the combined classifiers by MV also outperform MLR that is consistent with the results on multi-class data presented in [17]. The performance degradation of MLR might confirm its limited ability to determine decision boundaries for the multi-class data. Our experimental results also demonstrate this limitation could be improved to some extent by integrating the individual classifiers which are better performed.

With respect to the dichotomous structure, the drawback of that method is the way of measuring evidence, it ignores the fact that classifiers normally do not have the same performance on different classes, which could cause a deterioration in the performance of the combined classifiers.

Additionally compared with the performance of combined classifiers under the full list using DS, it can be observed that SUM is significantly better than DS. This finding suggests that to achieve the better performance of combining classifiers using DS, it is necessary to approximate the full list of probabilities (numeric scores) into some kind of structures, such as triplet. Inspecting the calculation process of Dempster's rule, it is not a surprise that the performance of combining classifiers in the form of triplets is better than that in a full list of decisions, because when a large number of decisions from each classifier are involved in combinations, it increases possibilities of conflict between two respective decisions − the larger the conflict, the poorer the performance of combined classifiers. This result is somehow consistent with the finding given in [13].

7 Conclusion

We have presented a formalism for modelling classifier outputs in terms of *triplets* and the general formulae for combining base classifiers represented in the form of triplets which underpin our class-indifferent combination method. A wide range of experiments have been carried out over the thirteen UCI data sets. Our results show that the performance of the best combined classifiers is better than that of the best individuals at most of the data sets and the corresponding ensemble sizes are 2-7 where the combinations of 2 and 3 classifiers take 61.5% in the thirteen best ensembles. The comparative analysis among the structures of triplet, dichotomous and full list show that the triplet is the best one, and compared with the combinations methods of MV, SUM, MLR and AdaBoost.M1, DS is better than the others in combining the individual classifiers.

A decision rule defined on the basis of the DS theory of evidence is different from the majority voting principle. DS makes use of evidence accumulated from multiple classifiers in the process of classifier combination. It not only considers the majority agreement on the decisions received from classifiers, but it also incorporates the degrees of belief associated with those decisions into the decision making process. So it provides an effective means to reconcile decisions made by multiple classifiers.

References

1. Al-Ani, A., Deriche, M.: A new technique for combining multiple classifiers using the dempster-shafer theory of evidence. Journal of Artificial Intelligence Research 17, 333–361 (2002)
2. Altincay, H.: On the independence requirement in Dempster-Shafer theory for combining classifiers providing statistical evidence. Journal of Applied Intelligence 25, 73–90 (2006)
3. Barnett, J.A.: Combining opinions about the order of rule execution. In: AAAI, pp. 477–481 (1991)
4. Bi, Y., McClean, S.I., Anderson, T.: On combining multiple classifiers using an evidential approach. In: Proc of AAAI 2006, pp. 324–329 (2006)
5. Bi, Y., Bell, D., Wang, H., Guo, G., Guan, J.: Combining Multiple Classifiers for Text Categorization using Dempster's rule of combination. Journal of Applied Artificial Intelligence 21, 211–239 (2007)
6. Bi, Y., Wu, S., Guo, G.: Combining Prioritized Decisions in Classification. In: The International Conference of Modelling Decisions for Artificial Intelligence (2007)
7. Blake, C.L., Keogh, C.J.E.: Uci repository of machine learning databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`
8. Dietterich, T.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
9. Duin, R.P.W., Tax, D.M.J.: Experiments with classifier combining rules. In: Kittler, J., Roli, F. (eds.) Multiple Classifier Systems, pp. 16–29 (2000)
10. Dzeroski, S., Zenko, B.: Is combining classifiers with stacking better than selecting the best one? Machine Learning 54(3), 255–273 (2004)
11. Duin, R.P.W., Kittler, J., Hatef, M., Matas, J.: On combining classifiers. IEEE Trans. on pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)

12. Kuncheva, L.: Combining classifiers: Soft computing solutions. In: Pal, S.K., Pal, A. (eds.) Pattern Recognition: From Classical to Modern Approaches, pp. 427–451 (2001)
13. Kuncheva, L., Whitaker, C.J.: Measures of diversity in classifier ensembles. Machine Learning 51, 181–207 (2003)
14. Melville, P., Mooney, R.J.: Constructing diverse classifier ensembles using artificial training examples. In: Proc. of IJCAI 2003, pp. 405–510 (2003)
15. Opitz, D.: Feature selection for ensembles. In: Proc. of AAAI 1999, pp. 379–384. AAAI Press, Menlo Park (1999)
16. Schapire, R.E.: A brief introduction to boosting. In: Proc. of IJCAI 1999, pp. 1401–1406 (1999)
17. Seewald, A.K.: How to make stacking better and faster while also taking care of an unknown weakness. In: Proceedings of ICML 2002, pp. 554–561 (2002)
18. Shafer, G., Logan, R.: Implementing Dempster's Rule for Hierarchical Evidence. Artificial Intelligence 33(3), 271–298 (1987)
19. Shafer, G.: A Mathematical Theory of Evidence, 1st edn. Princeton University Press, Princeton (1976)
20. Tsymbal, A., Pechenizkiy, M., Cunningham, P.: Sequential Genetic Search for Ensemble Feature Selection. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005) (2005)
21. Tumer, K., Robust, G.J.: On combining classifiers. Pattern Analysis and Applications 6(1), 41–46 (2002)
22. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
23. Xu, L., Krzyzak, A., Suen, C.Y.: Several methods for combining multiple classifiers and their applications in handwritten character recognition. IEEE Trans. on System, Man and Cybernetics 2 (3), 418–435 (1992)
24. Ault, T., Yang, Y., Pierce, T.: Combining multiple learning strategies for effective cross validation. In: Proc of ICML 2000, pp. 1167–1182 (2000)

An Improved Multi-task Learning Approach with Applications in Medical Diagnosis

Jinbo Bi[1], Tao Xiong[2], Shipeng Yu[1], Murat Dundar[1], and R. Bharat Rao[1]

[1] CAD and Knowledge Solutions, Siemens Medical Solutions
20 Valley Stream Parkway, Malvern, PA 19355, USA
jinbo.bi@siemens.com
[2] Risk Management, Applied Research, eBay Inc.
2145 Hamilton Avenue, San Jose, CA 95125, USA

Abstract. We propose a family of multi-task learning algorithms for collaborative computer aided diagnosis which aims to diagnose multiple clinically-related abnormal structures from medical images. Our formulations eliminate features irrelevant to all tasks, and identify discriminative features for each of the tasks. A probabilistic model is derived to justify the proposed learning formulations. By equivalence proof, some existing regularization-based methods can also be interpreted by our probabilistic model as imposing a Wishart hyperprior. Convergence analysis highlights the conditions under which the formulations achieve convexity and global convergence. Two real-world medical problems: lung cancer prognosis and heart wall motion analysis, are used to validate the proposed algorithms.

1 Introduction

Physicians routinely use computer aided diagnosis (CAD) systems in clinical practice [1]. It is well accepted that CAD systems decrease detection and recognition errors when used as a second reader [2]. Typically, the goal of a CAD system is to detect potentially abnormal structures in medical images. However, most CAD systems focus on the diagnosis of a single isolated abnormality using images taken only for the specific disease, which neglects a fundamental aspect of physicians diagnostic workflow where they examine not only primary abnormalities but also symptoms of related diseases.

For instance, an automated lung cancer CAD system can be built to separately identify solid nodules and ground glass opacities (GGOs). (Patients can have both structures, or GGOs can later become calcified GGOs which become solid or partly-solid nodules.) Radiologic classification of small adenocarcinoma of lung by means of thoracic thin-section CT discriminates between solid nodules and GGOs. Fig. 1 shows two CT slices with a nodule and a GGO respectively. A solid nodule is defined as an area of increased opacification more than 5mm in diameter, which completely obscures underlying vascular markings. A ground-glass opacity (GGO) is defined as an area of a slight homogeneous increase in density, which does not obscure underlying vascular markings [3]. Detecting

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 117–132, 2008.

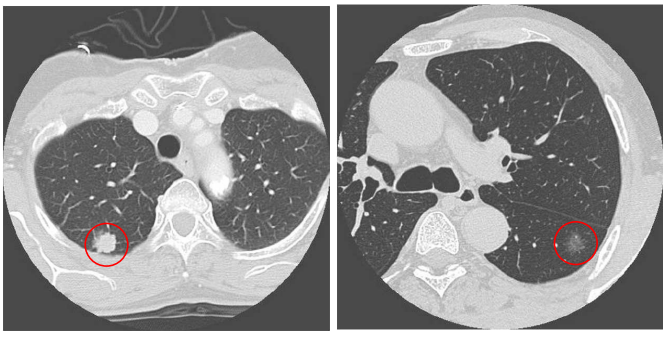


Fig. 1. Lung CT images: left, solid nodule; right, ground glass opacity (GGO)

nodules and detecting GGOs are two dependent tasks with their own respective characteristics, and is thus more sensible to be tackled jointly.

Another example is the wall motion analysis of the left ventricle which is used to diagnose ischemia diseases. The left ventricle wall is medically segmented into 16 segments. Fig. 2 shows an ultrasound image of left ventricle in the apical four chamber (A4C) view and the six segments seen from this view. The task is to predict the wall motion abnormality of each segment by extracting features from cardiac ultrasound images and classifying each segment as being normal versus abnormal. Left ventricle segments are physically connected, and if any segment has abnormalities, the neighboring segments are affected, which makes jointly learning the classifiers both necessary and beneficial.

We introduce a concept – "collaborative" computer aided diagnosis (CCAD) – that aims to improve the diagnosis of a single abnormality by fusing information, knowledge or data from various related sources, such as detecting nodules not only by itself but also by learning from multiple related abnormal structures simultaneously. In the machine learning field, the collaborative learning problem

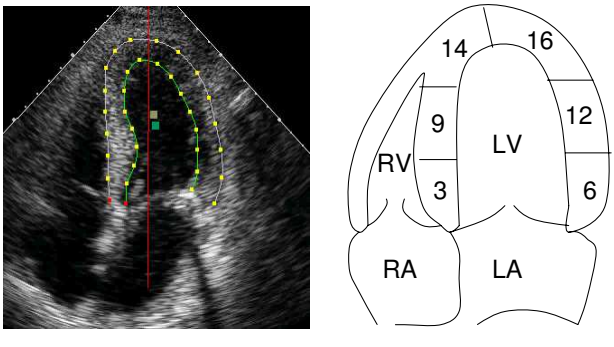


Fig. 2. Ultrasound image of heart: left, ultrasound image of A4C view; right, segments of left ventricle seen from A4C view

has been cast as multi-task learning (MTL), collaborative filtering or collaborative prediction. Multi-task learning is a learning methodology that estimates models for several tasks in a joint manner. Although almost all existing multi-task learning methods assume some relatedness among tasks, the definition of relatedness varies [4,5,6]. From the hierarchical Bayesian viewpoint [7], multi-task learning essentially seeks to learn a good prior over all tasks to capture task dependencies.

We model the across-task relatedness as sharing a common feature or kernel representation. Dimension reduction or sparse kernel representation is essential for CAD applications. Previous work on selecting features for multiple related tasks include the work in [8] that is based on maximum entropy discrimination, and the regularization-based methods [9,10] by applying a joint regularization of the model parameters. We derive a family of effective approaches, which generalizes our early multi-task learning study [12], by directly maximizing the joint *a posterior* distribution across tasks. By imposing a hyperprior that corresponds to a trace norm constraint [11] on model parameter variance, we are able to eliminate features irrelevant to all tasks as well as select discriminative features for each individual task.

2 MTL Algorithms

Assume that we have T tasks in total, for each task t, we have sample set $(\mathbf{X}_t \in R^{\ell_t \times d}, \mathbf{y}_t \in R^{\ell_t})$. The matrix $\mathbf{X}_t$ is the feature matrix or kernel matrix where the i-th row corresponds to the i-th example $\mathbf{x}_i^t$ of task t, and each column represents a feature or a kernel basis, and $\mathbf{y}_t$ denotes the label vector where the i-th component is y_i^t. We consider functions of the form $\mathbf{x}^\top \boldsymbol{\alpha}$ which is linear in terms of the model parameter $\boldsymbol{\alpha}$. We focus on models where $\mathbf{x}$ is in the original feature space but many discussions in this article can be extended to kernel spaces.

To learn the parameter vector α_t, single task learning methods minimize a regularized risk $L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t) + \lambda P(\boldsymbol{\alpha}_t)$ for an optimal $\boldsymbol{\alpha}_t$ where P is a regularization operator, such as a 2-norm penalty on $\boldsymbol{\alpha}_t$, i.e., $\sum_{j=1}^d (\alpha_{tj})^2$, or a 1-norm penalty, $\sum_{j=1}^d |\alpha_{tj}|$, L defines the loss term, and λ balances between L and P. For example, the logistic regression loss

$$L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t) = \sum_{i=1}^{\ell_t} \log(1 + \exp(-\sum_{j=1}^{d} \alpha_{tj} x_{ij}^t y_i^t)) \tag{1}$$

and the least squares loss

$$L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t) = \sum_{i=1}^{\ell_t} (\sum_{j=1}^{d} \alpha_{tj} x_{ij}^t - y_i^t)^2 \tag{2}$$

are both strictly convex functions in terms of $\boldsymbol{\alpha}_t$.

A family of joint learning algorithms can be derived by rewriting $\boldsymbol{\alpha}_t = \mathbf{C}\boldsymbol{\beta}_t$ where $\mathbf{C}$ is a diagonal matrix with diagonal vector equal to $\mathbf{c} \geq 0$, and we solve the following problem over all tasks:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\beta}_t, t=1,\cdots,T, \mathbf{c}\geq 0} & \sum\limits_{t=1}^{T} \left(L(\mathbf{C}\boldsymbol{\beta}_t, \mathbf{X}_t, \mathbf{y}_t) + P_1(\boldsymbol{\beta}_t)\right) \\ \text{subject to} & P_2(\mathbf{c}) \leq \gamma, \end{array} \tag{3}$$

where P_1 and P_2 are any suitable regularization operators. For each task t, solving problem (3) constructs a function $f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\alpha}_t = \mathbf{x}^\top \mathbf{C}\boldsymbol{\beta}_t = \sum_j x_j c_j \beta_{tj}$ where $\boldsymbol{\beta}_t$ is task-specific while the same $\mathbf{c}$ is used across different tasks. We call $\mathbf{c}$ an indicator vector indicating if an according feature is used in the model. Typically $\mathbf{c}$ comprises entries that are equal to 0 or 1, which leads to difficult combinatorial optimization problems, and thus has been relaxed to non-negative real values in Problem (3). If $c_j = 0$, the j-th variable is not used in any model for all tasks regardless of the value of a specific $\boldsymbol{\beta}$. Otherwise if $c_j > 0$, the j-th variable appears in all models but an appropriate $\boldsymbol{\beta}$ vector can rule out this feature for a particular task. In other words, $\mathbf{c}$ is used to eliminate any irrelevant features, and $\boldsymbol{\beta}_t$ selects the best suitable features for each individual task.

Many regularization terms can be considered for the choices of P_1 and P_2. For example, if the 2-norm regularization is employed for both P_1 and P_2, the problem (3) becomes

$$\begin{array}{ll} \min\limits_{\boldsymbol{\beta}_t, t=1,\cdots,T, \mathbf{c}\geq 0} & \sum\limits_{t=1}^{T} \left(L(\mathbf{C}\boldsymbol{\beta}_t, \mathbf{X}_t, \mathbf{y}_t) + \sum_{j=1}^{d} \beta_{tj}^2\right) \\ \text{subject to} & \sum_{j=1}^{d} c_j^2 \leq \gamma, \end{array} \tag{4}$$

where $\gamma > 0$ is a tuning parameter. Empirical results included in [12] demonstrate the effectiveness of the formulation (3) with $P_1(\cdot) = \sum_{j=1}^{d} \beta_{tj}^2$ and $P_2(\cdot) = \sum_{j=1}^{d} |c_j|$.

To effectively optimize (3), we design an alternating optimization algorithm, which is, in spirit, similar to the Expectation-Maximization approach. At iteration s, the "E" step estimates the optimal $\mathbf{c}^s$, which serves the common prior, based on $\boldsymbol{\beta}^{s-1}$. The "M" step estimates a new $\boldsymbol{\beta}_t^s$ for each t by maximizing the posterior based on $\mathbf{c}^s$. The algorithm does the following steps at the s-th iteration:

Algorithm $\mathcal{A}(\mathbf{C}^{s-1}, \ \boldsymbol{\beta}_t^{s-1}, t = 1, \cdots, T)$

– Fix $\mathbf{C} = \mathbf{C}^{s-1}$ (initially, to $\mathbf{I}$), convert $\tilde{\mathbf{X}}_t \leftarrow \mathbf{X}_t\mathbf{C}$, solve (5) for $\boldsymbol{\beta}_t^s$,

$$\forall\ t = 1, \cdots, T, \min_{\boldsymbol{\beta}_t} L(\boldsymbol{\beta}_t, \tilde{\mathbf{X}}_t, \mathbf{y}_t) + P_1(\boldsymbol{\beta}_t). \tag{5}$$

– Fix $\boldsymbol{\beta}_t = \boldsymbol{\beta}_t^s$, convert $\hat{\mathbf{X}}_t \leftarrow \mathbf{X}_t\mathbf{B}_t$ where $\mathbf{B}_t$ is a diagonal matrix with diagonal elements equal to $\boldsymbol{\beta}_t^s$, solve problem (6) for $\boldsymbol{c}^s$,

$$\min_{\mathbf{c}\geq 0} \ L(\mathbf{c}, \hat{\mathbf{X}}_t, \mathbf{y}_t), \qquad \text{subject to} \ \ P_2(\mathbf{c}) \leq \gamma. \tag{6}$$

---

3 A Statistical Justification

Let $p(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t)$ specify the likelihood for task t, with a noise model which is independent of other tasks. Here $\mathbf{w}_t$ is the model parameter to be determined. In a hierarchical Bayesian framework, we assume all the function weights $\mathbf{w}_t$ are *i.i.d.* sampled from a common prior $p(\cdot)$, which accounts for the dependencies between different tasks. Typically a zero mean Gaussian prior with covariance $\mathbf{\Sigma}$ is assigned to the weights $\mathbf{w}_t$, i.e., $\mathbf{w}_t \sim N(\mathbf{0}, \mathbf{\Sigma})$. Then the *a posteriori* distribution of all function coefficients $\{\mathbf{w}_t\}$ can be calculated via Bayes rule as, $p(\mathbf{W}|\mathbf{X}, \mathbf{y}, \mathbf{\Sigma}) \propto \prod_t p(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t)p(\mathbf{w}_t|\mathbf{\Sigma})$ where $\mathbf{W}$ is a matrix containing all weight vectors $\mathbf{w}_t$ as rows, and $\mathbf{X}$, $\mathbf{y}$ contain data from all tasks.

We are interested in learning the shared covariance matrix $\mathbf{\Sigma}$ rather than fixing it. A Bayesian treatment would be to assign a hyperprior to $\mathbf{\Sigma}$ and learn $\mathbf{W}$ and $\mathbf{\Sigma}$ jointly. Since $\mathbf{\Sigma}$ is symmetric and positive definite, one choice of the prior is $p(\mathbf{\Sigma}) \propto |\mathbf{\Sigma}|^{T/2} \exp\left(-\frac{1}{2}\operatorname{tr}(\mathbf{\Sigma})\right)$, with $\operatorname{tr}(\cdot)$ the matrix trace. This is essentially a Wishart distribution with degrees of freedom $d + T + 1$ and scale matrix $\mathbf{I}$. Now the joint *a posteriori* distribution of $(\mathbf{W}, \mathbf{\Sigma})$ is

$$p(\mathbf{W}, \mathbf{\Sigma}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{\Sigma}) \prod_t p(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t)p(\mathbf{w}_t|\mathbf{\Sigma}). \tag{7}$$

The *maximum a posteriori* (MAP) estimate is to find a point estimate that maximizes the posterior (7). This is equivalent to solving the following optimization problem $\min_{\mathbf{w}_t, \mathbf{\Sigma}} \quad \sum_{t=1}^{T} \left(L(\mathbf{w}_t, \mathbf{X}_t, \mathbf{y}_t) + \mathbf{w}_t^\top \mathbf{\Sigma}^{-1} \mathbf{w}_t \right) + \operatorname{tr}(\mathbf{\Sigma})$ by taking the negation of the logarithm of (7) and removing the normalization constant. Here the loss function for each task t is $L(\mathbf{w}_t, \mathbf{X}_t, \mathbf{y}_t) \propto -\log p(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t)$. This can also be equivalently written as

$$\begin{array}{ll} \min\limits_{\mathbf{w}_t, t=1,\cdots,T, \mathbf{\Sigma}} & \sum\limits_{t=1}^{T} L(\mathbf{w}_t, \mathbf{X}_t, \mathbf{y}_t) + \mathbf{w}_t^\top \mathbf{\Sigma}^{-1} \mathbf{w}_t, \\ \text{subject to} & \operatorname{tr}(\mathbf{\Sigma}) \leq \gamma \end{array} \tag{8}$$

with an appropriately chosen $\gamma > 0$. For each of the task t, this trace condition essentially requires that the expected variance of each weight component w_t^j of $\mathbf{w}_t, \forall t$, is proportional to γ. With a small γ, some components will become small to achieve *sparse estimates* of $\mathbf{w}_t$. Thus this formulation leads to a *jointly sparse structure* of the weight matrix $\mathbf{W}$.

If we decompose the matrix $\mathbf{\Sigma}$ to its eigen-form $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ where $\mathbf{U}$ is an orthonormal matrix and the diagonal matrix $\mathbf{\Lambda}$ contains eigen-values $\sigma_j \geq 0$, the problem (8) becomes an equivalent form with $\boldsymbol{\alpha}_t = \mathbf{U}^\top \mathbf{w}_t$:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}_t, t=1,\cdots,T, \mathbf{U}, \boldsymbol{\sigma} \geq 0} & \sum\limits_{t=1}^{T} \left(L(\boldsymbol{\alpha}_t, \mathbf{X}_t\mathbf{U}, \mathbf{y}_t) + \sum\limits_{j=1}^{d} \frac{1}{\sigma_j} \alpha_{tj}^2 \right), \\ \text{subject to} & \sum\limits_{j=1}^{d} \sigma_j \leq \gamma. \end{array} \tag{9}$$

Problem (9) implies that the original input $\mathbf{x}$ has been transformed to $\mathbf{U}^\top \mathbf{x}$ and then a linear function is constructed in the transformed space where features are independent.

Many image features in CAD applications are computationally expensive, so one of the major goals is to reduce the number of image features in the models. Since the resulting orthonormal $\mathbf{U}$ may not be sparse, we assume $\mathbf{U} = \mathbf{I}$ to enforce the sparsity on the original image features instead of sparse representations in the transformed space. By showing equivalence between (4) and (9), the probabilistic model in this section provides a statistical justification for our algorithms.

Theorem 1. *For any optimal solution of Problem (9) where* $\mathbf{U} = \mathbf{I}$*, there is a corresponding optimal solution to Problem (4), and vice versa.*

The proof can be obtained by change of variables as follows: $\beta_{tj} = \alpha_{tj}/\sqrt{\sigma_j}$, $\forall t = 1, \ldots, T$, $c_j = \sqrt{\sigma_j}$, $j = 1, \ldots, d$. Correspondingly, $\alpha_{tj} = c_j \beta_{tj}$ and $\sum_j c_j^2 = \sum_j \sigma_j \leq \gamma$.

4 Connection to Existing Methods

Feature selection for multi-task learning using a joint regularization has been recently proposed in [9] where a so-called ℓ_1/ℓ_2 norm is applied to the weight matrix $\mathbf{A}$ formed by all $\boldsymbol{\alpha}_t$ as rows. A more recent work [10] dedicated to multi-task feature learning has defined a new norm as $\|\mathbf{A}\|_{2,1} = \sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2}$, which is the same as the ℓ_1/ℓ_2 norm in [9]. Assuming $\mathbf{U} = \mathbf{I}$, both work essentially solves the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_t, t=1,\cdots,T} \sum_{t=1}^{T} L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t), \\ &\text{subject to } \sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2} \leq \kappa, \end{aligned} \tag{10}$$

or an equivalent problem as follows

$$\min_{\boldsymbol{\alpha}_t, t=1,\cdots,T} \sum_{t=1}^{T} L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t) + \lambda \left(\sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2} \right)^2, \tag{11}$$

where κ and λ are pre-specified parameters, and these two problems are equivalent for properly chosen κ and λ. The problem in [9] does not use the squared regularized term as in problem (10) whereas the formulation in [10] uses the square of $\|\mathbf{A}\|_{2,1}$ as the second term in problem (11).

As shown in [9,10], these regularization-based algorithms advance the multi-task learning research, but there has been lack of investigation if a probabilistic interpretation exists for these methods. The following theorem characterizes the connection between our formulation and (11). Hence, our probabilistic model is also feasible to justify these approaches that these methods assume a Wishart prior on the common covariance $\boldsymbol{\Sigma}$ of function weights $\mathbf{w}_t, \forall t$.

Theorem 2. *The Karush-Kuhn-Tucker (KKT) conditions of Problem (9) with* $\mathbf{U} = \mathbf{I}$ *are identical to the KKT conditions of Problem (11) for any convex and continuously differentiable loss function* $L(\boldsymbol{\alpha}, \mathbf{X}, \mathbf{y})$.

Proof. The Lagrangian of the problem (9) is:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}_t, \boldsymbol{\sigma}, a, \mathbf{b}) = \textstyle\sum_{t=1}^{T} L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t) + \sum_{t=1}^{T}\sum_{j=1}^{d} \frac{\alpha_{tj}^2}{\sigma_j} \\ + a(\textstyle\sum_{j=1}^{d} \sigma_j - \gamma) - \mathbf{b}^{\mathbf{T}}\boldsymbol{\sigma},\end{aligned}$$

where a and $\mathbf{b}$ are Lagrangian multipliers, and $a \geq 0$ is a scalar and $\mathbf{b} \geq \mathbf{0}$ is a vector.

Problem (9) minimizes a convex objective over a convex feasible region, and thus is a convex program. Then the KKT necessary and sufficient conditions are as follows:

$$\frac{\partial \mathcal{L}}{\partial \sigma_j} = -\textstyle\sum_{t=1}^{T} \frac{\alpha_{tj}^2}{\sigma_j^2} + a - b_j = 0, \tag{12}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_{tj}} = \frac{\partial L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t)}{\partial \alpha_{tj}} + 2\frac{\alpha_{tj}}{\sigma_j} = 0, \tag{13}$$

$$\textstyle\sum_{j=1}^{d} \sigma_j \leq \gamma, \quad a \geq 0, \quad \mathbf{b} \geq 0, \quad \boldsymbol{\sigma} \geq 0 \tag{14}$$

$$a(\textstyle\sum_{j=1}^{d} \sigma_j - \gamma) = 0 \tag{15}$$

$$b_j \sigma_j = 0, \quad j = 1, \ldots, d \tag{16}$$

Now we discuss the various cases in terms of the Lagrange multipliers a and b_j.

(1) If $b_j > 0$, by the complementary condition (16), $\sigma_j = 0$. It implies $\alpha_{\cdot j} = 0$ which denotes that for a specific number j, $\alpha_{tj} = 0, \forall t = (1, \ldots, T)$.
(2) If $b_j = 0$ (implying $\sigma_j > 0$) and $a = 0$, by KKT condition (12), $\sum_{t=1}^{T} \alpha_{tj}^2 = 0$. Hence, $\alpha_{\cdot j} = 0$.
(3) If $b_j = 0$ and $a > 0$ (implying $\sum_j \sigma_j = \gamma$ by (15)), then $a = \frac{1}{\sigma_j^2} \sum_{t=1}^{T} \alpha_{tj}^2$, and further we have $\sigma_j = \gamma \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2} / \sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2}$. Substituting σ_j into KKT condition (13) yields the optimality condition, which can be summarized as follows:

$$\forall (t = 1, \ldots, T, j = 1, \ldots, d),$$
$$\begin{cases} \frac{\partial L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t)}{\partial \alpha_{tj}} + \frac{2}{\gamma} \left(\sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2} \right) (\sum_{t=1}^{T} \alpha_{tj}^2)^{-\frac{1}{2}} \alpha_{tj} = 0 \\ \text{or } \alpha_{\cdot j} = 0, \end{cases}$$

Now let $\lambda = 1/\gamma$ in Problem (11). Due to the convexity of Problem (11), its KKT conditions are necessary and sufficient and can be shown as

$$\forall (t = 1, \ldots, T, j = 1, \ldots, d),$$
$$\begin{cases} \frac{\partial l(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t)}{\partial \alpha_{tj}} + \frac{2}{\gamma} \left(\sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2} \right) (\sum_{t=1}^{T} \alpha_{tj}^2)^{-\frac{1}{2}} \alpha_{tj} = 0 \\ \text{or } \alpha_{\cdot j} = 0, \text{ and } \alpha_{\cdot j} = 0 \in \partial g(\alpha_{\cdot j}), \end{cases}$$

where we use $g(\alpha_{\cdot j})$ to denote the objective function in (11) as a function of $\alpha_{\cdot j}$, and ∂f to denote the subgradient of function g. The use of subgradient is necessary since the objective g becomes nondifferentiable as its argument goes to zero. The equivalence is established by comparing the two sets of KKT conditions.

Theorem 2 establishes an equivalence between the learning formulations (10), (11) and (9) (more precisely (4)). Hence our probabilistic model in Section 3 can interpret all these formulations as assuming a common covariance matrix $\mathbf{\Sigma}$ across $\mathbf{w}_t$, $\forall t$ and employing a Wishart hyperprior on $\mathbf{\Sigma}$. Furthermore, as a byproduct of Theorem 2, a closed-form solution for $\mathbf{c}$ can be further derived to simplify calculation in Algorithm $\mathcal{A}$ specifically for the formulation (4).

Theorem 3. *Given $\hat{\boldsymbol{\beta}}_t$, $t = 1, \cdots, T$, that are optimal solutions of Problem (4),*

$$c_j = \sqrt{\frac{\gamma \sum_{t=1}^{T} \hat{\beta}_{tj}^2}{\sum_{j=1}^{d} \sum_{t=1}^{T} \hat{\beta}_{tj}^2}}, \quad j = 1, \cdots, d$$

are optimal to Problem (4).

Proof. The proof can be obtained by re-examining Theorem 1 from which we have $\alpha_{tj} = \sqrt{\sigma_j}\beta_{tj}$ and $c_j = \sqrt{\sigma_j}$, and Theorem 2 from which we have $\sigma_j = \gamma\sqrt{\sum_{t=1}^{T} \alpha_{tj}^2}/(\sum_{j=1}^{d} \sqrt{\sum_{t=1}^{T} \alpha_{tj}^2})$. Substituting α_{tj} into the formula for σ_j yields $\sigma_j = \gamma \sum_{t=1}^{T} \beta_{tj}^2/(\sum_{j=1}^{d} \sum_{t=1}^{T} \beta_{tj}^2)$. Taking the square root of σ_j yields the formula for c_j.

5 Convergence Analysis

Although alternating optimization has been used to develop many efficient algorithms, the convergence proof does not necessarily exist. Convergence analysis usually encloses local convergence and global convergence properties. Local convergence implies that the algorithm converges to a solution $(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}})$ if being initialized from a close neighborhood of $(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}})$. A global convergence proves that the algorithm converges when initialized at any arbitrary points in the feasible region $\mathcal{S}$.

The local convergence property of Algorithm $\mathcal{A}$ is analyzed for Formulation (4). The key point is the requirement of the local strict convexity of the loss function L with respect to $(\boldsymbol{\beta}_t, \mathbf{c})$.

Theorem 4. *Let $(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}})$ be a local minimizer of Problem (4). If $\exists$ a neighborhood $\mathcal{N}$ of $(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}})$, such that the loss function L has continuous second-order derivatives and is strictly convex in $\mathcal{N}$, then $\exists\ \hat{\mathcal{N}}((\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}}))$ for any initial point in $\hat{\mathcal{N}}((\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}}))$, Algorithm $\mathcal{A}$ converges q-linearly to $(\hat{\boldsymbol{\beta}}_t, \hat{\mathbf{c}})$.*

Proof. Solving Problem (4) is equivalent to minimizing (5) with a properly chosen $\bar{\gamma} > 0$

$$\min_{\boldsymbol{\beta}_t, t=1,\cdots,T, \mathbf{c}\geq 0} \quad g(\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_T, \mathbf{c}) = \sum_{t=1}^{T} L(\mathbf{C}\boldsymbol{\beta}_t, \mathbf{X}_t, \mathbf{y}_t) + \sum_{t=1}^{T}\sum_{j=1}^{d} \beta_{tj}^2 + \bar{\gamma}\sum_{j=1}^{d} c_j^2$$

The objective function g has continuous second-order derivatives with respect to $\boldsymbol{\beta}_t$ and $\mathbf{c}$ and is strict convex in $\mathcal{N}$ due to the local property of the loss function L. Then the local convergence result developed in [13] on unconstrained problems is applied to show our theorem.

Global convergence analysis is usually more difficult and requires stronger conditions. We use the results developed in the mathematical programming field [13,14] to obtain a global convergence analysis which requires that both sub-problems, (5) and (6), have an unique optimal solution. This condition highly relies on the property of loss functions. If strictly convex loss functions are employed, such as the logistic regression loss or least squares loss, the loss term $L(\mathbf{C}\boldsymbol{\beta}_t, \mathbf{X}_t, \mathbf{y}_t)$ is bi-convex with respect to $(\boldsymbol{\beta}_t, \mathbf{c})$, in other words, is strictly convex with respect to $\boldsymbol{\beta}_t$ if $\mathbf{c}$ is fixed, and vice versa. The strict bi-convexity guarantees that sub-problems have an unique solution.

Let us denote the feasible set of the problem (4) as $\mathcal{S}$. In Algorithm $\mathcal{A}$, $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2$ where $\mathcal{S}_1$ is the feasible region for $\mathbf{c}$, $\mathcal{S}_1 = \{\mathbf{c} \mid \|\mathbf{c}\|^2 \leq \gamma\}$, and $\mathcal{S}_2$ is the feasible region for $\boldsymbol{\beta}_t,\ t = 1, \cdots, T$. Problem (4) has a 2-norm regularization on $\boldsymbol{\beta}$, so each $\boldsymbol{\beta}_t$ has to remain in the set $\mathcal{S}_2 = \{(\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_T) | \sum_{t=1}^{T}\sum_{j=1}^{d} \beta_{tj}^2 \leq \tilde{\gamma}\}$ for a $\tilde{\gamma} > 0$.

Theorem 5. *Let Ω be the set of fixed points of $\mathcal{A}$ as*

$$\{(\mathbf{c}, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_T) \in \mathcal{S} \mid (\mathbf{c}, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_T) = \mathcal{A}(\mathbf{c}, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_T)\}.$$

If the loss function L is strictly convex in terms of $\boldsymbol{\beta}_t,\ \forall\ t$, for fixed $\mathbf{c}$ and is also strictly convex in terms of $\mathbf{c}$ for fixed $\boldsymbol{\beta}$, and the regularizers P_1 and P_2 are strictly convex respectively in terms of $\boldsymbol{\beta}_t$ and $\mathbf{c}$, then for any initial point in $\mathcal{S}$, Algorithm $\mathcal{A}$

(i) either converges to Ω;

(ii) or the limit of every convergence subsequence is in Ω.

Proof. To achieve the results (i) or (ii), the theorem shown in [13,14] requires the following conditions: (a) each sub-problem in $\mathcal{A}$ has an unique optimal solution; (b) $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2$, where each $\mathcal{S}_i$ is a compact subset in a real space of proper dimension. We thus validate the satisfaction of these conditions.
Since the objective function of the unconstrained equivalent form (5) is strictly convex in terms of one set of variables when fixing the other due to the strict convexity of L, P_1 and P_2, (a) holds. Obviously, $\mathcal{S}_1 = \{\mathbf{c} | P_2(\mathbf{c}) \leq \gamma\}$ is a closed and bounded ball in a d-dimensional space, and $\mathcal{S}_2 = \{\boldsymbol{\beta} | \sum_t P_1(\boldsymbol{\beta}_t) \leq \tilde{\gamma}\}$ defines a closed and bounded ball in a $(d \times T)$-dimensional space. Thus both sets are compact subsets of a real space.

Some common loss functions satisfy the conditions in Theorem 5. For example, in the logistic regression loss function (1) and the least squares loss function (2), $L(\boldsymbol{\alpha}_t, \mathbf{X}_t, \mathbf{y}_t)$ is strictly convex in terms of $\boldsymbol{\alpha}_t$. Hence $L(\mathbf{C}\boldsymbol{\beta}_t, \mathbf{X}_t, \mathbf{y}_t)$ is strictly convex in terms of $\mathbf{c}$ if all $\boldsymbol{\beta}_t$ are fixed, and in terms of $\boldsymbol{\beta}_t$ if $\mathbf{c}$ is fixed. For such a loss function, the global convergence holds.

Particularly, Problem (4) is equivalent to Problem (9) which is a convex program for any convex loss function, so any local optimal solution obtained by Algorithm $\mathcal{A}$ is also a global minimizer of (4). In our experiments, we implement Algorithm $\mathcal{A}$ with the logistic regression loss and the least squares loss with the 2-norm regularization for P_1 and P_2, and thus the algorithm globally converges.

6 Experiments

We validate the proposed collaborative learning approach by comparing it to standard single-task learning (STL) approaches where multiple tasks are tackled independently, and to two commonly-used multi-task learning (MTL) methods, a regularization-based MTL method in [15] and a Bayesian MTL method based on Gaussian processes (CGP) in [5]. Notice that Algorithm $\mathcal{A}$ with the 1-norm penalty term for both P_1 and P_2 in Problem (3) has been implemented in our early work [12] where a pooling method which trains a single model for all tasks has also been compared. Readers can consult [12] for corresponding results.

6.1 Synthetic Data

We generated synthetic data to verify the behavior of the developed algorithms regarding the selected features and the performance in comparison with single-task learning (STL) logistic regression. The synthetic data was generated as follows:

- generate $\mathbf{x} \in R^{10}$ with each component $x_i \sim \mathbf{Uniform}[-1, 1]$;
- set $T = 3$ and the coefficient vectors of the 3 tasks to

$$\begin{aligned} \alpha_1 &= [1, 1, 1, 0, 0, 0, 0, 0, 0, 0], \\ \alpha_2 &= [0, 1, 1, 1, 0, 0, 0, 0, 0, 0], \\ \alpha_3 &= [0, 0, 1, 1, 1, 0, 0, 0, 0, 0]; \end{aligned}$$

- $y = \text{sign}(\alpha_t^\top \mathbf{x})$ for every sample $\mathbf{x}$ of task t.

For each task, we generated training sets of sizes $\ell = [20, 40, 60, 80]$, each used in a different trial, 150 samples for tuning and 1000 samples for testing, and repeated each trial 20 times. The misclassification rates averaged over the

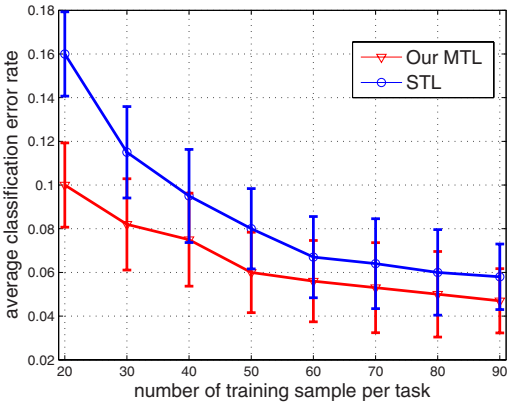


Fig. 3. Performance on synthetic data: error rates versus training sample sizes

3 tasks and 20 runs are shown in Fig. 3 for different training sample sizes, respectively by our approach and STL. Fig. 3 obviously shows the superiority of our approach. As expected, the difference of the two approaches becomes smaller as the sample size of each task becomes larger.

We show bar plots of the averaged estimated coefficient vectors by our approach in Fig. 4-left and the STL logistic regression in Fig. 4-right. Our approach successfully removed irrelevant features. For lucid presentation, each coefficient vector was normalized by its norm, averaged over all trials, and shown on Fig. 4. Although STL produced reasonable classifiers for each task, it could not delete all irrelevant features using data available for each single task.

6.2 Lung Cancer Data

A prototype version of our lungCAD system [16] (not commercially available) was applied on a proprietary anonymized patient data set collected from multiple

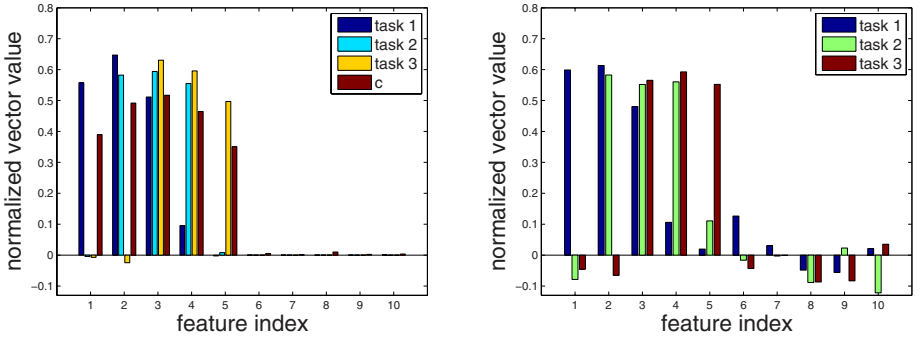


Fig. 4. Performance on synthetic data: left, coefficient vectors by our MTL; right, coefficient vectors by STL logistic regression

hospitals. The nodule dataset consisted of 176 CT images that were randomly partitioned into two groups by a third party agency for development and evaluation respectively: 90 volumes in training and 86 volumes in test. The GGO dataset consisted of 60 CT images. Due to the limited size of GGO set, GGO detection performance could not be measured reliably, so GGO cases were used only for improving nodule detection performance. In total, 129 nodules and 53 GGOs were labeled by radiologists, and 81 nodules appeared in the training set and 48 in the test set. The lungCAD system was independently applied to the training, test nodule sets and the GGO set, generating totally 11056, 13985 and 10265 suspicious candidates in the respective sets. Among them, 131, 81 and 87 candidates were true detections associated with nodules or GGOs. A total of 86 numerical image features were calculated. The statistics of the lungCAD data set is characterized in Table 1.

Table 1. Specifications of lungCAD data sets

	Nodule train	Nodule test	GGO
No. patients	90	86	60
No. candidates	11056	13985	10265
No. cancer	81	48	53
No. positives	131	81	87
No. False Positives /vol	121	161	169
No. feature	86	86	86

The first set of experiments were conducted as follows. We randomly sampled 50% (45 volumes) of the nodule training cases and 50% (30 volumes) of the GGO cases to train a classifier that was tested on 86 test nodule cases, and repeated 15 trials. In the first trial, we tuned the model parameter γ in Algorithm $\mathcal{A}$ and the regularized parameters in [15] according to a 3-fold cross validation performance within training, and we fixed them for other trials. Fig. 5 shows the test ROC curves averaged over the 15 trials with error variance bars. Our algorithm $\mathcal{A}$ produces a curve that dominates the ROC curves corresponding to other approaches. It also had a relatively small model variance by referencing the error bars. The regularized MTL and CGP were superior to STL learning, inferior to our method, and the regularized MTL also presented a relatively large error variance as shown by the error bars.

We conducted more complete performance comparisons using the AUC measure by randomly sampling p% of training nodule cases with a fixed amount of GGO cases where $p = 10,\ 25,\ 50,\ 75,\ 100$, and 15 trials were performed for each p. We averaged the AUC numbers over 15 trials for each value of p, and illustrated them in Fig. 6 together with error bars. The resulting models achieved better performance with less help from related tasks when more samples of the nodule detection task were used.

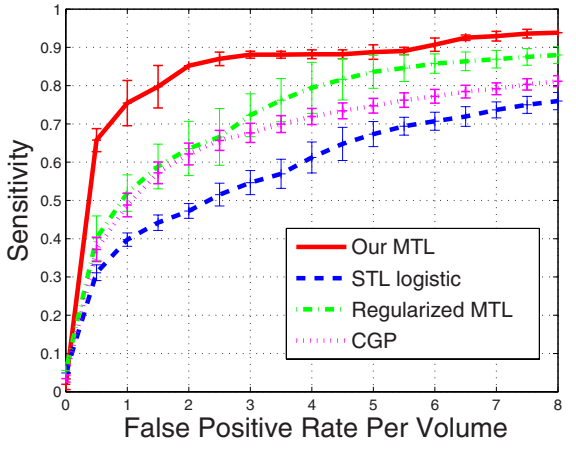


Fig. 5. On Lung Cancer Data: test ROC plots of models trained using 50% of nodule and GGO training patient volumes

6.3 Heart Wall Motion Data

Cardiac wall motion data has a different structure from lung cancer data (in lungCAD data, different patient data were provided for different tasks). Here we collected 220 ultrasound images of patients hearts, and 432 image features were extracted from the left ventricle of each heart to characterize the global motion and segment-level wall motion of the LV. Hence each heart was represented by a feature vector of 432 components. Overall 16 labels, one for each segment, were provided to a single feature vector. Hence the same set of patients were provided

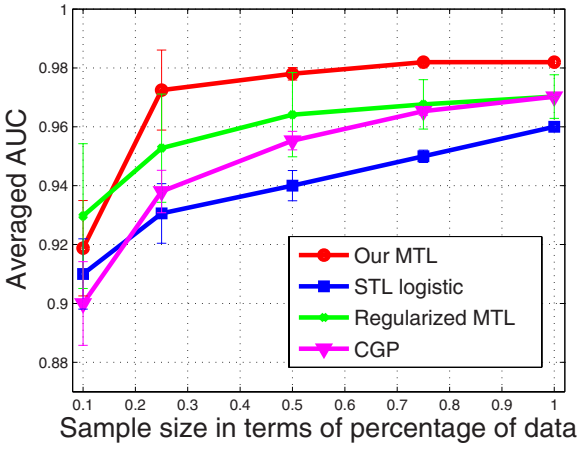


Fig. 6. On Lung Cancer Data: the plot of averaged AUC versus training sample size

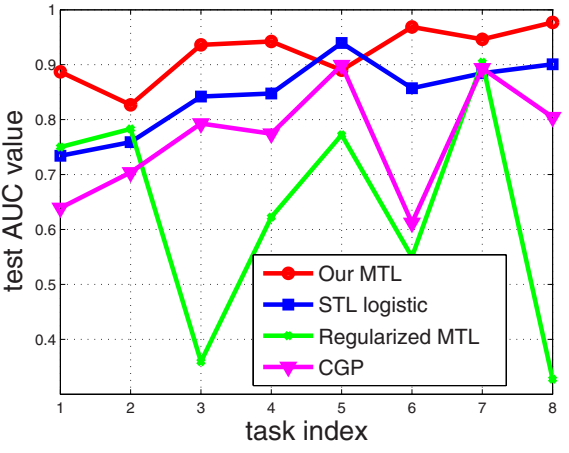


Fig. 7. On Heart Data: the plot of test AUC versus task index

for the different 16 tasks. This is sometimes referred to as multi-label prediction problems.

Although great efforts were made to collect a reasonable number of abnormal studies, the normal versus abnormal segment-level distribution was extremely unbalanced since most patients only have one or two abnormal segments. Many of the segments had fewer than 3 abnormal cases. Only 8 segments (out of 16 segments), for which enough abnormal cases (25 cases on average) were present in the 220 cases, were used in our experiments.

The 220 cases were randomly split 15 times into two sets of an equal size, one for training and one for test. We tuned model parameters such as γ using a 2-fold

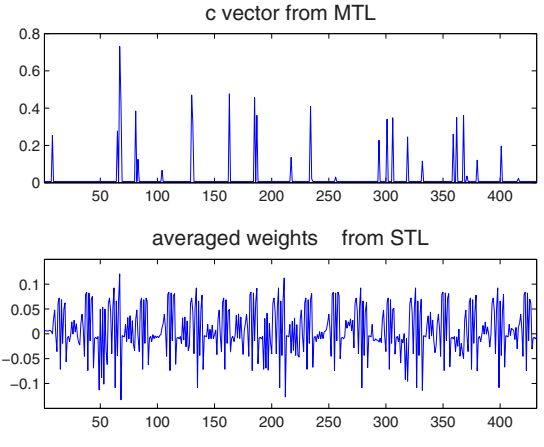


Fig. 8. On Heart Data: comparison of the selected features by our MTL and STL

cross validation within the training set. The average test AUC values for each of the 8 tasks are depicted in Figure 7 which clearly shows the effectiveness of our MTL approach. The regularized MTL and CGP were not originally proposed for sparse estimation, which may result in poor performance on data where dimension is much larger than the available sample size, such as the heart data.

The $\mathbf{c}$ vector learned by our approach was very sparse as shown in Fig. 8 which shows only 21 features were used by all the 8 classifiers combined. Notice that each classifier only chooses features from the features selected by $\mathbf{c}$. Whereas, STL logistic regression used much more features as the averaged weight vector $\boldsymbol{\alpha}$ in Fig. 8 was dense.

7 Conclusions

We have designed a series of approaches to learning multiple tasks jointly. Efficient algorithms have been developed through alternating optimization to find the optimal solutions of these approaches. Convergence analysis shows that the algorithms globally converge for strictly convex loss functions and regularization conditions. Our framework also provides a probabilistic interpretation for existing regularized multi-task learning methods. Although the proposed algorithms are general enough to be applied to any multi-task setting, they are motivated by the challenges of the real-world medical diagnostic problems. Computational results of the proposed approach on medical diagnostic problems demonstrate superiority to some early multi-task learning approaches. The proposed approach has been deployed in our lungCAD system which has received clinical approval from Food and Drug Administration. Possible extension of this work includes the examination of general feature representation without the independence assumption among features and the related algorithm design.

References

1. Roehrig, J.: The promise of CAD in digital mamography. European Journal of Radiology 31, 35–39 (1999)
2. Armato-III, S.G., Giger, M.L., MacMahon, H.: Automated detection of lung nodules in CT scans: preliminary results. Medical Physics 28(8), 1552–1561 (2001)
3. Suzuki, K., Kusumoto, M., Watanabe, S., Tsuchiya, R., Asamura, H.: Radiologic classfication of small adenocarcinoma of the lung: Radiologic-pathologic correlation and its prognostic impact. The Annals of Thoracic Surgery CME Program 81, 413–420 (2006)
4. Caruana, R.: Multitask learning. Machine Learning 28(1), 41–75 (1997)
5. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian processes from multiple tasks. In: ICML 2005 (2005)
6. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research 6, 1855–1887 (2005)
7. Heskes, T.: Empirical Bayes for learning to learn. In: Proc. 17th International Conf. on Machine Learning, pp. 367–374. Morgan Kaufmann, San Francisco (2000)

8. Jebara, T.: Multi-task feature and kernel selection for SVMs. In: Proceedings of the 21st International Conference on Machine learning (2004)
9. Obozinski, G., Taskar, B., Jordan, M.I.: Multi-task feature selection. Technical report, UC Berkeley (2006)
10. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems, vol. 19, pp. 41–48. MIT Press, Cambridge (2007)
11. Srebro, N., Rennie, J.D.M., Jaakola, T.S.: Maximum-margin matrix factorization. In: NIPS 2005 (2005)
12. Xiong, T., Bi, J., Rao, R.B., Cherkassky, V.: Probabilistic joint feature selection for multi-task learning. In: SIAM International Conference on Data Mining (2006)
13. Bezdek, J.C., Hathaway, R.J.: Convergence of alternating optimization. Neural, Parallel Sci. Comput. 11, 351–368 (2003)
14. Bezdek, J.C., Hathaway, R.J.: Some notes on alternating optimization. In: Pal, N.R., Sugeno, M. (eds.) AFSS 2002. LNCS (LNAI), vol. 2275, pp. 288–300. Springer, Heidelberg (2002)
15. Evegniou, T., Pontil, M.: Regularized multi–task learning. In: Proc. of 17–th SIGKDD Conf. on Knowledge Discovery and Data Mining (2004)
16. Rao, R.B., Bi, J., Fung, G., Salganicoff, M., Obuchowski, N., Naidich, D.: LungCAD: a clinically approved, machine learning system for lung cancer detection. In: ACM International Conference on Knowledge Discovery and Data Mining (2007)

Semi-supervised Laplacian Regularization of Kernel Canonical Correlation Analysis

Matthew B. Blaschko, Christoph H. Lampert, and Arthur Gretton

Max Planck Institute for Biological Cybernetics
Department of Empirical Inference
Spemannstr. 38, 72076 Tübingen, Germany
firstname.lastname@tuebingen.mpg.de

Abstract. Kernel canonical correlation analysis (KCCA) is a dimensionality reduction technique for paired data. By finding directions that maximize correlation, KCCA learns representations that are more closely tied to the underlying semantics of the data rather than noise. However, meaningful directions are not only those that have high correlation to another modality, but also those that capture the manifold structure of the data. We propose a method that is simultaneously able to find highly correlated directions that are also located on high variance directions along the data manifold. This is achieved by the use of semi-supervised Laplacian regularization of KCCA. We show experimentally that Laplacian regularized training improves class separation over KCCA with only Tikhonov regularization, while causing no degradation in the correlation between modalities. We propose a model selection criterion based on the Hilbert-Schmidt norm of the semi-supervised Laplacian regularized cross-covariance operator, which we compute in closed form.

1 Introduction

Kernel canonical correlation analysis (KCCA) is a fundamental technique for dimensionality reduction that relies on paired data to learn directions that maximize correlation between the projected representations in each space [1,2]. Techniques based on only one space are susceptible to failure in the event that there are high-variance, semantically meaningless noise directions. KCCA overcomes this weakness by requiring that the projected data be correlated to a projection of the other modality, and has been shown to increase class separability when compared to single modality dimensionality reduction [3]. While KCCA often gives superior results to single modality dimensionality reduction techniques, correlation with some output modality may not be the only criterion of interest. We wish to find directions that not only relate the two modalities, but also lie along the data manifold, in order to better represent the structure of the data and improve class separability. In this work, we describe a method to incorporate these two goals into a common optimization by employing semi-supervised Laplacian regularization. This method gives an embedding of the data that makes use of the information between modalities, as well as the information within each single

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 133–145, 2008.

modality. By using Laplacian regularization, we are able to learn directions that tend to lie along the data manifold estimated from a much larger set of data [4]. This gives us greater confidence that the learned directions represent the underlying statistical structure of the data and that we have not been misled by small sample effects. We show experimentally that learning along the manifold results in increased performance, even in the fully supervised setting, in that the learned embeddings give better class separability on a variety of datasets.

One way to evaluate the performance of KCCA is to take the sum of the squared correlations that it reveals. This quantity turns out to be the Hilbert-Schmidt norm of the normalized covariance operator between the feature representations of each modality, and is referred to as the Hilbert-Schmidt normalized independence criterion [5]. The underlying concept of semi-supervised Laplacian regularization of KCCA can also be applied to an empirical estimate of this operator, and therefore also to the independence criterion. Here, we make use of this Laplacian regularized estimate to define a model selection criterion for the regularization parameters that can be computed in closed form from the kernel matrices and Laplacian.

The rest of the paper is organized as follows. We discuss related work in Section 2 and give a review of KCCA in Section 3. In Section 4 we present the semi-supervised Laplacian regularization of KCCA. In Section 4.3 we discuss the relationship between the proposed algorithm and a recently introduced semi-supervised Fisher linear discriminant analysis algorithm. We describe our model selection criterion in Section 5 and also introduce the semi-supervised Laplacian regularized estimate of the HSNIC. Experimental results are presented in Section 6. Finally, we conclude in Section 7.

2 Related Work

Although KCCA has been applied in many situations, including cross media information retrieval [2,6], multi-modal data clustering [3], analysis of fMRI data [7], extraction of gene clusters [8], testing for independence [9,10], and ICA [11], to our knowledge there have been no semi-supervised extensions of the algorithm. Laplacian regularization is a common technique for semi-supervised learning [4,12]. [13] have recently proposed a semi-supervised Fisher linear discriminant analysis algorithm based on Laplacian regularization, which we show in Section 4.3 to be a special case of the algorithm proposed here.

In our experiments, we will perform model selection by making use of various statistics computed on the correlation operator spectrum (see Section 5): we therefore provide a brief overview of methods used to evaluate and summarize this spectrum. A variety of statistics on the correlation operator spectrum are presented in [9] (for the spline kernel RKHS), where these are used for independence testing. Statistics on the correlation operator used for independent component analysis in [11] include the maximum singular value and kernel generalized variance, where the latter is an upper bound near independence on the mutual information [14]. Finally, a closed form expression for the

Hilbert-Schmidt norm of the correlation operator is provided in [5], where it is shown that this norm is an estimate of the mean squared contingency. Finally, the spectrum of two correlation operators can be compared directly for model selection, as in [2].

3 A Review of Kernel Canonical Correlation Analysis

3.1 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) seeks to utilize paired datasets to simultaneously find projections from each feature space that maximize the correlation between the projected representations [1]. Given a sample from a paired dataset[1] $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ we would like to simultaneously find directions w_x and w_y that maximize the correlation of the projections of x onto w_x with the projections of y onto w_y. This is expressed as

$$\max_{w_x, w_y} \frac{\hat{E}\left[\langle x, w_x\rangle \langle y, w_y\rangle\right]}{\sqrt{\hat{E}\left[\langle x, w_x\rangle^2\right] \hat{E}\left[\langle y, w_y\rangle^2\right]}}, \tag{1}$$

where $\hat{E}$ denotes the empirical expectation. We denote the covariance matrix of (x, y) by C and use the notation C_{xy} (C_{xx}) to denote the cross (auto) covariance matrices between x and y. Equation (1) is equivalent to

$$\max_{w_x, w_y} \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x \;\; w_y^T C_{yy} w_y}}. \tag{2}$$

This Rayleigh quotient can be optimized as a generalized eigenvalue problem, or by decomposing the problem using the Schur complement as described in [2].

There is a natural extension of CCA in the event where there are more than two modalities. This can be written as a generalized eigenvector problem that subsumes two-way CCA as a special case

$$\begin{pmatrix} C_{11} & \ldots & C_{1k} \\ \vdots & \ddots & \vdots \\ C_{k1} & \ldots & C_{kk} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} = \lambda \begin{pmatrix} C_{11} & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & C_{kk} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix}. \tag{3}$$

3.2 Kernel Canonical Correlation Analysis

We can extend CCA, *e.g.* to non-vectorial domains by defining kernels over x and y: $k_x(x_i, x_j) = \langle \phi_x(x_i), \phi_x(x_j)\rangle$ and $k_y(y_i, y_j) = \langle \phi_y(y_i), \phi_y(y_j)\rangle$, and searching

[1] We assume the samples have zero mean for notational convenience.

for solutions that lie in the span of $\phi_x(x)$ and $\phi_y(y)$: $w_x = \sum_i \alpha_i \phi_x(x_i)$ and $w_y = \sum_i \beta_i \phi_y(y_i)$. In this setting we use an empirical estimator for C:

$$\hat{C}_{xy} = \frac{1}{n} \sum_{i=1}^{n} \phi_x(x_i) \cdot \phi_y(y_i)^T, \tag{4}$$

where n is the sample size, and $\phi_x(x_i)$ and $\phi_y(y_i)$ are assumed to have 0 mean. $\hat{C}_{xx}$ and $\hat{C}_{yy}$ are defined similarly. Denoting the kernel matrices defined by our sample as K_x and K_y, the solution of Equation (2) is equivalent to maximizing the following with respect to coefficient vectors, α and β

$$\frac{\alpha^T \frac{1}{n} K_x K_y \beta}{\sqrt{\alpha^T \frac{1}{n} K_x^2 \alpha \beta^T \frac{1}{n} K_y^2 \beta}} = \frac{\alpha^T K_x K_y \beta}{\sqrt{\alpha^T K_x^2 \alpha \beta^T K_y^2 \beta}}. \tag{5}$$

As discussed in [2] this optimization leads to degenerate solutions in the case that either K_x or K_y is invertible so we maximize the following regularized expression

$$\frac{\alpha^T K_x K_y \beta}{\sqrt{\alpha^T \left(K_x^2 + \varepsilon_x K_x\right) \alpha \beta^T \left(K_y^2 + \varepsilon_y K_y\right) \beta}}, \tag{6}$$

which is equivalent to Tikhonov regularization of the norms of w_x and w_y in the denominator of Equation (2). In the limit case that $\varepsilon_x \to \infty$ and $\varepsilon_y \to \infty$, the algorithm maximizes covariance instead of correlation.

The formulation of CCA in Equation (3) is also readily regularized and kernelized, and allows one to take advantage of more than two modalities at a time.

4 Semi-supervised Kernel Canonical Correlation Analysis

If we have additional data available that do not have correspondences to the other modality, we can search for solutions that lie in the span of the larger set of training points, and regularize using the additional data. We propose Laplacian regularization, which tends to find solutions that lie along an empirical estimate of the data manifold [4]. This gives increased robustness to the algorithm, and increases class separability in the absence of label information.

4.1 The Two-Modality Case

We have training data $\{x_1, \ldots, x_n\}$ with corresponding data $\{y_1, \ldots, y_n\}$ as well as additional training data $\{x_{n+1}, \ldots, x_{n+p_x}\}$ and $\{y_{n+1}, \ldots, y_{n+p_y}\}$ that do not have correspondences. We use the variables $m_x = n + p_x$ ($m_y = n + p_y$) to denote the total number of samples in modality x (y). We denote the $d \times n$ data matrix $X = (x_1, \ldots, x_n)$, and the matrix including all data with and without correspondences $\hat{X} = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+p_x})$, and similarly for Y

and $\hat{Y}$. Furthermore we denote kernel matrices between the various sets of data as follows: $\Phi_x(X)^T\Phi_x(X) = K_{xx}$, $\Phi_x(\hat{X})^T\Phi_x(X) = K_{\hat{x}x}$, $\Phi_x(\hat{X})^T\Phi_x(\hat{X}) = K_{\hat{x}\hat{x}}$, *etc.*. Kernel matrices for Y are defined analogously. We wish to optimize the following generalization of Equation (6)

$$\frac{\alpha^T K_{\hat{x}x} K_{y\hat{y}} \beta}{\sqrt{\alpha^T \left(K_{\hat{x}x} K_{x\hat{x}} + R_{\hat{x}}\right) \alpha \beta^T \left(K_{\hat{y}y} K_{y\hat{y}} + R_{\hat{y}}\right) \beta}}, \tag{7}$$

where $R_{\hat{x}} = \varepsilon_x K_{\hat{x}\hat{x}} + \frac{\gamma_x}{m_x^2} K_{\hat{x}\hat{x}} \mathcal{L}_{\hat{x}} K_{\hat{x}\hat{x}}$ and $\mathcal{L}_{\hat{x}}$ is the empirical graph Laplacian estimated from the m_x samples of labeled and unlabeled data.

4.2 The General Case

In the general case, we have more than two modalities. As a result, the data that has correspondences between modalities 1 and 2 can be different than the data that has correspondences between modalities 2 and 3, *etc.*. We abuse the notation $K_{\hat{ij}}$ to denote the kernel matrix computed between all the data for modality i and the data for modality i that also has correspondences to the data in modality j. This matrix has dimensionality $m_i \times n_{ij}$, where m_i is the total number of training examples (with or without correspondences) for modality i, and n_{ij} is the number of correspondences between modalities i and j.

The following generalizes Equations (3) and (7)

$$\begin{pmatrix} \mathbf{0} & \dots & \frac{1}{n_{1k}} K_{\hat{1}k} K_{1\hat{k}} \\ \vdots & \ddots & \vdots \\ \frac{1}{n_{1k}} K_{\hat{k}1} K_{k\hat{1}} & \dots & \mathbf{0} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix} = \tag{8}$$
$$\lambda \begin{pmatrix} \frac{1}{m_1} K_{\hat{1}1} K_{1\hat{1}} + R_{\hat{1}} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \frac{1}{m_k} K_{\hat{k}k} K_{k\hat{k}} + R_{\hat{k}} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}.$$

4.3 Fisher Linear Discriminant Analysis

There is an intimate relationship between CCA and Fisher linear discriminant analysis (LDA) [15]. LDA is a special case of CCA where the second modality is the labels [16,17], consequently, any semi-supervised algorithm for CCA implies a semi-supervised LDA algorithm as well. Recently [13] have proposed a semi-supervised LDA approach. If we use the identity kernel on the labels, set the label regularization parameters to 0, and set $\varepsilon_x = 0$, the directions learned from Equation (7) are the same as those found using the method of [13].

Similarly, when one of the spaces is one-dimensional (*i.e.* the kernel matrix is rank 1), Laplacian regularized KCCA gives a generalization of kernel ridge regression.

5 Model Selection

We propose a model selection algorithm based on the Hilbert-Schmidt normalized information criterion (HSNIC). The HSNIC is closely related to KCCA and is equivalent to the squared ℓ_2 norm of the spectrum of the normalized cross-covariance operator, which in the limit is independent of the kernel[2] used in its estimation [5,10]. Because the spectrum of the normalized cross-covariance operator is identical to the spectrum of the solutions to KCCA, this provides us with a useful statistic upon which we can base our model selection. HSNIC gives us access to the ℓ_2 norm of the spectrum, which is dominated by the first KCCA directions for kernels with quickly decaying spectra, such as the Gaussian kernel [11,18].

We first derive the semi-supervised empirical HSNIC estimate in Section 5.1 and then use this result to define our model selection criterion in Section 5.2.

5.1 Semi-supervised Empirical HSNIC Estimate

The HSNIC is the Hilbert-Schmidt norm of the normalized cross-covariance operator, V_{xy}, which we define to be regularized using the Laplace-Beltrami operators on the manifolds of the data [4], $\Delta_{\mathcal{M}_x}$ and $\Delta_{\mathcal{M}_y}$,

$$V_{xy} = \left(\Sigma_{xx} + \varepsilon_x I + \gamma_x \Delta_{\mathcal{M}_x}\right)^{-\frac{1}{2}} \Sigma_{xy} \left(\Sigma_{yy} + \varepsilon_y I + \gamma_y \Delta_{\mathcal{M}_y}\right)^{-\frac{1}{2}}. \tag{9}$$

We estimate the normalized cross-covariance operator empirically using a finite sample of data, yielding

$$\hat{V}_{xy} = \left(\frac{1}{n} X X^T + \varepsilon_x I + \frac{\gamma_x}{m_x^2} \hat{X} \mathcal{L}_{\hat{x}} \hat{X}^T\right)^{-\frac{1}{2}} \frac{1}{n} X Y^T \cdot \left(\frac{1}{n} Y Y^T + \varepsilon_y I + \frac{\gamma_y}{m_y^2} \hat{Y} \mathcal{L}_{\hat{y}} \hat{Y}^T\right)^{-\frac{1}{2}}. \tag{10}$$

The semi-supervised Laplacian regularized empirical estimate of the HSNIC is therefore

$$\|\hat{V}_{xy}\|_{HS}^2 = \operatorname{Tr}\left[\hat{V}_{xy} \hat{V}_{xy}^T\right] = \operatorname{Tr}\left[M_x M_y\right], \tag{11}$$

where

$$M_x = I - n\left(nI + \frac{1}{\varepsilon_x} K_{xx} - \frac{1}{\varepsilon_x} K_{x\hat{x}} \left(\frac{m_x^2 \varepsilon_x}{\gamma_x} I + \mathcal{L}_{\hat{x}} K_{\hat{x}\hat{x}}\right)^{-1} \mathcal{L}_{\hat{x}} K_{\hat{x}x}\right)^{-1}, \tag{12}$$

and M_y is defined analogously. See Section A for the derivation.

[2] Assuming that the kernel comes from the class of *characteristic* kernels, as defined in [10]. A Gaussian kernel is sufficient.

5.2 HSNIC Model Selection Criterion

HSNIC is an interesting model selection criterion for many problems as it provides an estimate of the dependence between X and Y [10]. As discussed earlier, KCCA in high dimensional feature spaces requires regularization to return non-trivial projection directions: in the event that all regularization is set to 0, HSNIC estimates perfect correlation if the kernel matrices, K_{xx} and K_{yy}, are invertible. Since choosing the parameters that maximize HSNIC risks overfitting, it is more meaningful to consider the amount by which the dependence witnessed by HSNIC increases over its value at independence (i.e., in the absence of correlations between X and Y). We can simulate the latter quantity by randomly permuting the labels relative to the data: if we were to average several such permutations, we would obtain an estimate of HSNIC at independence. The averaging procedure is computationally expensive, however: thus, we use a single data permutation to approximate the HSNIC value at independence. We observed on our data that the values of HSNIC for different permutations were highly concentrated about their mean, which makes this a reasonable approximation. The model selection criterion consists of the ratio between the non-permuted and the permuted HSNIC values. If this ratio is high, we are confident that the correlation found is genuine and is not a result of overfitting. The HSNIC estimate for the permuted dataset is easily computed using a random permutation matrix, P,

$$\|\hat{V}_{xy_R}\|^2_{HS} = \mathrm{Tr}\left[M_x P^T M_y P\right], \tag{13}$$

where $\hat{V}_{xy_R}$ is the empirical estimate computed using $Y_R = YP$ in place of Y in Equation (10). This can be verified with an analogous derivation to that in Section A. We denote the model selection criterion

$$\rho(\varepsilon_x, \gamma_x, \varepsilon_y, \gamma_y) = \frac{\|\hat{V}_{xy}\|^2_{HS}}{\|\hat{V}_{xy_R}\|^2_{HS}}, \tag{14}$$

and maximize with respect to its parameters. The cost of computing Equation (14) is only marginally higher than computing Equation (11) as we can reuse the computation of M_x and M_y in the permuted version.

6 Experimental Results

6.1 Data

We have performed experiments on a number of datasets of images with associated text. We have used the three datasets included in the UIUC-ISD collection [19]. These consist of images collected from search engines using ambiguous search terms, "bass," "crane," and "squash," the webpages in which the images originally appeared, and an annotation of which sense of the word the image represents, *e.g.* fish vs. musical instrument. There are 2881 images in the Bass dataset which have been grouped into 6 categories, 2650 in the Crane dataset

grouped into 9 categories, and 1948 images in the Squash dataset grouped into 6 categories. For all three datasets, we extracted 128 dimensional SURF descriptors without rotation invariance and with the keypoint threshold set to 0 [20] and constructed a codebook of 1000 visual words using k-means with 50000 sampled descriptors. Images were represented by a normalized histogram of these visual words. For the text representation, we used term frequency histograms extracted from the webpage title, removing special characters and stop words using the list from [21]. Both image and text similarities were computed using a χ^2 kernel,

$$k(x, x') = e^{-\frac{1}{2A} \sum_{i=1}^{d} \frac{(x_i - x'_i)^2}{x_i + x'_i}}, \tag{15}$$

with normalization parameter A set to the median of the χ^2 distances in the training set.

Additionally, we have used the multimedia image-text web database used in [2,22] which consists of samples from three classes: sports, aviation, and paintball, with 400 image-text pairs each. Images were represented using HSV color and Gabor textures as in [2,22]. Text was represented using term frequencies. As in [2] we have used a Gaussian kernel for the image space, and a linear kernel for text.

6.2 Evaluation Methodology

To evaluate the performance of the algorithm, the following evaluation is performed. We randomly split the data into equally sized train and test portions. The train portion is further split into data with and without correspondences between the different modalities. Semi-supervised Laplacian regularized KCCA is trained using data with and without correspondences, using parameters learned with grid search on the objective described in Section 5.2. Test data are embedded using the learned parameters, and correlations are computed between the embeddings of the two modalities. We repeat this procedure 40 times, and evaluate the performance using two metrics: the mean ℓ_2 norm of the cross-correlation coefficients, to determine how well the projected data are correlated; and the ratio of the determinant of the total scatter matrix and the determinant of sum of within class scatter matrices for each modality to determine how well the within-class variation along the data manifold is captured:

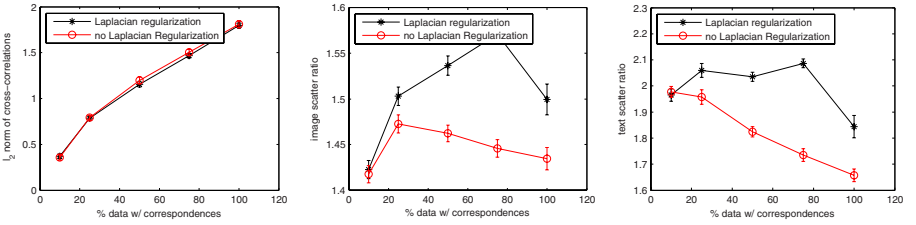
$$\frac{|S_t|}{|\sum_{i=1}^{c} S_{c_i}|}, \tag{16}$$

where μ denotes the mean of the embedded test data, c_i denotes the test data that are in class i, μ_i denotes the mean of class i,

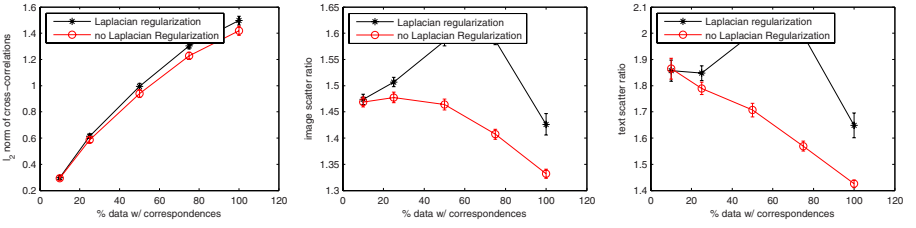
$$S_t = \sum_{j \in test} (x_j - \mu)(x_j - \mu)^T, \tag{17}$$

and

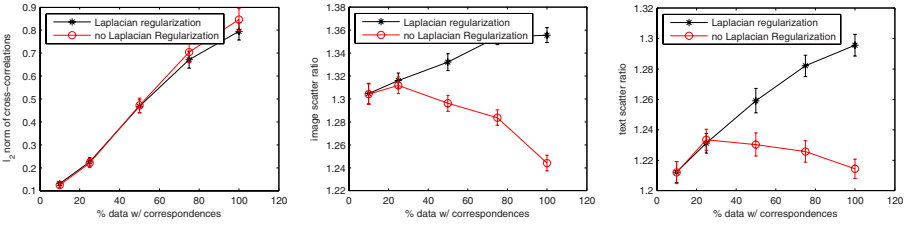
$$S_{c_i} = \sum_{j \in c_i} (x_j - \mu_i)(x_j - \mu_i)^T. \tag{18}$$



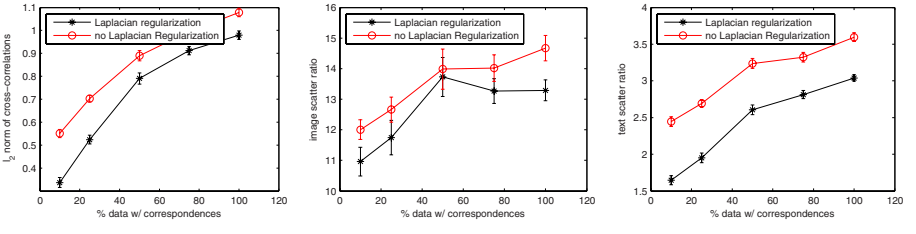
(a) Bass



(b) Crane



(c) Squash



(d) Sports, Aviation, Paintball

Fig. 1. Experimental results for four different datasets. The first column is the ℓ_2 norm of the cross correlations between the modalities of the held out data. The second and third columns are the scatter ratios for images and text, respectively.

Although class labels are available for the dataset, we only use them at test time during this evaluation in order to measure the separation of semantic classes achieved by the embeddings. We compute the embeddings without the semi-supervised Laplacian regularization and also visualize the norm of the resulting correlation coefficients, and scatter ratios.

In all experiments except for the "Sports Aviation Paintball" dataset, we have defined the Laplacian using the similarity matrix, W, defined by the kernel described in Section 6.1. In the "Sports Aviation Paintball" dataset, we use a linear kernel on the text modality for consistency with previous publications [2,3]. Insted, we have used a Gaussian kernel to compute the Laplacian matrix for the text modality. In all cases, we use the symmetric normalized Laplacian, $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$, where D is the diagonal matrix whose entries are the row sums of W.

6.3 Results

Figure 1 gives results for the four datasets described in Section 6.1. The plots have been computed by varying the percentage of training data for which correspondences between images and text have been provided to the algorithms. For more pairs of data, we see that correlations are better represented for KCCA with and without Laplacian regularization, as expected. The advantage of Laplacian regularization is shown by improved class separability (as measured by scatter ratios) for three of the four datasets. This indicates that the manifold structure of these datasets is important for class separability, and that this is captured without sacrificing performance on correlation.

Laplacian regularization slightly decreases cross-correlation and scatter ratios in the "Sport, Aviation, Paintball" dataset (first column). This can be described in part by the relatively simple structure of the "Sport, Aviation, Paintball" dataset. PCA, kernel-PCA, and KCCA all give similar embeddings for this dataset, with the majority of variance contained in only two dimensions [3]. The use of Laplacian regularization is therefore unnecessary as there is little non-linearity in the data manifold; manifold structure is effectively captured by linear high variance directions and non-parametric Laplacian regularization degrades performance.

For the "Bass," "Crane," and "Squash" datasets, Laplacian regularization is able to capture relevant discriminative structure that is available in each modality without sacrificing performance in finding directions that show correlation between the modalities.

7 Conclusions and Future Work

In this work we have proposed the use of Laplacian regularized kernel canonical correlation analysis as a dimensionality reduction technique. Experimental results show increased performance in class separation for datasets that have sufficient nonlinear structure. We have proposed a model selection criterion based

on the Hilbert-Schmidt norm of the Laplacian regularized normalized cross covariance operator and have derived its solution in closed form (Equation (11)).

The Hilbert-Schmidt normalized information criterion is an important statistical object that can be used to test for independence of sets of variables, which gives rise to many applications in machine learning. A promising area for future work is to experimentally validate the benefit of using the Laplacian regularized empirical estimate in applications where only Tikhonov regularization has been previously applied. Examples include causality inference [10] and ICA [11].

All experiments here have been performed using only two modalities. Laplacian regularization of KCCA for multiple modalities, as described in Equation (8) warrants further experimental evaluation. This is particularly relevant, *e.g.*, in multi-language text corpora for which correspondences for some but not all documents are known. Laplacian regularization would allow better modeling of the characteristics of each of the individual languages.

Finally, depending on the structure of a dataset, iterated Laplacian regularization may be appropriate in some cases [23]. This gives stronger conditions on the structure of the manifold which may help in avoiding overfitting.

Acknowledgments

The first author is supported by a Marie Curie fellowship under the EU funded project PerAct, EST 504321. This work is funded in part by the CLASS project, IST 027978.

References

1. Hotelling, H.: Relations Between Two Sets of Variates. Biometrika 28, 321–377 (1936)
2. Hardoon, D.R., Szedmák, S., Shawe-Taylor, J.R.: Canonical Correlation Analysis: An Overview with Application to Learning Methods. Neural Computation 16, 2639–2664 (2004)
3. Blaschko, M.B., Lampert, C.H.: Correlational Spectral Clustering. In: CVPR (2008)
4. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. JMLR 7, 2399–2434 (2006)
5. Fukumizu, K., Bach, F.R., Gretton, A.: Statistical Consistency of Kernel Canonical Correlation Analysis. JMLR 8, 361–383 (2007)
6. Li, Y., Shawe-Taylor, J.: Using kcca for japanese—english cross-language information retrieval and document classification. J. Intell. Inf. Syst. 27, 117–133 (2006)
7. Hardoon, D.R., Mourão-Miranda, J., Brammer, M., Shawe-Taylor, J.: Unsupervised Analysis of fMRI Data Using Kernel Canonical Correlation. NeuroImage 37, 1250–1259 (2007)
8. Yamanishi, Y., Vert, J.P., Nakaya, A., Kanehisa, M.: Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. Bioinformatics 19, i323–330 (2003)

9. Dauxois, J., Nkiet, G.M.: Nonlinear canonical analysis and independence tests. Ann. Statist. 26, 1254–1278 (1998)
10. Fukumizu, K., Gretton, A., Sun, X., Schölkopf, B.: Kernel Measures of Conditional Dependence. In: NIPS (2007)
11. Bach, F.R., Jordan, M.I.: Kernel Independent Component Analysis. JMLR 3, 1–48 (2002)
12. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
13. Cai, D., He, X., Han, J.: Semi-supervised discriminant analysis. In: ICCV (2007)
14. Gretton, A., Herbrich, R., Smola, A., Bousquet, O., Schölkopf, B.: Kernel methods for measuring independence. J. Mach. Learn. Res. 6, 2075–2129 (2005)
15. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 7, 179–188 (1936)
16. De Bie, T.: Semi-supervised learning based on kernel methods and graph cut algorithms. Phd thesis, K.U.Leuven (Leuven, Belgium), Faculty of Engineering (2005)
17. Bach, F.R., Jordan, M.I.: A Probabilistic Interpretation of Canonical Correlation Analysis. Technical Report 688, Department of Statistics, University of California, Berkeley (2005)
18. Braun, M.L.: Accurate error bounds for the eigenvalues of the kernel matrix. JMLR 7, 2303–2328 (2006)
19. Loeff, N., Alm, C.O., Forsyth, D.A.: Discriminating Image Senses by Clustering with Multimodal Features. In: ACL (2006)
20. Bay, H., Tuytelaars, T., Gool, L.J.V.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
21. van Rijsbergen, C.J.: Information Retrieval. Butterworths (1975)
22. Kolenda, T., Hansen, L.K., Larsen, J., Winther, O.: Independent Component Analysis for Understanding Multimedia Content. In: IEEE Workshop on Neural Networks for Signal Processing, pp. 757–766 (2002)
23. Zhou, D., Schölkopf, B.: Discrete regularization. In: Chapelle, O., Schölkopf, B., Zien, A. (eds.) Semi-supervised learning. Adaptive computation and machine learning, pp. 221–232. MIT Press, Cambridge (2006)

A Derivation of Semi-supervised Empirical HSNIC Estimate

The measure of interest is the Hilbert-Schmidt norm of the semi-supervised empirical estimate of the normalized cross-covariance operator

$$\|\hat{V}_{xy}\|_{HS}^2 = \operatorname{Tr}\left[\hat{V}_{xy}\cdot\hat{V}_{xy}^T\right] \tag{19}$$

$$= \frac{1}{n^2}\operatorname{Tr}\left[X^T\left(\frac{1}{n}XX^T+\varepsilon_x I+\frac{\gamma_x}{m_x^2}\hat{X}\mathcal{L}_{\hat{x}}\hat{X}^T\right)^{-1}X\right.$$
$$\left.Y^T\left(\frac{1}{n}YY^T+\varepsilon_y I+\frac{\gamma_y}{m_y^2}\hat{Y}\mathcal{L}_{\hat{y}}\hat{Y}^T\right)^{-1}Y\right]. \tag{20}$$

Using the Woodbury matrix identity, $(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$, we substitute $A = \varepsilon_x I + \frac{1}{n}XX^T$, $B = \hat{X}$, $C = \frac{\gamma_x}{m_x^2}\mathcal{L}_{\hat{x}}$, and $D = \hat{X}^T$. The following holds

$$\begin{aligned}
&\left(\frac{1}{n}XX^T + \varepsilon_x I + \frac{\gamma_x}{m_x^2}\hat{X}\mathcal{L}_{\hat{x}}\hat{X}^T\right)^{-1} = \\
&\left(\varepsilon_x I + \frac{1}{n}XX^T\right)^{-1} - \left(\varepsilon_x I + \frac{1}{n}XX^T\right)^{-1}\hat{X}\cdot \\
&\left(\frac{m_x^2}{\gamma_x}\mathcal{L}_{\hat{x}}^{-1} + \hat{X}^T\left(\varepsilon_x I + \frac{1}{n}XX^T\right)^{-1}\hat{X}\right)^{-1}\cdot \\
&\hat{X}^T\left(\varepsilon_x I + \frac{1}{n}XX^T\right)^{-1}. \qquad (21)
\end{aligned}$$

We apply the same identity again with the substitution $A = \varepsilon_x I$, $B = X$, $C = \frac{1}{n}I$, and $D = X^T$ to achieve the result

$$\left(\varepsilon_x I + \frac{1}{n}XX^T\right)^{-1} = \frac{1}{\varepsilon_x}I - \frac{1}{\varepsilon_x^2}X\left(nI + \frac{1}{\varepsilon_x}X^TX\right)^{-1}. \qquad (22)$$

Plugging in the results of Equations (21) and (22), along with the analogous term for Y, into Equation (20), we achieve the result

$$\|\hat{V}_{xy}\|_{HS}^2 = \mathrm{Tr}\left[M_x M_y\right], \qquad (23)$$

where

$$\begin{aligned}
M_x = \frac{1}{n}\Bigg(&\frac{1}{\varepsilon_x}K_{xx} - \frac{1}{\varepsilon_x^2}K_{xx}\left(nI + \frac{1}{\varepsilon_x}K_{xx}\right)^{-1}K_{xx} - \\
&\left(\frac{1}{\varepsilon_x}K_{x\hat{x}} - \frac{1}{\varepsilon_x^2}K_{xx}\left(nI + \frac{1}{\varepsilon_x}K_{xx}\right)^{-1}K_{x\hat{x}}\right)\cdot \\
&\Bigg(\frac{m_x^2}{\gamma_x}\mathcal{L}_{\hat{x}}^{-1} + \frac{1}{\varepsilon_x}K_{\hat{x}\hat{x}} - \\
&\qquad\frac{1}{\varepsilon_x^2}K_{\hat{x}x}\left(nI + \frac{1}{\varepsilon_x}K_{xx}\right)^{-1}K_{x\hat{x}}\Bigg)^{-1}\cdot \\
&\left(\frac{1}{\varepsilon_x}K_{\hat{x}x} - \frac{1}{\varepsilon_x^2}K_{\hat{x}x}\left(nI + \frac{1}{\varepsilon_x}K_{xx}\right)^{-1}K_{xx}\right)\Bigg), \qquad (24)
\end{aligned}$$

and M_y is defined analogously. We can further simplify this expression by applying the Woodbury matrix identity in reverse twice, which results in

$$M_x = I - n\left(nI + \frac{1}{\varepsilon_x}K_{xx} - \frac{1}{\varepsilon_x}K_{x\hat{x}}\left(\frac{m_x^2\varepsilon_x}{\gamma_x}I + \mathcal{L}_{\hat{x}}K_{\hat{x}\hat{x}}\right)^{-1}\mathcal{L}_{\hat{x}}K_{\hat{x}x}\right)^{-1}. \qquad (25)$$

Sequence Labelling SVMs Trained in One Pass

Antoine Bordes[1,2], Nicolas Usunier[1], and Léon Bottou[2]

[1] LIP6, Université Paris 6, 104 Avenue du Pdt Kennedy, 75016 Paris, France
[2] NEC Laboratories America, Inc., 4 Independence Way, Princeton, NJ08540, USA
{bordes,usunier}@poleia.lip6.fr, leonb@nec-labs.com

Abstract. This paper proposes an online solver of the dual formulation of support vector machines for structured output spaces. We apply it to sequence labelling using the exact and greedy inference schemes. In both cases, the per-sequence training time is the same as a perceptron based on the same inference procedure, up to a small multiplicative constant. Comparing the two inference schemes, the greedy version is much faster. It is also amenable to higher order Markov assumptions and performs similarly on test. In comparison to existing algorithms, both versions match the accuracies of batch solvers that use exact inference after a single pass over the training examples.

1 Introduction

The sequence labelling task consists in predicting a sequence of *labels* $(y^1 \dots y^T)$ given an observed sequence of *tokens* $(x^1 \dots x^T)$. This task is an example of a *structured output* learning system (see e.g. [1]). It appears in practical problems in computational linguistics and signal processing.

Two informal assumptions are crucial for this task. The first states that a label y^t depends only on the surrounding labels and tokens. The second states that this dependency is invariant with t. These assumptions are expressed through the parametric formulation of the models, and, in the case of probabilistic models, through conditional independence assumptions (e.g. Markov models). Part of the model specification is then the *inference procedure* that recovers the predicted labels for any input sequence. *Exact inference* can be carried out with the Viterbi algorithm. The more efficient *greedy inference*, which predicts the labels in the order of the sequence using the past predictions, can also be competitive in terms of accuracy by considering higher order Markov assumptions. The parameters for both inference schemes can be learned using structured output learning algorithms.

Batch sequence algorithms optimize a global objective function that depends on all training sequences or tokens [2,3,4,5,6]. They mainly consist of an iterative procedure that run several times over the entire dataset until some convergence criterion is met. The number of epochs of these algorithms usually increases with the number of examples, leading to training times that grow faster than the size of the training set.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 146–161, 2008.

A crucial issue with these algorithms is their *scalability*. When learning the parameters for exact inference, support vector methods (e.g. [3]) require the application of the costly Viterbi algorithm each time a sequence is visited in the iterative process. Output space factorization methods (e.g. [5]) solve an alternative problem with additional variables that encode the structure of the possible predicted sequences. In these methods, each sequence adds a number of such variables that is polynomial in the length of the sequence. Learning for greedy inference reduces to a smaller multiclass classification problem and is therefore much faster. However algorithms then focus on tokens rather than sequences, dramatically increasing the size of the training set. Batch sequence learning algorithms, having a computational cost that grows more than linearly with the number of sequences, are impracticable on large datasets because of the high per-sequence training cost.

Online sequence learning algorithms have been proposed as a scalable alternative to batch algorithms. They run a single pass on the training set, sequentially updating their parameters depending on the loss observed after each sequence (e.g. [7]) or token (e.g. [8]). Their computational cost therefore depends linearly on the number of observations.

Proponents of such algorithms often mention that generalization bounds for online algorithms are no worse than generalization bounds for batch algorithms [9], or that specific algorithms like the second order stochastic gradient descent (SOSGD) provably loose nothing relatively to the batch optimization of the same cost [10]. However, the error bounds are not tight, such theoretical guarantees are thus not very informative, and SOSGD algorithms requires an impractically large inverse Hessian matrix. In practice, it appears that online algorithms are still significantly less accurate than batch algorithms.[1]

In this paper, we propose an online algorithm for the optimization of the dual formulation of support vector methods for structured output spaces [2,3]. Following recent works on the fast optimization of Support Vector Machines [6,11], the algorithm performs SMO-like optimization steps over pairs of dual variables, alternating between unseen patterns and currently support patterns. It can be seen as an adaptation of LaRank ([6]), originally proposed for the batch optimization of multiclass SVMs, to online structured output learning.

The algorithm we propose shares the scalability property of other online algorithms, its training time increasing linearly with the number of examples. Similarly to [3,6], its number of support vectors is conveniently bounded. Finally, using an extension of [12] to structured outputs, we show that our algorithm has at least the same theoretical guarantees in terms of regret (difference between the online error and the optimal train error) as *passive-aggressive* online algorithms. Its only drawback is to keep in memory the current vector expansion. However this memory usage grows at most linearly with the number of examples and does not impact the computational cost. This drawback is shared by other online algorithms such as kernel or averaged perceptrons.

[1] A common workaround consists in performing several passes over the training examples. But this is no longer an online algorithm and it no longer enjoys the theoretical guarantees of online algorithms.

We present an empirical evaluation of our algorithm on standard benchmarks for sequence labelling. We test both exact and greedy inference. The performances are very close to state-of-the-art batch optimizers of the same dual, while being an order of magnitude faster. The new algorithm is then only a constant time slower than perceptron-like algorithms using the same inference scheme, while being significantly better in terms of accuracies. We therefore obtain new kinds of compromises in terms of training time/test performance. For example, the greedy version of our algorithm is approximately as fast as an online perceptron using exact inference, while being almost as accurate as a batch optimizer.

2 Representation and Inference

In the rest of this paper, we use bold characters for sequences such as the sequence of tokens $\mathbf{x} = (x^1 \dots x^T)$ or the sequence of labels $\mathbf{y} = (y^1 \dots y^T)$. Subsequences are denoted using superscripts, as in $\mathbf{y}^{\{t-k..t-1\}} = (y^{t-k} \dots y^{t-1})$. We call $\mathcal{X}$ the set of possible tokens and $\mathcal{Y}$ the set of possible labels, augmented with a special symbol to represent the absence of a label. By convention, a label y^s is the special symbol whenever $s \leq 0$. Angle brackets $\langle .,. \rangle$ are used to represent the canonical dot product.

An *inference procedure* assigns a label y^t to each corresponding x^t taking into account the correlations between labels at different positions in the sequence. This work takes into account correlations between $k+1$ successive labels (Markov assumption of order k). More specifically, we assume that the inference procedure determines the predicted label sequence $\mathbf{y}$ on the sole basis of the scores

$$s^t(w, \mathbf{x}, \mathbf{y}) = \left\langle w, \Phi_{\mathrm{g}} \left(x^t, \mathbf{y}^{\{t-k..t-1\}}, y^t \right) \right\rangle \quad t = 1...T,$$

where $w \in \mathbb{R}^D$ is a parameter vector and function $\Phi_{\mathrm{g}} : \mathcal{X} \times \mathcal{Y}^k \times \mathcal{Y} \to \mathbb{R}^D$ determines the feature space.

2.1 Exact Inference

Exact inference maximizes the sum $\sum_{t=1}^{T} s^t(w, \mathbf{x}, \mathbf{y})$ over all possible label sequences $\mathbf{y}$. For a given input sequence $\mathbf{x}$, the prediction function $\mathbf{f}_{\mathrm{e}}(w, \mathbf{x})$ is then defined by

$$\begin{aligned} \mathbf{f}_{\mathrm{e}}(w, \mathbf{x}) &= \arg\max_{\mathbf{y} \in \mathcal{Y}^T} \sum_{t=1}^{T} s^t(w, \mathbf{x}, \mathbf{y}) \\ &= \arg\max_{\mathbf{y} \in \mathcal{Y}^T} \langle w, \Phi_{\mathrm{e}}(\mathbf{x}, \mathbf{y}) \rangle \ , \end{aligned} \tag{1}$$

where $\Phi_{\mathrm{e}}(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \Phi_{\mathrm{g}}(x^t, \mathbf{y}^{\{t-k..t-1\}}, y^t)$.

2.2 Greedy Inference

Greedy inference predicts the successive labels y^t in sequence by maximizing $\langle w, \Phi_{\mathrm{g}}(x^t, \mathbf{y}^{\{t-k..t-1\}}, y^t)\rangle$ where the previously predicted labels $\mathbf{y}^{\{t-k..t-1\}}$ are frozen. For a given input sequence $\mathbf{x}$, the prediction function $\mathbf{f}_{\mathrm{g}}(w, \mathbf{x})$ is then defined by the recursion

$$f_{\mathrm{g}}^t(w, \mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \left\langle w, \Phi_{\mathrm{g}}\big(x^t, \mathbf{f}_{\mathrm{g}}^{\{t-k..t-1\}}(w, \mathbf{x}), y\big)\right\rangle . \tag{2}$$

2.3 Comparison

Although greedy inference is an approximation of exact inference, their different computational complexity leads to a more nuanced picture. Exact inference solves (1) using the Viterbi algorithm. It requires a time proportional to $DT\mathrm{card}(\mathcal{Y})^{k+1}$ and becomes intractable when the order k of the Markov assumption increases. On the other hand, the recursion (2) runs in time proportional to $DT\mathrm{card}(\mathcal{Y})$. Therefore greedy inference is practicable with large k.

In practice, greedy inference with large k can sometimes achieve a higher accuracy than exact inference with Markov assumptions of lower order.

3 Training

In this section we write the convex optimization problem used for determining the parameter vector for both cases of exact and greedy inference. We first present a large margin formulation of the multiclass problem and show how it applies to both problems.

3.1 Large-Margin Multiclass Problem

We consider training patterns $p_1 \ldots p_n$ and their classes $c_1 \ldots c_n$. Following the formulation of large-margin learning with interdependent output spaces [2,3], the parameters of a prediction function of the form $f(p) = \arg\max_c \langle w, \Phi(p, c)\rangle$ can be determined by minimizing the convex function

$$\min_{w} \quad \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n} \xi_i \tag{3}$$

$$\text{subject to} \quad \begin{array}{ll} \forall i & \xi_i \geq 0 \\ \forall i & \forall c \neq c_i \quad \langle w, \Phi(p_i, c_i) - \Phi(p_i, c)\rangle \geq \Delta(c_i, c) - \xi_i \,, \end{array}$$

where $\Delta(c_i, c)$ is the true loss incurred by predicting class c instead of the true class c_i. Following [6], we rewrite this optimization problem in dual form by introducing one coefficient β_i^c for each pattern p_i and each class $c \in \mathcal{C}$.

$$\max_{\boldsymbol{\beta}} \quad -\sum_{i,c} \Delta(c, c_i)\beta_i^c - \frac{1}{2}\sum_{i,j,c,\bar{c}} \beta_i^c \beta_j^{\bar{c}} \langle \Phi(p_i, c), \Phi(p_j, \bar{c})\rangle$$

$$\text{subject to} \quad \begin{array}{ll} \forall i & \forall c \quad \beta_i^c \leq \delta(c, c_i)\, C \\ \forall i & \quad {}_c\beta_i^c = 0 \end{array} \tag{4}$$

where $\delta(c, \bar{c})$ is 1 when $c = \bar{c}$ and 0 otherwise. The solution of the primal problem is then recovered from the optimal coefficients β_i^c as

$$w = \sum_{i,c} \beta_i^c \Phi(p_i, c)\,.$$

Multiple algorithms have been proposed to efficiently solve problem (4) even in cases where the number of classes is very large [3,6]. As such, an optimizer of problem (4) can be used for learning the parameters of sequence labelling models for both exact and greedy inference. For clarity in the presentation, we give the instantiation of the dual objective for both problems using the notations introduced in section 2.

3.2 Training for Exact Inference

Since the exact inference prediction function (1) can be written under the form $\arg\max_c \langle w, \Phi(p, c)\rangle$, the above formulation applies directly. The patterns p_i are the token sequences $\mathbf{x}_i$ and the classes c are complete label sequences $\mathbf{y}$. The feature function $\Phi(p_i, c) = \Phi_{\mathrm{e}}(\mathbf{x}_i, \mathbf{y})$ has been defined in (1) and the loss $\Delta(\mathbf{y}, \bar{\mathbf{y}})$ is the Hamming distance between the sequences $\mathbf{y}$ and $\bar{\mathbf{y}}$.

The dual problem is then

$$\begin{aligned} \max_{\boldsymbol{\beta}} \quad & -\sum_{i,\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i)\beta_i^{\mathbf{y}} - \frac{1}{2}\sum_{ij}\sum_{\mathbf{y}\bar{\mathbf{y}}} \beta_i^{\mathbf{y}}\beta_j^{\bar{\mathbf{y}}} K_{\mathrm{e}}^{ij\mathbf{y}\bar{\mathbf{y}}} \\ & \text{subject to} \quad \begin{array}{ll} \forall i & \forall \mathbf{y} \quad \beta_i^{\mathbf{y}} \le \delta(\mathbf{y}, \mathbf{y}_i)\, C \\ \forall i & \sum_{\mathbf{y}} \beta_i^{\mathbf{y}} = 0\,. \end{array} \end{aligned} \tag{5}$$

with the kernel matrix $K_{\mathrm{e}}^{ij\mathbf{y}\bar{\mathbf{y}}} = \langle \Phi_{\mathrm{e}}(\mathbf{x}_i, \mathbf{y}), \Phi_{\mathrm{e}}(\mathbf{x}_j, \bar{\mathbf{y}})\rangle$.

The solution is then $w = \sum_{i\mathbf{y}} \beta_i^{\mathbf{y}}\, \Phi_{\mathrm{e}}(\mathbf{x}_i, \mathbf{y})$.

3.3 Training for Greedy Inference

The greedy inference prediction function (2) does not readily have the form $\arg\max_c \langle w, \Phi(p, c)\rangle$ because of its recursive structure. However, each prediction f_{g}^t has the desired form with one pattern p_{it} for each training token x_i^t, and with classes c taken from the set of labels and compared with $\Delta(y, \bar{y}) = 1 - \delta(y, \bar{y})$.

This approach leads to difficulties because the feature function $\Phi(p_{it}, y) = \Phi_{\mathrm{g}}(x_i^t, \mathbf{f}_{\mathrm{g}}^{\{t-k..t-1\}}, y)$ depends on the prediction function. We avoid this difficulty by approximating the predicted labels $\mathbf{f}_{\mathrm{g}}^{\{t-k..t-1\}}$ with the true labels $\mathbf{y}_i^{\{t-k..t-1\}}$.

The corresponding dual problem is then

$$\begin{aligned} \max_{\boldsymbol{\beta}} \quad & -\sum_{ity} \Delta(y, y_i^t)\beta_{it}^y - \frac{1}{2}\sum_{itjr}\sum_{y\bar{y}} \beta_{it}^y\beta_{jr}^{\bar{y}} K_{\mathrm{g}}^{itjry\bar{y}} \\ & \text{subject to} \quad \begin{array}{ll} \forall i,t & \forall y \quad \beta_{it}^y \le \delta(y, y_i^t)\, C \\ \forall i,t & \sum_{y} \beta_{it}^y = 0\,. \end{array} \end{aligned} \tag{6}$$

with the kernel matrix $K_{\mathrm{g}}^{itjry\bar{y}} = \Big\langle \Phi_{\mathrm{g}}(x_i^t, \mathbf{y}_i^{\{t-k..t-1\}}, y)\,,\, \Phi_{\mathrm{g}}(x_j^r, \mathbf{y}_j^{\{r-k..r-1\}}, \bar{y})\Big\rangle$.

The solution is then $w = \sum_{ity} \beta_{it}^y\, \Phi_{\mathrm{g}}(x_i^t, \mathbf{y}_i^{\{t-k..t-1\}}, y)$.

3.4 Discussion

Both dual problems (5) and (6) are defined using very different sets of coefficients $\boldsymbol{\beta}$. Experiments (section 6) show considerable differences in sparsity. Yet the two kernel matrices K_{e} and K_{g} generate parameter vectors w in the same feature space which is determined by the choice of the feature function Φ_{g}, or equivalently the choice of the kernel K_{g}.

We use the following kernels in the rest of this paper.

$$K_{\mathrm{g}}^{itjry\bar{y}} = \delta(y,\bar{y}) \;\; k(x_i^t, x_j^r) + \sum_{s=1}^{k} \delta({y_i}^{t-s}, {\bar{y}_j}^{r-s}) \quad ,$$

$$K_{\mathrm{e}}^{ij\mathbf{y}\bar{\mathbf{y}}} = \sum_{tr} \delta(y^t, \bar{y}^r) \;\; k(x_i^t, x_j^r) + \sum_{s=1}^{k} \delta({y}^{t-s}, {\bar{y}}^{r-s}) \quad ,$$

where $k(x,\bar{x}) = \langle x, \bar{x} \rangle$ is a linear kernel defined on the tokens. These two kernels satisfy the identity $\Phi_{\mathrm{e}}(\mathbf{x},\mathbf{y}) = \sum_i \Phi_{\mathrm{g}}(x^t, \mathbf{y}^{\{t-k..t-1\}}, y^t)$ by construction. Furthermore, the exact inference kernel K_{e} is precisely the kernel proposed in [2].

The greedy kernel approximates the predicted labels with the true labels. The same approximation was used in LaSO [8] and in the first iteration of SEARN [4]. In the context of an online algorithm, other approximations would have been possible, such as the sequence of predicted labels for the previous values of the parameter. However, the simpler approximation works slightly better in our experiments.

4 Online Optimization

This section discusses the online optimization of problems (4), and therefore problems (5) and (6). We call our algorithm OLaRank (for Online LaRank), because it uses the same building blocks as the LaRank algorithm [6]. We first summarize these building blocks. Then, we describe how we adapted the original LaRank for our purposes.

4.1 OLaRank Building Blocks

The LaRank algorithm is a batch optimizer of the dual problem (4), for the specific case of multiclass classification with one representative vector per class. It can however be straightforwardly adapted to the general case of structured outputs we consider here. Assuming these adaptations done, LaRank can be applied to both problems (5) and (6).

Each elementary step of the LaRank algorithm maximizes the objective function (4) restricted to only two coefficients β_i^c and $\beta_i^{\bar{c}}$ associated with a same pattern p_i. Because these coefficients appear in the same equality constraint, they must move by opposite amounts. The maximization then simply becomes one-dimensional and can be solved analytically (see details in [6]). Each elementary step monotonically improves the dual objective function.

[6] define three ways to select pairs of coefficients, namely ProcessNew, ProcessOld, and Optimize. They prove that a reasonable mixture of these operations approaches the solution of problem (4) with a predefined accuracy, after a number of steps that grows linearly with the number of patterns and does not depend on the number of classes.

OLaRank uses exactly the same three basic operations, with a difference in the alternation between them. For clarity, we remind these definitions from [6]: *support vectors* are the pairs (p_i, c) such that $\beta_i^c \neq 0$, and *support patterns* are the patterns p_i for which at least one of the β_i^c coefficients is nonzero.

- ProcessNew randomly picks a fresh example (p_i, c_i) and chooses the best pair of coefficients β_i^c and $\beta_i^{\bar{c}}$ according to the gradient vector of the dual objective function.
- ProcessOld randomly picks a support pattern p_i and chooses the best pair of coefficient β_i^c and $\beta_i^{\bar{c}}$ according to the gradient vector.
- Optimize randomly picks a support pattern p_i and chooses the best pair of coefficient β_i^c and $\beta_i^{\bar{c}}$ according to the gradient vector but solely among the coefficients β_i^c that are not zero.

The ProcessOld and Optimize steps differ essentially in their computational costs. In the case of exact inference, ProcessOld requires to run the Viterbi algorithm to find the label sequence that maximizes the gradient for the chosen support pattern. Optimize only considers label sequences that are currently support vectors, avoiding the costly inference procedure. In the case of greedy inference, ProcessOld needs to compute the score of the current token for each possible label, while Optimize only considers the labels that are currently in the support vector expansion.

4.2 Scheduling

The LaRank algorithm schedules the three types of steps using a complex adaptive scheme that takes into account both the computing time and the progress of the objective function. This scheme works well for simple multiclass problems. However, we had mixed experiences with the exact inference models, because the ProcessOld operations incur a penalization in terms of computation time due to the Viterbi algorithm. In the end, ProcessOld was not sufficiently applied, leading to poor performance.

OLaRank implements a much simpler approach. We call Reprocess the combination of one ProcessOld step followed by ten Optimize steps. All our experiments are carried out by repeatedly performing one ProcessNew step followed by a predefined number n_{R} of Reprocess combinations. The number n_{R} depends on each problem and is determined like an hyper-parameter using a validation set (see table 1 and figure 3).

Notice that only performing ProcessNew steps (i.e. $n_{\mathrm{R}} = 0$) yields a typical *passive-aggressive* online algorithm [13]. Therefore, the Reprocess operation is the element that lets OLaRank match the test accuracy of batch optimization after a single sweep over the training data (see section 6).

5 Theoretical Analysis

This section displays theoretical results concerning the OLaRank algorithm presented in the previous section: a bound on the number of support vectors and another on the regret.

5.1 Sparsity Guarantee

All three basic operations (ProcessNew, ProcessOld and Optimize) do nothing unless they can find a search direction that fulfills two conditions: (1) the derivative of the objective function along this direction must be greater than $\tau > 0$, and, (2) a movement of size $\kappa > 0$ along that direction is possible without leaving the constraint polytope. Therefore OLaRank is an Approximate Stochastic Witness Direction Search (ASWDS) algorithm as defined in [11]. The number of support vectors it adds during learning can be bounded by the following proposition:

Proposition 1. *While training on n examples,* OLaRank *will add no more than*

$$\min\left\{n(2+n_{\mathrm{R}}),\ \max\{\frac{2\rho_{max}nC}{\tau^2},\frac{2nC}{\kappa\tau}\}\right\}$$

support vectors, with $\rho_{max} = \max_{i,c} ||\Phi(p_i, c) - \Phi(p_i, c_i)||^2$.

The bound is actually a bound on the number of ProcessNew and ProcessOld iterations, since each ProcessNew adds either 0 or 2 support vectors, and each ProcessOld adds either 0 or 1 support vector. The term $n(2 + n_{\mathrm{R}})$ is immediate considering the scheduling described in section 4. The term $\max\{\frac{2\rho_{max}nC}{\tau^2}, \frac{2nC}{\kappa\tau}\}$ is a bound on the maximal number of optimization steps carried out by LaRank (and therefore OLaRank). Its proof follows the theorem of [6].

In practice, the smaller term is obviously $n(2+n_{\mathrm{R}})$, since reasonable values of n_{R} are between 1 and 20. The interest of the second term of the bound is at the limit when n_{R} tends to infinity. Then OLaRank converges, at each round, to a $\kappa\tau$-approximate solution of problem (4), restricted to consider only the examples that are currently support patterns (see theorem 18 of [11] for the convergence of ASWDS algorithms). In that case, the bound proves that the number of support vectors still grows at most linearly with the number of examples given κ and τ.

5.2 Regret Bound

The OLaRank algorithm performs an iterative optimization of the dual, where only the parameters corresponding to already seen examples can be modified at each step. The primal-dual view of online learning of [12] allows to obtain online learning rates for that kind of algorithms in the case of SVMs for binary classification. In this section, we extend their result to structured predictors (i.e. online optimizers of equation (4)).

Regret Bound for Online Structured Predictors. The learning rates are expressed with the notion of *regret* defined by the difference between the mean loss incurred by the algorithm on the course of learning and the empirical loss of a given weight vector,

$$\text{regret}(n, w) = \frac{1}{n}\sum_{i=1}^{n} \ell(w_i, (p_i, c_i)) - \frac{1}{n}\sum_{i=1}^{n} \ell(w, (p_i, c_i))$$

with w_i the primal weight vector before seeing the i-th example, and $\ell(w, (p, c))$ the loss incurred by any weight vector w on the example (p, c). In our setup, the loss $\ell(w_i, (p_i, c_i))$ is defined as

$$\max_{c \in \mathcal{C}} \Delta(c_i, c) - \langle w, \Phi(p_i, c_i) - \Phi(p_i, c) \rangle_+$$

where $[x]_+ = \max(x, 0)$.

The following theorem gives a bound on the regret for any algorithm performing an online optimization of the dual of equation (4):

Theorem 1. *Assume that for all i, the dual increase after seeing example (p_i, c_i) is at least $C\mu_\rho(\ell(w_i, (p_i, c_i)))$, with*

$$\mu_\rho(x) = \frac{1}{\rho C} \min(x, \rho C) \left(x - \frac{1}{2} \min(x, \rho C) \right)$$

then, we have:

$$\forall w,\ \textit{regret}(n, w) \leq \frac{||w||^2}{2nC} + \frac{\rho C}{2}.$$

The proof exactly follows section 5 of [12]. The crucial point of this theorem is to directly relate the dual increase when seeing an example and the regret bound: the more we can prove that the dual increases in the course of learning, the more we can have guarantees on the performance.

Application. The following result allows to use theorem 1 to bound the regret for the OLaRank algorithm:

Proposition 2. *For a given i, the dual increase after performing a* ProcessNew *step on example (p_i, c_i) is equal to $C\mu_{\rho^i}(\ell(w_i, (p_i, c_i)))$, with $\rho^i = ||\Phi(p_i, c_i) - \Phi(p_i, c_i^*)||^2$ and $c_i^* = \arg\max_{c \in \mathcal{C}} \left(\Delta(c_i, c) + \langle w_i, \Phi(p_i, c) \rangle \right)$.*

This proposition is easily established by directly calculating the dual increase caused by ProcessNew step (see [6]) and expressing the result using the function μ_ρ. Since Reprocess cannot decrease the dual, the whole OLaRank algorithm increases the dual by at least $C\mu_{\rho^i}(\ell(w_i, (p_i, c_i)))$ after seeing example i. Moreover, as μ_ρ monotonically decreases with ρ theorem 1 can be applied to OLaRank with $\rho = \max_i \rho^i$.

Interpretation. Proposition 2 first shows that OLaRank has the same guarantees (in terms of regret) than a typical *passive-aggressive* algorithm as the latter is equivalent to performing only ProcessNew operations.

In addition, Theorem 1 provides a partial justification of the Reprocess function. Indeed, it expresses that we can relate the dual increase to the regret. As such, if, for instance, Reprocess operations bring a dual increase of the same order of magnitude as ProcessNew operations at each round, then the regret of OLaRank would be typically two times smaller than the current bound. Although we do not have any analytical results concerning the dual increase ratio between ProcessNew and Reprocess operations, the theorem suggests that the true regret of OLaRank should be much smaller than the bound.

Finally, we can notice that the regret we consider here does not match the true applicative setting of greedy inference. Indeed, we consider in the regret bound a set of examples that is fixed independently of the parameter vector w with which we compare. But on test examples the greedy inference scheme uses the past predictions instead of the true labels. Nevertheless the bound is informative to compare online to batch learning. Indeed, if we consider the examples (p_i, c_i) in the regret bound to be the training set, Theorem 1 relates the online error with the error of the batch optimal. Then, we can claim that the online error of OLaRank will not be too far from the batch optimal trained with the same set of examples. The partial justification for the Reprocess function of the previous paragraph is still valid.

6 Experiments

This section reports experiments performed on various label sequence learning tasks to study the behavior of our online learning algorithm. We denote OLaRankGreedy, OLaRank using greedy inference and OLaRankExact when using exact inference. Since such tasks are common in the recent literature, we focus on fully supervised tasks where labels are provided for every time index. After presenting the experimental tasks we chose, we compare the performances of OLaRankExact and OLaRankGreedy to both batch and online methods to empirically validate their efficiency. We then investigate how the choice of the inference method influences the performances.

6.1 Experimental Setup

Experiments were carried out on three datasets. The Optical Character Recognition dataset (OCR) contains handwritten words, with average length of 8 characters, written by 150 human subjects and collected by [14]. This is a small dataset for which the performance evaluation is performed using 10-fold cross-validation. The Chunking dataset from the CoNLL 2000 shared task[2] consists of sentences divided in syntactically correlated segments or chunks. This dataset has more than 75,000 input features. The Wall Street Journal dataset[3] (WSJ) is a larger dataset with around 1 million words in more than 40,000 sentences and with a large number of features. The labels associated with each word are "part-of-speech" tags.

[2] http://www.cnts.ua.ac.be/conll2000/chunking/

[3] http://www.cis.upenn.edu/~treebank/

Table 1. Datasets and parameters used for the experiments

	Training Set Sequences(Tokens)	Testing Set Sequences(Tokens)	Classes	Features	C	OLaRankGreedy n_R	OLaRankGreedy k	OLaRankExact n_R	OLaRankExact k
OCR	650 (∼4,600)	5500 (∼43,000)	26	128	0.1	5	10	10	1
Chunking	8,931 (∼212,000)	2,012 (∼47,000)	21	∼76,000	0.1	1	2	5	1
WSJ	42,466 (∼1,000,000)	2,155 (∼53,000)	44	∼130,000	0.1	1	2	5	1

Table 1 summarizes the main characteristics of these three datasets and specifies the parameters we have used for both batch and online algorithms: the constant C, the number n_R of Reprocess steps for each ProcessNew step, and the order k of the Markov assumptions. They have been chosen by cross-validation for the batch setting, online algorithms using the same parameters as their batch counterparts. Exact inference algorithms such as OLaRankExact are limited to first order Markov assumptions for tractability reasons.

6.2 General Performances

We report the training times for a number of algorithms as well as the percentage of correctly predicted labels on the test sets (For Chunking, we also provide F1 scores on test sets). Results for exact inference algorithms are reported in table 2. Results for greedy inference algorithms are reported in table 3.

Batch Counterparts. As discussed in the introduction, our main points of comparison are the prediction accuracies achieved by batch algorithms that fully optimize the same dual problems as our online algorithms.

In the case of exact inference, our baseline is given by the SVM-HMM results using the cutting plane optimization algorithm described by [3]. In the case of greedy inference, we simply produced baseline results by running OLaRankGreedy several times over the training set until the Karush-Kuhn-Tucker conditions are satisfied. These results are labelled LaRankGreedyBatch.

Tables 2 and 3 show that both OLaRankGreedy and OLaRankExact reach competitive testing set performances relative to these baselines while saving a lot of training time.

Figure 1 depicts relative time increments. Denoting t_0 the running time of a model on a small portion of the training set of size s_0, the time increment on a training set of size s is defined as t_s/t_0. We also define the corresponding size increment as s/s_0. This allows to represent scaling in time for different models. Figure 1 thus shows that our models scale linearly in time while a common batch method as SVM-HMM does not.

The dual objective values reached by the online algorithms based on OLaRank and by their batch counterparts are quite similar as presented on table 4. OLaRankExact and OLaRankGreedy have good optimization abilities; they both reach a dual value close to the convergence point of their corresponding batch algorithms. We also provide the dual of PAExact and PAGreedy, the *passive-aggressive* versions (i.e. without Reprocess) of OLaRankExact and

Table 2. Compared accuracies and times of methods using exact inference

		OCR	Chunking *(F1 score)*	WSJ
CRF	Test. accuracy (%)	-	96.03 *(93.75)*	96.75
(batch)	Train. time (sec.)	-	568	3,400
SVM-HMM	Test. accuracy (%)	78.20	95.98 *(93.64)*	96.81
(batch)	Train. time (sec.)	180	48,000	350,000
CRF	Test. accuracy (%)	-	95.26 *(92.47)*	94.42
(online)	Train. time (sec.)	-	30	240
PerceptronExact	Test. accuracy (%)	51.44	93.74 *(89.31)*	91.49
(online)	Train. time (sec.)	0.2	10	180
PAExact	Test. accuracy (%)	56.13	95.15 *(92.21)*	94.67
(online)	Train. time (sec.)	0.5	15	185
OLaRankExact	Test. accuracy (%)	75.77	95.82 *(93.34)*	96.65
(online)	Train. time (sec.)	4	130	1380

Table 3. Compared accuracies and times of methods using greedy inference

		OCR	Chunking *(F1 score)*	WSJ
LaRankGreedyBatch	Test. accuracy (%)	83.77	95.86 *(93.59)*	96.63
(batch)	Train. time (sec.)	15	490	9,000
PerceptronGreedy	Test. accuracy (%)	51.82	93.24 *(88.84)*	92.70
(online)	Train. time (sec.)	0.05	3	10
PAGreedy	Test. accuracy (%)	61.23	94.61 *(91.55)*	94.15
(online)	Train. time (sec.)	0.1	5	25
OLaRankGreedy	Test. accuracy (%)	81.15	95.81 *(93.46)*	96.46
(online)	Train. time (sec.)	1.4	20	175

OLaRankGreedy. These low values illustrate the crucial influence of Reprocess in the optimization process, independent of the inference method.

Other Comparisons. We also provide comparisons with a number of popular algorithms for both exact and greedy inference.

For exact inference, the CRF results were obtained using a fast stochastic gradient descent implementation[4] of Conditional Random Fields: online results were obtained after one stochastic gradient pass over the training data; batch results were measured after reaching a performance peak on a validation set. The PerceptronExact results were obtained using the perceptron update described by [7] and the same exact inference scheme as OLaRankExact. The PAExact results were obtained with the *passive-aggressive* version of OLaRankExact, that is without Reprocess or Optimize steps.

For greedy inference, we report results for both PerceptronGreedy and PA-Greedy. Like OLaRank, these algorithms were used in a strict online setup, performing only a single pass over the training examples.

[4] `http://leon.bottou.org/projects/sgd`

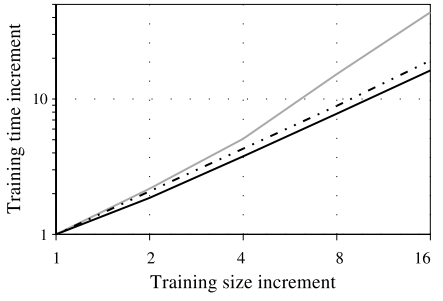


Fig. 1. Scaling in time on Chunking dataset, log-log plot. Solid black line: OLaRankGreedy, Dashed black line: OLaRankExact, Gray line: SVM-HMM.

Table 4. Values of dual objective after training phase

	Chunking	WSJ
SVM-HMM (batch)	1360	9072
PAExact (online)	443	2122
OLaRankExact (online)	1195	7806
LaRankGreedyBatch (batch)	940	8913
PAGreedy (online)	410	2922
OLaRankGreedy (online)	905	8505

Results in tables 2 and 3 clearly display a gap between the accuracies of these common online methods and the accuracies achieved by our two algorithms OLaRankGreedy and OLaRankExact. The OLaRank based algorithms are the only online algorithms able to match the accuracies of the batch algorithms. Although higher than those of other online algorithms, their training times remain reasonable. For example, on our largest dataset, WSJ, OLaRankGreedy reaches a test set accuracy competitive with the most accurate algorithms while requiring less training time than PerceptronExact (about four milliseconds per training sequence).

Results on the Chunking and WSJ benchmarks have been widely reported in the literature. Our Chunking results are competitive with the best results reported in the evaluation of the CoNLL 2000 shared task [15] (F1 score 93.48). More recent works include including [16] (F1 score 94.13, training time 800 seconds) and [17] (F1 score 94.19, training time 5000 seconds). The Stanford Tagger [18] reaches 97.24% accuracy on the WSJ task but requires 150,000 seconds of training. These state-of-the-art systems slightly exceed the performances reported in this work because they exploit highly engineered feature vectors. Both OLaRankExact and OLaRankGreedy need a fraction of these training times to achieve the full potential of our relatively simple feature vectors.

6.3 Comparing Greedy and Exact Inference

This section focuses on an empirical comparison of the differences caused by the inference scheme on learning.

Inference Cost. The same training set contains more training examples for an algorithm based on a greedy inference scheme. This has a computational cost. However the training time gap between PAExact and PAGreedy in tables 2 and 3 indicates that using exact inference entails much higher computational costs because the inference procedure is more complex (see section 2.3).

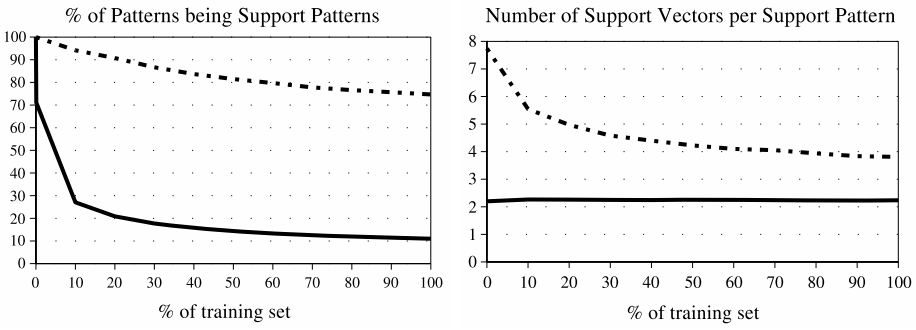


Fig. 2. Sparsity measures during learning on Chunking dataset. (Solid line: OLaRankGreedy, Dashed line: OLaRankExact.).

Sparsity. As support vectors for OLaRankExact are complete sequences, local dependencies are not represented in an invariant fashion. OLaRankExact thus needs to store an important proportion of its training examples as support pattern while OLaRankGreedy only requires a small fraction of them as illustrated in figure 2. Moreover, for each support pattern, OLaRankExact also needs to store more support vectors.

Reprocess. Figure 3 displays the gain in test accuracy that OLaRankGreedy and OLaRankExact get according to the number of Reprocess. This gain is computed relatively to the *passive-aggressive* algorithms which are similar but do not perform any Reprocess. OLaRankExact requires more Reprocess (10 on OCR) than OLaRankGreedy (only 5) to reach its best accuracy. This empirical result is verified on all three datasets. Using exact inference instead of greedy inference causes additional computations in the OLaRank algorithm.

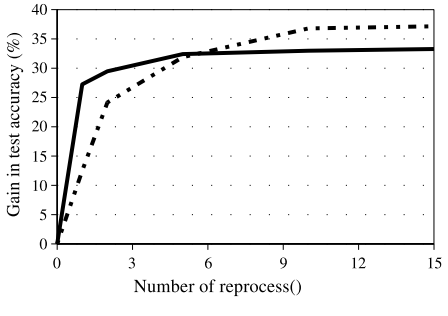


Fig. 3. Gain in test accuracy compared to the *passive-aggressives* according to n_R on OCR. (Solid line: OLaRankGreedy, Dashed line: OLaRankExact).

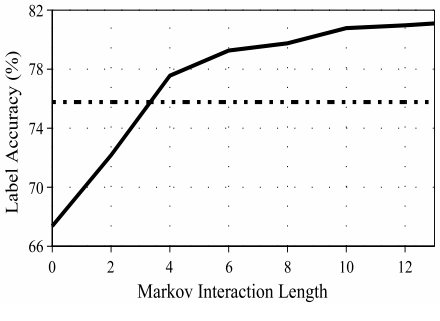


Fig. 4. Test accuracy according to the Markov interaction length on OCR. (Solid line: OLaRankGreedy, Dashed line: OLaRankExact for which $k = 1$).

Markov Assumption Length. This section indicates that using exact inference in our setup involves both time and sparsity penalties. Moreover the loss in accuracy that could occur when using a greedy inference process and not an exact one can be compensated by using Markov assumptions of order higher than 1. As shown on figure 4 it can even lead to better generalization performances.

7 Conclusion

The OLaRank algorithm applied to sequence labelling combines the linear scaling property of perceptrons and the accuracy of batch solvers.

Using the OLaRank algorithm with both exact and greedy inference leads to two competitive sequence labelling algorithm. Both learn in a single pass over the training data and reach the performance of equivalent batch algorithms. Both offer training times that scale linearly with the number of examples. Both have been shown to achieve good performances on well studied benchmark tasks.

Online learning and greedy inference offer the most attractive combination of short training time, high sparsity and accuracy.

Acknowledgments

Part of this work was funded by NSF grant CCR-0325463 and by the Network of Excellence IST-2002-506778 PASCAL. Antoine Bordes was also supported by the DGA.

References

1. Bakır, G., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B., Vishwanathan, S.V.N. (eds.): Predicting Structured Outputs. MIT Press, Cambridge (2007)
2. Altun, Y., Tsochantaridis, I., Hofmann, T.: Hidden Markov support vector machines. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003). ACM Press, New York (2003)
3. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
4. Daumé III, H., Langford, J., Marcu, D.: Search-based structured prediction as classification. In: NIPS Workshop on Advances in Structured Learning for Text and Speech Processing, Whistler, Canada (2005)
5. Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2004)
6. Bordes, A., Bottou, L., Gallinari, P., Weston, J.: Solving multiclass support vector machines with LaRank. In: Ghahramani, Z. (ed.) Proceedings of the 24th International Machine Learning Conference, Corvallis, Oregon. OmniPress (2007)
7. Collins, M.: Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: EMNLP 2002: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Morristown, NJ, pp. 1–8. Association for Computational Linguistics (2002)

8. Daumé III, H., Marcu, D.: Learning as search optimization: approximate large margin methods for structured prediction. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 169–176. ACM Press, New York (2005)
9. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory 50(9), 2050–2057 (2004)
10. Murata, N., Amari, S.: Statistical analysis of learning dynamics. Signal Processing 74(1), 3–28 (1999)
11. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. Journal of Machine Learning Research 6, 1579–1619 (2005)
12. Shalev-Shwartz, S., Singer, Y.: A primal-dual perspective of online learning algorithms. Machine Learning Journal 69(2/3), 115–142 (2007)
13. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
14. Kassel, R.: A Comparison of Approaches on Online Handwritten Character Recognition. PhD thesis, MIT. Spoken Language System Group (1995)
15. Kudoh, T., Matsumoto, Y.: Use of support vector learning for chunk identification. In: Proceedings of CoNLL 2000 and LLL 2000, pp. 252–259 (2000)
16. Zhang, T., Damerau, F., Johnson, D.: Text chunking based on a generalization of winnow. Journal of Machine Learning Research 2, 615–637 (2002)
17. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL 2003, pp. 213–220. Association for Computational Linguistics (2003)
18. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of HLT-NAACL 2003, pp. 252–259. Association for Computational Linguistics (2003)

Semi-supervised Classification from Discriminative Random Walks*

Jérôme Callut[1,2] , Kevin Françoisse[1,2],
Marco Saerens[1,2], and Pierre Dupont[1,3]

[1] UCL Machine Learning Group (MLG)
[2] Louvain School of Management, ISYS unit,
Université catholique de Louvain,
B-1348 Louvain-la-Neuve, Belgium
{Jerome.Callut, Kevin.Francoisse, Marco.Saerens}@uclouvain.be
[3] Department of Computing Science and Engineering, INGI,
Université catholique de Louvain,
B-1348 Louvain-la-Neuve, Belgium
Pierre.Dupont@uclouvain.be

Abstract This paper describes a novel technique, called $\mathcal{D}$-walks, to tackle semi-supervised classification problems in large graphs. We introduce here a betweenness measure based on passage times during random walks of bounded lengths. Such walks are further constrained to start and end in nodes within the same class, defining a distinct betweenness for each class. Unlabeled nodes are classified according to the class showing the highest betweenness. Forward and backward recurrences are derived to efficiently compute the passage times. $\mathcal{D}$-walks can deal with directed or undirected graphs with a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes. Experiments on various real-life databases show that $\mathcal{D}$-walks outperforms NetKit [5], the approach of Zhou and Schölkopf [15] and the regularized laplacian kernel [2]. The benefit of $\mathcal{D}$-walks is particularly noticeable when few labeled nodes are available. The computation time of $\mathcal{D}$-walks is also substantially lower in all cases.

1 Introduction

Mining and learning problems involving structured data such as graphs, trees or sequences have received much attention recently. The present work is concerned with semi-supervised classification of nodes in a graph. Given an input graph with some nodes being labeled, the problem is to predict the missing node labels. This problem has numerous applications such as classification of individuals in social networks, linked documents (*e.g.* patents or scientific papers) categorization or protein function prediction, to name a few. Even when the data is not

* Part of this work was supported by the STRATEGO project funded by the Region wallonne, Belgium.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 162–177, 2008.

initially structured as a graph, it can be convenient to build a neighborhood graph of similar examples from an affinity matrix.

Several approaches have been proposed to tackle semi-supervised classification problems in graphs. Kernel methods [12,15,14] embed the nodes of the input graph into an Euclidean feature space where a decision boundary can be estimated. For instance, the methods proposed in [15] rely on a kernel obtained by computing the inverse of a regularized graph Laplacian matrix. Such a kernel can be interpreted in terms of *commute times* during random walks performed on the graph. The method proposed in the present paper is related to the former approach, however it relies on *passage times* during random walks rather than on commute times. Despite their good predictive performance, kernel methods on graphs cannot easily scale up to large problems due to their high time complexity. NetKit [5] is an alternative relational learning approach. This general framework builds a model based on three components: a local classifier to generate class-priors, a relational classifier, which relies on the relations in the network to guess the class membership, and a so-called collective inferencing component which further refines the class predictions. The main advantage of this framework is that each of the three components can be instantiated with various existing methods making it easily adaptable to many situations. This flexibility comes however with a time-consuming tuning process to optimize performance. Compared to the above mentioned kernel methods, it provides good performance while having a lower time complexity.

The approach proposed in this paper, called $\mathcal{D}$-walks (discriminative random walks), relies on random walks performed on the input graph seen as a Markov chain. More precisely, a betweenness measure, based on passage times during random walks of bounded length, is derived for each class (or label category). Unlabeled nodes are assigned to the category for which the betweenness is the highest. The $\mathcal{D}$-walks approach has the following properties: (i) it has a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes; such a low complexity allows to deal with very large graphs, (ii) it can handle directed or undirected graphs, (iii) it can deal with multi-class problems and (iv) it has a unique hyper-parameter, the walk length, that can be tuned efficiently. Moreover, an extension of the technique is proposed to handle descriptive features on nodes (*e.g.* in social networks, such features could be the age or the number of children of individuals) via a similarity function.

This paper is organized as follows. Section 2 reviews basic notions about discrete time Markov chain and passage times during random walks. Section 3 introduces the $\mathcal{D}$-walks approach for semi-supervised classification in graphs. Section 4 introduces an extension to incorporate node features into the $\mathcal{D}$-walks classification technique. Links and differences between the $\mathcal{D}$-walks method and alternative approaches based on random walks are further described in section 5. Finally, section 6 reports comparative experimental results on real-life data.

2 Background

The semi-supervised node classification technique proposed in section 3 is based on *passage times* during random walks on the input graph seen as a *Markov chain* (MC). These classical notions are briefly reviewed in the present section. For a more detailed introduction to the MC theory, the reader is referred to the classical text books [4,7].

Definition 1 (Discrete time Markov Chain (MC)). *A* discrete time Markov Chain (MC) *is a stochastic process* $\{X_t \mid t \in \mathbb{N}\}$ *where the random variable* X *takes its value at any discrete time* t *in a countable set* $\mathcal{N}$ *and such that:*

$$P[X_t = q \mid X_{t-1}, X_{t-2}, \ldots, X_0] = P[X_t = q \mid X_{t-1}]. \tag{1}$$

This condition states that the probability of the next outcome only depends on the last value of the process (order 1 Markov property).

A finite MC can be represented by a 3-tuple $T = \langle \mathcal{N}, P, \boldsymbol{\iota} \rangle$ where $\mathcal{N}$ is a finite set of states with $n = |\mathcal{N}|$, P is a $n \times n$ row-stochastic matrix encoding the (homogeneous) transition probabilities $p_{qq'} = P[X_t = q' \mid X_{t-1} = q]$ for all $q, q' \in \mathcal{N}$ and $\boldsymbol{\iota}$ is an n-dimensional vector representing the initial probability distribution, *i.e.* $\iota_q = P[X_0 = q]$ for all $q \in \mathcal{N}$.

A *random walk* on a MC can be defined as follows: a random walker starts in a state q according to the initial distribution $\boldsymbol{\iota}$. Next, he moves to some state $q' \in \mathcal{N}$ according to the transition probability matrix P. Repeating this last operation $l \in \mathbb{N}$ times results in a l-step random walk. In a MC, a state q is said to be *absorbing* if there is a probability 1 to go from q to itself. In other words, once an absorbing state has been reached in a random walk, the walker will stay on this state forever. A MC for which there is a probability 1 to end up in an absorbing state is called an *absorbing MC*. In such a model, the state set can be divided into the absorbing state set $\mathcal{N}_A$ and its complementary set, the transient state set $\mathcal{N}_T = \mathcal{N} \setminus \mathcal{N}_A$. The *passage time* function counts the number of times a given node has been visited during a random walk.

Definition 2 (Passage Time). *Given a MC,* $T = \langle \mathcal{N}, P, \boldsymbol{\iota} \rangle$*, the* passage time *is a function* $\mathrm{pt} : \mathcal{N} \times \mathcal{N} \to \mathbb{N}$ *such that* $\mathrm{pt}(q)$ *is the number of times the process reaches the state* q*:*

$$\mathrm{pt}(q) = |\{t \in \mathbb{N} \mid X_t = q\}| \tag{2}$$

The *mean passage time* (MPT) denotes the expectation of this quantity: $\mathbb{E}[\mathrm{pt}(q)]$. The MPT is clearly infinite for absorbing states. For transient states, the MPT can be obtained by computing the so-called *fundamental matrix*: $N = (I - P_T)^{-1}$ where I is the $|\mathcal{N}_T| \times |\mathcal{N}_T|$ identity matrix and P_T is the transition probability matrix restricted to transient states (subscript T). The entry $n_{q'q}$ contains the MPT in state $q \in \mathcal{N}_T$ during random walks starting in state q'. Hence, $\mathbb{E}[\mathrm{pt}(q)] = [\boldsymbol{\iota}'_T N]_q$ where $\boldsymbol{\iota}'_T$ is the (transpose of the) initial probability vector reduced to

transient states[1]. It should be stressed that this expectation is taken over random walks of any (positive) length. In contrast, the betweenness proposed in section 3.3 relies on passage times computed for walks of bounded length.

3 Discriminative Random Walks

This section presents the $\mathcal{D}$-walks approach to perform semi-supervised classification of nodes in a graph. This technique is based on a betweenness measure computed from passage times during random walks performed in the input graph. The problem statement is detailed in section 3.1. A betweenness measure using random walks of unbounded length is introduced in section 3.2. Section 3.3 refines this measure by considering bounded walks up to a prescribed length. Section 3.4 shows how to classify unlabeled nodes based on the proposed betweenness measure.

3.1 Problem Statement

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote an input graph where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. In the sequel, $n = |\mathcal{N}|$ denotes the number of nodes and $m = |\mathcal{E}|$ the number of edges in the graph. Let also A denote the $n \times n$ adjacency matrix of $\mathcal{G}$. Since the graph $\mathcal{G}$ may be directed and weighted, the matrix A is not necessarily symmetric nor binary. Furthermore, it is assumed that A is an affinity matrix: the edge weights are positive and the higher the value, the easier the connection between the corresponding nodes. The graph $\mathcal{G}$ is assumed partially labeled. The nodes in the *labeled set* $\mathcal{L} \subset \mathcal{N}$ are assigned to a category from a discrete set $\mathcal{Y}$. The *unlabeled set* is defined as $\mathcal{U} = \mathcal{N} \setminus \mathcal{L}$. The label of a node $q \in \mathcal{L}$ is written y_q and $\mathcal{L}_y$ denotes the set of nodes in class y, with $n_y = |\mathcal{L}_y|$. A local consistency of the node labeling is assumed, *i.e.* nodes within a neighborhood are likely to share the same label. The task is to classify unlabeled nodes in the graph.

3.2 $\mathcal{D}$-Walks Betweenness

Random walks on a graph can be modeled by a discrete-time Markov chain $\{X_t \in \mathcal{N}\}_{t \in \mathbb{N}}$ (MC) describing the sequence of nodes visited during the walk. The random variable X_t represents the state of the MC reached at the discrete time index t. Since each state of the Markov chain corresponds to a distinct node of the graph, the state set of the MC is simply the set of nodes $\mathcal{N}$. The terms *nodes* and *states* are thus used interchangeably in this paper. The transition probability to state q' at time t, given that the last state is q at time $t-1$, is defined as:

$$P[X_t = q' \mid X_{t-1} = q] = p_{qq'} \triangleq \frac{a_{qq'}}{\sum_{q' \in \mathcal{N}} a_{qq'}}. \tag{3}$$

[1] It is assumed here that random walks cannot start in an absorbing state.

Thus, from any state q, the probability to jump to state q' is proportional to the weight $a_{qq'}$ of the edge from q to q' (and then normalized). These transition probabilities are stored in an $n \times n$ transition matrix $P = \{p_{qq'}\}_{q,q' \in \mathcal{N}}$.

We introduce *discriminative random walks* ($\mathcal{D}$-walks, for short) to define a betweenness measure used for node classification (see section 3.4). A $\mathcal{D}$-walk is a random walk starting in a labeled node and ending when any node having the same label (possibly the starting node itself) is reached for the first time.

Definition 3 ($\mathcal{D}$-walks). *Given a MC defined on the state set $\mathcal{N}$ and a class $y \in \mathcal{Y}$, a $\mathcal{D}$-walk is a sequence of state $q_0, \ldots, q_l$ such that $y_{q_0} = y_{q_l} = y$ and $y_{q_t} \neq y$ for all $0 < t < l$.*

The notation $\mathcal{D}^y$ refers to the set of all $\mathcal{D}$-walks starting and ending in a node of class y.

The betweenness function $B(q, y)$ measures how much a node $q \in \mathcal{U}$ is located "between" nodes of class $y \in \mathcal{Y}$. The betweenness $B(q, y)$ is formally defined as the expected number of times node q is reached during $\mathcal{D}^y$-walks.

Definition 4 ($\mathcal{D}$-walks betweenness). *Given an unlabeled node $q \in \mathcal{U}$ and a class $y \in \mathcal{Y}$, the* $\mathcal{D}$-walks betweenness function $\mathcal{U} \times \mathcal{Y} \to \mathbb{R}^+$ *is defined as follows:*

$$B(q, y) \triangleq \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}^y\right] \tag{4}$$

This betweenness measure is closely related to the one proposed by Newman in [6]. Our measure is however relative to a specific class y rather than to the whole graph, which is more informative in the context of classification. $B(q, y)$ can be computed using standard absorbing MC techniques (see section 2). Nodes belonging to $\mathcal{L}_y$ are first duplicated such that the original nodes are used as absorbing states and the duplicated ones as starting states. The transition matrix P is augmented as follows: (i) one duplicates the rows of P corresponding to nodes in $\mathcal{L}_y$ at the bottom of the matrix, (ii) one adds n_y columns full of zeroes[2] at the right of P and (iii) one defines $p_{qq'} = 1 \iff q' = q$ and 0 otherwise, for all $q \in \mathcal{L}_y$. The augmented matrix is denoted here by ${}^yP$. The initial distribution vector is adapted accordingly, resulting in the vector ${}^y\boldsymbol{\iota}$. The betweenness is finally computed as follows:

$$B(q, y) = [{}^y\boldsymbol{\iota}'_T (I - {}^yP_T)^{-1}]_q \tag{5}$$

where ${}^yP_T$ and ${}^y\boldsymbol{\iota}_T$ respectively denote the transition matrix and the initial distribution vector restricted to transient states (*i.e.* all states but those in $\mathcal{L}_y$). The betweenness computation has a computational complexity $\mathcal{O}(n^3)$ due to the matrix inversion. The space complexity is $\mathcal{O}(n^2)$ since the inverse matrix is generally dense even for sparse input graphs. Such high complexities makes the technique only tractable for graphs containing a few thousands nodes. Optimized matrix inversion, possibly relying on a spectral decomposition, can be used to

[2] The n_y added states are only used to start walks and cannot be reached from any node in the graph.

tackle this problem. We propose an alternative approach, detailed in the next section, by adapting the betweenness definition. It offers a much lower time and space complexity algorithm, which also proved to be more accurate for semi-supervised classification.

3.3 Bounded $\mathcal{D}$-Walks Betweenness

This section describes a betweenness measure based on $\mathcal{D}$-walks of bounded length. The notation $\mathcal{D}_l^y$ refers to the set of all $\mathcal{D}$-walks of length *exactly equal* to l, starting and ending in a node of class y. We also consider $\mathcal{D}_{\leq L}^y$ referring to the set of all bounded $\mathcal{D}$-walks *up to* a given length L. The bounded $\mathcal{D}$-walks betweenness measure $B_L(q, y)$ is formally defined hereafter.

Definition 5 (Bounded $\mathcal{D}$-walks betweenness). *Given an unlabeled node $q \in \mathcal{U}$ and a class $y \in \mathcal{Y}$, the* $\mathcal{D}$-walks betweenness function $\mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ *is defined as follows:*

$$B_L(q, y) \triangleq \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_{\leq L}^y\right]$$

Bounding the walk length has two major benefits: (i) better classification results are systematically obtained with respect to unbounded walks[3] in our experiments, (ii) the betweenness measure can be computed very efficiently. The unbounded betweenness can also be approximated with the bounded one by considering large but finite L values. More precisely, it can be shown that the bounded betweenness converges *almost geometrically fast* with L towards the unbounded betweenness value [1].

An efficient betweenness computation can be achieved using forward and backward variables, similar to those used in the Baum-Welch algorithm for Hidden Markov Models (HMM) parameter estimation [9]. Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the forward variable $\alpha^y(q, t)$ computes the probability to reach state q after t steps without passing through[4] nodes in class y, while starting initially from any state in class y. The forward variables are computed using the following recurrence:

$$\left|\begin{array}{ll} (\text{case } t = 1) & \alpha^y(q, 1) = \sum_{q' \in \mathcal{L}_y} \frac{1}{n_y} p_{q'q} \\ (\text{case } t \geq 2) & \alpha^y(q, t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \alpha^y(q', t-1)\, p_{q'q} \end{array}\right. \tag{6}$$

It is assumed that walks can start in any state of class y with a uniform probability $1/n_y$. Notice that the case $t = 1$ allows transitions outgoing from a state in class y (*i.e.* a starting state) whereas such transitions are forbidden in the induction case ($t \geq 2$), as walks are stopped as soon as a node in class y is reached.

[3] The maximum walk length has to be chosen for instance by cross-validation (see section 6).

[4] In contrast with *leaving from* a node q, *passing through* q means to jump from some node q' to q and then to leave from q.

Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the backward variable $\beta^y(q,t)$ computes the probability that state q is attained by the process t steps before reaching any node labeled y for the first time. The backward variables are computed using the following recurrence:

$$\begin{cases} (\text{case } t = 1) & \beta^y(q,1) = \sum_{q' \in \mathcal{L}_y} p_{qq'} \\ (\text{case } t \geq 2) & \beta^y(q,t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \beta^y(q', t-1)\, p_{qq'} \end{cases} \tag{7}$$

Transitions incoming to a state in class y are only allowed at the end of walks (case $t = 1$) since we are interested in walks ending when any node in $\mathcal{L}_y$ is reached for the first time. The time complexity of the forward and backward recurrences is $\Theta(mL)$, where m is the number of edges and L is the maximal walk length considered.

In order to compute $B_L(q,y)$, let us first calculate the MPT in a node $q \in \mathcal{U}$ during $\mathcal{D}_l^y$-walks (*i.e.* walks of length *exactly equal* to l): $\mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right]$. The (length-conditionned) passage time function $\mathrm{pt}(q)$ can be decomposed as a sum of indicator variables: $\mathrm{pt}(q) = \sum_{t=1}^{l-1} \mathbb{1}\{X_t = q\}$. Consequently, the desired expectation is given by

$$\mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right] = \mathbb{E}\left[\sum_{t=1}^{l-1} \mathbb{1}\{X_t = q\} \,\middle|\, \mathcal{D}_l^y\right] = \sum_{t=1}^{l-1} \mathbb{E}\left[\mathbb{1}\{X_t = q\} \mid \mathcal{D}_l^y\right] \tag{8}$$

$$= \sum_{t=1}^{l-1} P[X_t = q \mid \mathcal{D}_l^y] = \sum_{t=1}^{l-1} \frac{P[X_t = q \wedge \mathcal{D}_l^y]}{P[\mathcal{D}_l^y]} \tag{9}$$

The joint probability in the numerator of equation (9) can be computed as $P[X_t = q \wedge \mathcal{D}_l^y] = \alpha^y(q,t)\beta^y(q,l-t)$, that is the probability to start in any node of $\mathcal{L}_y$, to reach node q at time t and to complete the walk $l-t$ steps later. The probability to perform a $\mathcal{D}_l^y$-walk is obtained as $P[\mathcal{D}_l^y] = \sum_{q' \in \mathcal{L}_y} \alpha^y(q', l)$. Therefore, the desired expectation is given by

$$\mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right] = \frac{\sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{q' \in \mathcal{L}_y} \alpha^y(q', l)} \tag{10}$$

Finally, the betweenness measure based on walks *up to* length L is obtained as an expectation of the betweennesses for all length $1 \leq l \leq L$:

$$B_L(q,y) = \sum_{l=1}^{L} \frac{P[\mathcal{D}_l^y]}{Z}\, \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_l^y\right] \tag{11}$$

$$= \frac{\sum_{l=1}^{L} \sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{l=1}^{L} \sum_{q' \in \mathcal{L}_y} \alpha^y(q', l)} \tag{12}$$

$$= \frac{\left(\sum_{t=1}^{L} \alpha^y(q,t)\right)\left(\sum_{l=1}^{L-t} \beta^y(q,l)\right)}{\sum_{l=1}^{L} \sum_{q' \in \mathcal{L}_y} \alpha^y(q', l)} \tag{13}$$

The distribution $P[\mathcal{D}_l^y]$ is defined on all discrete times $t \in [1, \infty)$, however only walks up to length L are considered here. Therefore, we introduce a normalization constant Z ensuring that $\sum_{l=1}^{L} P[\mathcal{D}_l^y]/Z$ sums to one, making the expectation well-defined. Once the values of the forward and backward variables are stored in lattices, $B_L(q, y)$ can be obtained with a time complexity in $\Theta(L.n_y)$ by precomputing all the terms of the inner sum contained in the numerator of equation (13). The complexity for computing the betweenness for all unlabeled nodes with respect to a specific class y is thus $\Theta(m.L)$ as it is dominated by the cost of the recurrence computations. The space complexity is $\Theta(m + L.n)$, *i.e.* the space required to store the graph and the forward and backward lattices.

3.4 Classification Using the Betweenness

Unlabeled nodes are classified using a maximum *a posteriori* (MAP) decision rule from the betweenness computed for each class. The class-conditionned likelihood of a node q with respect to a specific class y is defined from its class betweenness as follows:

$$P[q \mid y] \triangleq \frac{B_L(q, y)}{\sum_{y' \in \mathcal{Y}} B_L(q, y')} \tag{14}$$

The label of a node $q \in \mathcal{U}$ is predicted by:

$$\hat{y}_q = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} P[q \mid y]\ P[y] \tag{15}$$

where $P[y]$ is estimated as the proportion of nodes belonging to class y. Taking the argmax on the class posteriors corresponds to a crisp node classification. A fuzzy node classification can be obtained by directly outputting the class posteriors. Finally, the time complexity to classify all unlabeled nodes using the bounded $\mathcal{D}$-walks betweenness is $\Theta(|\mathcal{Y}|.m.L)$ where m is the number of edges in the graph and L is the maximal walk length allowed. The space complexity is the same as for computing the betweenness relative to a specific class, *i.e.* $\Theta(m + L.n)$.

4 Incorporating Node Features

This section presents an extension of the $\mathcal{D}$-walks approach to take node features or attributes into account during classification. For instance, such features could be the age, the nationality or the marital status of individuals in a social network. Incorporating such features can clearly improve the classification performance as the system can discriminate using both structural and descriptive information on nodes. The feature vector corresponding to a node q is denoted by $\phi(q)$. We assume that a similarity function $k(\phi(q), \phi(q'))$ between feature vectors is available. The resulting $n \times n$ similarity matrix for all pairs of nodes is written K. Note that the similarity function needs not be a genuine Mercer kernel (*i.e.* matrix K is not required to be positive-definite). However, it is assumed that $k(\phi(q), \phi(q'))$ takes positive values.

Our approach for incorporating nodes features is based on a reweighting of the adjacency matrix. Each entry $a_{qq'}$ of the adjacency matrix is simply multiplied by the similarity between the concerned nodes $k_{q,q'}$. One of our modeling hypothesis (see section 3.1) states that the higher the edge weight the easier the connection. The edge reweighting proposed here is consistent with this assumption as the connection between two nodes will be reinforced proportionally to their feature-based similarity. In matrix form, the reweighted adjacency matrix ${}^{\phi}A$ is obtained as follows:

$$ {}^{\phi}A = A \circ K \tag{16}$$

where $\circ$ denotes the element-wise (Hadamard) product between matrices. This multiplicative reweighting preserves the sparseness of the graph and the similarity function needs only to be evaluated for pairs (q, q') such that $a_{qq'} > 0$. Consequently, the complexity of the $\mathcal{D}$-walks approach using node features is $\Theta(|\mathcal{Y}|.m.L + m.\mathrm{Sim})$ where Sim is the time complexity of one similarity function evaluation. The space complexity remains unchanged with respect to the original $\mathcal{D}$-walks approach.

5 Some Links between $\mathcal{D}$-Walks and Related Approaches

It is worth stressing the links and differences between the $\mathcal{D}$-walks method described in section 3.3 and competing approaches. To simplify a bit the analysis we restrict first our attention to undirected graphs. For such graphs, the weighted adjacency matrix A is symmetric.

The method of Zhou and Schölkopf [15], compared experimentally with $\mathcal{D}$-walks in section 6, can also be interpreted in terms of random walks. This method computes the matrix $S = D^{-1/2}AD^{-1/2}$ where D denotes the diagonal matrix of node degrees: $d_q = \sum_{q \in \mathcal{N}} a_{qq'}$. A specific element of S is simply given by:

$$ s_{qq'} = \frac{a_{qq'}}{\sqrt{d_q}\sqrt{d_{q'}}} \tag{17}$$

In contrast with the transition probability matrix $P = D^{-1}A$, the S matrix is not necessarily row-stochastic. Obviously, $P = S$ whenever every node has the same degree. Semi-supervised classification in [15] relies on the inverse of a regularized and normalized[5] laplacian $\tilde{L}$:

$$ \tilde{L}^{-1} = (I - \alpha S)^{-1} \text{ with } 0 < \alpha < 1 \tag{18}$$

In particular, the classification rule for the unlabeled node q can be written:

$$ \hat{y}_q = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{\{q' \mid y_{q'} = y\}} [\tilde{L}^{-1}]_{qq'}, \text{ where } y_{q'} \text{ is the label of node } q' \tag{19}$$

[5] The approach of Zhou et al [15] can be considered as a variant of [2,3] which uses the regularized laplacian matrix $D - \alpha A$ instead of the standard one. Performance of the later approach is also reported in our experiments.

Equation (18) can be reformulated using a Neumann series:

$$(I - \alpha S)^{-1} = \sum_{l=0}^{\infty} (\alpha S)^l = I + \alpha S + \alpha^2 S^2 + \dots \tag{20}$$

The first equality is valid only if the series converges[6]. Under this assumption, the sum of matrix powers in (20) is highly similar to those computed (however through recurrence relations) in the bounded $\mathcal{D}$-walks approach.

To push the analogy a bit further, let us consider a scaled transition probability matrix $\tilde{P} = \alpha P$. Such matrix defines a lazy random walk in the original graph since a transition from state q to a distinct state q' has a probability

$$\tilde{p}_{qq'} = \frac{\alpha\, a_{qq'}}{d_{q'}}, q \neq q' \tag{21}$$

and, consequently, the random walk remains on state q with probability $1 - \alpha$. Finally, equation (20) relies on the successive powers of αS and one specific element of this matrix is written:

$$\alpha s_{qq'} = \alpha \frac{a_{qq'}}{\sqrt{d_q}\sqrt{d_{q'}}} = \sqrt{\tilde{p}_{qq'}\tilde{p}_{q'q}} \tag{22}$$

One observes that the quantity of interest in this approach is the geometric mean of the transition probability in one direction and the opposite direction during such lazy random walks. This symmetric quantity is consistent with the commute times interpretation detailed in [15].

The unique hyper-parameter is here α which controls the degree of laziness of the random walks. In contrast, the unique hyper-parameter for the $\mathcal{D}$-walks method is the maximal walk length L, which corresponds to the maximal matrix power considered. This difference can be interpreted as a distinct regularization choice [10]. Note that bounding the maximal walk length may not be the most appropriate choice when defining a graph kernel. We argue that this choice is however better for the semi-supervised classification problem considered here, as confirmed experimentally in section 6.

Another important aspect of the $\mathcal{D}$-walks approach is to consider only walks that start and end in a labeled node of a given class but do not go through labeled nodes of the same class at intermediate steps. Intuitively, the random walks explain the connections with labeled nodes of a given class without diffusing unconditionally through the graph.

The method described in [15] was extended to directed graphs [14]. These approaches require that the random walk model has a unique stationary distribution. For directed graphs, this property can be satisfied by considering teleporting random walks as in [8]. The price to pay is the introduction of an additional hyper-parameter which defines the trade-off between the teleporting uniform probability and the natural transition probability derived from the adjacency matrix. The $\mathcal{D}$-walks approach does not require such an additional parameter

[6] This condition is satisfied if the spectral radius of αS is < 1.

to deal with directed graphs. This is an immediate consequence of bounding the walk length rather than relying on the asymptotic behavior of the walks.

An alternative semi-supervised classification method using random walks was proposed in [11]. This method considers walks starting from an unlabeled node and reaching a labeled node in exactly L steps. Again, nothing prevents the walks to go through labeled nodes of the same class during the intermediate steps. Szummer and Jaakkola also proposed the use of a different length parameter from each unlabeled node but this method did not pay off experimentally. The recurrences detailed in section 3.3 are also specific to the $\mathcal{D}$-walks approach. They are essential to guarantee a low computational complexity to be able to deal with large graphs.

6 Experiments

The experiments presented in this section have two goals. First, the classification performance of our approach is compared against competing methods. Secondly, the computing times on large-scale problems data are analyzed, demonstrating the efficiency and the scalability of the approach.

6.1 Data

The three case studies used for this paper are real-world representation of networked data. They all come from different domains that have been the subject of prior study in machine learning [5].

IMDb: The collaborative Internet Movie Database (`IMDb`) has several applications such as making movie recommendation or movie category classification. The classification problem focuses on the prediction of the movie notoriety (whether the movie is a box-office or not). It contains a graph of movies linked together whenever they share the same production company. The weight of an edge in the resulting graph is the number of production companies two movies have in common. The graph contains 1196 movies which have the following class distribution (see Table 1):

Table 1. Class distribution for the `IMDb` data set

Category	**Size**
High-revenue	572
Low-revenue	597
Total	1169
Majority class accuracy	51.07%
Number of Edges	40564
Mean degree	36.02
Min degree	1
Max degree	181

Table 2. Class distribution for the CORA data set

Category	Size
Case-Based	402
Genetic Algorithms	551
Neural Networks	1064
Probabilistic Methods	529
Reinforcement Learning	335
Rule Learning	230
Theory	472
Total	3583
Majority class accuracy	29.7%
Number of Edges	22516
Mean degree	6.28
Min degree	1
Max degree	311

CORA: The CORA dataset contains computer science research papers. It includes the full citation graph as well as the topic of each paper as labels. Two papers are linked if one cites the other. The weight of an edge is generally one unless two papers cite each other, in which case it is two. The graph contains 3582 nodes with the following class distribution (see Table 2):

WebKB: WebKB consists of sets of web pages gathered from four computer science departments (one for each university), with each page manually labeled into 6 categories: course, department, faculty, project, staff, and student. Two pages are linked by co-citation (if x links to z and y links to z, then x and y are co-citing z). The composition of the data set is shown in Table 3.

Table 3. Class distribution for the WebKB data set

	Size			
Category	**Cornell**	**Texas**	**Washington**	**Wisconsin**
course	54	51	170	83
department	25	36	20	37
faculty	62	50	44	37
project	54	28	39	25
staff	6	6	10	11
student	145	163	151	155
Total	346	334	434	348
Majority class accuracy	41.9%	48.8%	39.2%	44.5%
Number of Edges	26832	32988	30462	33250
Mean degree	77.55	98.77	70.19	95.55
Min degree	1	1	1	1
Max degree	191	215	286	229

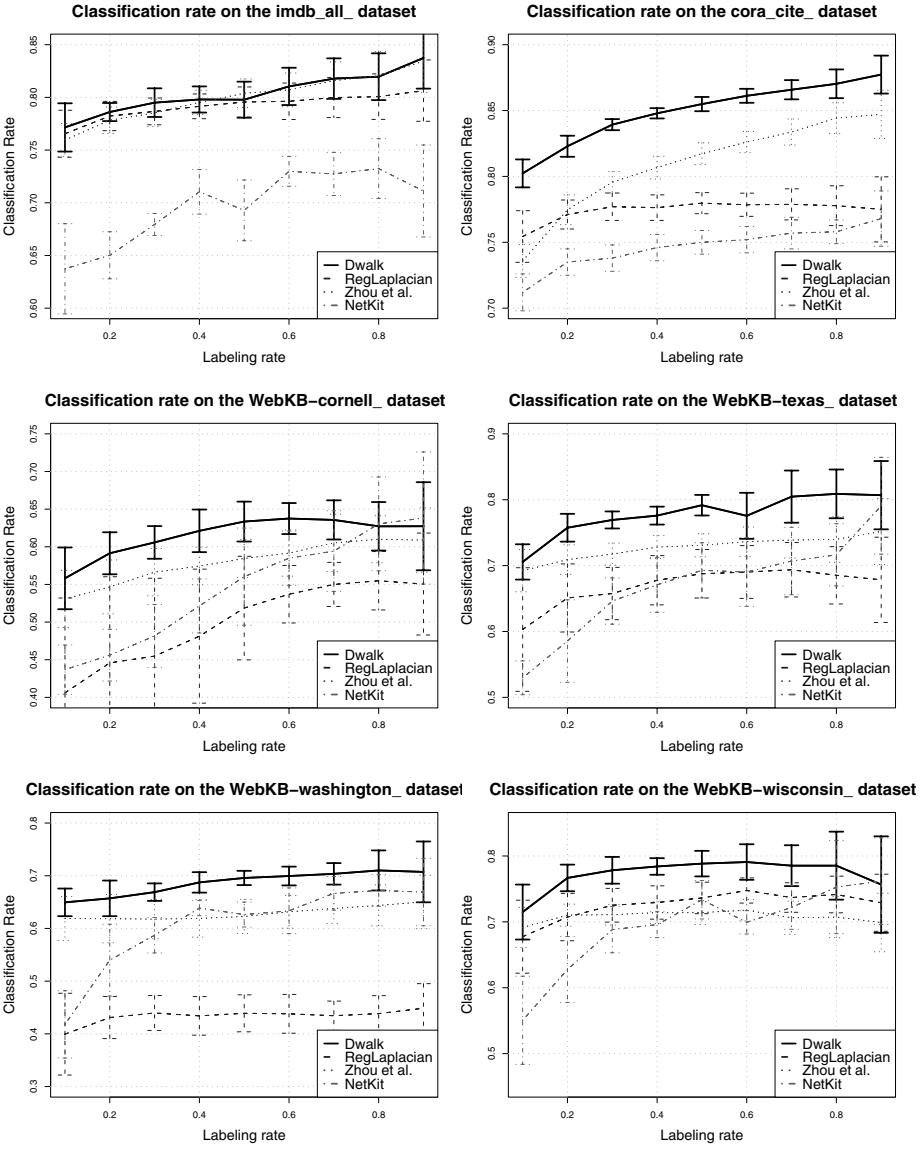


Figure 1. Classification rate of $\mathcal{D}$-walks and the three competing methods on each dataset. Error bars report standard deviations over 10 independent runs.

6.2 Methodology

Performances of the $\mathcal{D}$-walks approach and competing methods are reported in Figure 1 for the four datasets presented above. A network topology, along with some node labels, generally contains a considerable amount of useful information

for unlabeled nodes classification. Only a small proportion of labeled nodes might be sufficient to predict the unlabeled set. For this reason, we have considered several labeling rates, *i.e.* proportion of nodes for which the label is known. The labels of remaining nodes are removed and used as test. For each considered labeling rate, 10 random deletions were performed on which performances are averaged.

Our approach has been compared to the following existing methods: NetKit ([5]; "NetKit"), Zhou and Schölkopf method ([15]; "Zhou et al.") and the regularized laplacian kernel ([2] [3]; "RegLaplacian"). Hyper-parameters of each approach were optimized using an internal 10-folds cross-validation. For the $\mathcal{D}$-walks approach, the unique parameter to tune is L. The best value for L is computed in the range $\{1, \ldots, 100\}$. Optimal L values typically fall between 6 and 30, showing the interest of bounding the walk length (see the right hand side of Figure 2). Zhou et al. and the regularized laplacian kernel also require the regularization parameter α to be tuned between 0 and 1. Concerning NetKit, testing all the module configurations would have been very time-consuming. We chose here the parameters that generally provide good results [5]. More precisely, the local classifier inducer uses the class prior, the relational classifier inducer uses the weighted vote Relational Neighbor classifier and a relaxation labeling is used for the collective inferencing.

6.3 Results

This section details the classification accuracy of each method over the various datasets. Figure 1 reports classification rates on test data obtained by each approach as a function of the labeling rate.

As could be expected, the classification rate generally improves with higher labeling rate. We can clearly observe that the $\mathcal{D}$-walks approach outperforms competing approaches most noticeably when fewer labeled nodes are considered. Moreover, the variance of this method is generally significantly lower. $\mathcal{D}$-walks is also the fastest method. On data sets like `CORA` with 3583 nodes, it requires less than a second of CPU on a standard PC including the tuning of its hyper-parameter L with cross-validation. NetKit takes about 4.5 seconds and Zhou et al. approach requires about one minute when explicitly computing the matrix inverse.

The scalability of the $\mathcal{D}$-walks approach was assessed on much larger data sets which were artificially generated, up to 10 million edges. These random graphs were generated using the algorithm presented in [13]. This technique allows one to generate an undirected and unweighted graph with a prescribed degree sequence drawn from a power law. The parameters of the power law were tuned such that the mean degree equals 10. The computing times on a standard PC[7] for increasing graph sizes and L values are shown in Figure 2. Memory and time requirements are strongly limiting factors to apply the other approaches on such large graphs.

[7] Intel Core 2 Duo 2.4Ghz.

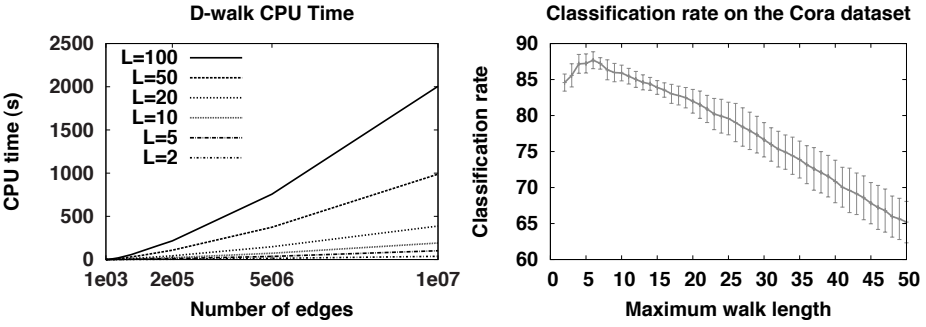


Figure 2. On the left, the CPU time of the $\mathcal{D}$-walks approach on artificially generated graphs of growing sizes, and for increasing maximum walk lengths L. The right side represents the classification rate on the `Cora` dataset, with a 0.9 labeling rate, when increasing the maximum walk length L. In this setting, the interest of bounding the walk length is clearly observed.

7 Conclusion

We propose in this paper a novel approach to the semi-supervised classification of nodes in a graph. The input graph is interpreted as a Markov chain (MC) in which random walks are performed. Particular random walks, called $\mathcal{D}$-walks, are introduced to define a node betweenness measure. A $\mathcal{D}$-walk starts in any node of a specific class y and ends as soon as some node of the same class is reached for the first time. The betweenness of a node q with respect to a class y is defined as the average number of times q is visited during $\mathcal{D}$-walks. Our betweenness measure is then refined by considering walks up to a prescribed length. Bounding the walk length provides systematically a better classification rate with the additional benefit that the betweenness can be computed very efficiently using forward and backward recurrences. The classification of all the unlabeled nodes in the graph can be performed with a time complexity $\Theta(|\mathcal{Y}|.m.L)$ where $|\mathcal{Y}|$ is the number of classes, m is the number of edges in the input graph and L is the maximum walk length considered. Moreover, the memory requirement is $\Theta(m + L.n)$, allowing one to take benefit of the possible graph sparseness. Such low complexities enable to deal with very large graphs containing several millions of nodes and edges. An extension of this approach is proposed to incorporate descriptive node features (*i.e.* attributes attached to each node) during classification.

Experiments on real-life databases show that the $\mathcal{D}$-walks approach outperforms three state-of-the-art approaches. The benefit of using bounded walks is empirically observed as better classification rates are obtained with walk length typically bounded between 2 and 10. We also show experimentally that the proposed approach easily scales up to large-scale problems thanks to its linear time complexity.

Our future work includes several extensions of the proposed approach. The interest of incorporating node features, as described in section 4, should be assessed

experimentally. A typical case study would be the labeling of protein-protein interaction network. The node features could include gene expression measurements coding for the corresponding proteins. The $\mathcal{D}$-walks method could be further extended to perform regression on a graph, that is to predict a continuous value on nodes rather than a class label. Finally, we would like to investigate the use of bounded random walks in item recommendation systems through collaborative filtering.

References

1. Callut, J.: First Passage Times Dynamics in Markov Models with Applications to HMM Induction, Sequence Classification, and Graph Mining. Phd thesis dissertation, Universite catholique de Louvain (October 2007)
2. Chebotarev, P., Shamis, E.: The matrix-forest theorem and measuring relations in small social groups. Automation and Remote Control 58(9), 1505–1514 (1997)
3. Chebotarev, P., Shamis, E.: On proximity measures for graph vertices. Automation and Remote Control 59(10), 1443–1459 (1998)
4. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Springer, Heidelberg (1983)
5. Macskassy, S.A., Provost, F.: Classi cation in networked data: A toolkit and a univariate case study. J. Mach. Learn. Res. 8, 935–983 (2007)
6. Newman, M.E.J.: A measure of betweenness centrality based on random walks. Social networks 27, 39–54 (2005)
7. Norris, J.R.: Markov Chains. Cambridge University Press, United Kingdom (1997)
8. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report, Computer System Laboratory, Stanford University (1998)
9. Rabiner, L., Juang, B.-H.: Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cliffs (1993)
10. Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT/Kernel 2003. LNCS (LNAI), vol. 2777, pp. 144–158. Springer, Heidelberg (2003)
11. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: Advances in Neural Information Processing Systems, vol. 14, pp. 945–952 (2002)
12. Tsuda, K., Noble, W.S.: Learning kernels from biological networks by maximizing entropy. Bioinformatics 20(1), 326–333 (2004)
13. Viger, F., Latapy, M.: Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 440–449. Springer, Heidelberg (2005)
14. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: ICML 2005: Proceedings of the 22nd international conference on Machine learning, pp. 1036–1043. ACM, New York (2005)
15. Zhou, D., Schölkopf, B.: Learning from labeled and unlabeled data using random walks. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 237–244. Springer, Heidelberg (2004)

Learning Bidirectional Similarity for Collaborative Filtering

Bin Cao[1], Jian-Tao Sun[2], Jianmin Wu[2], Qiang Yang[1], and Zheng Chen[2]

[1] The Hong Kong University of Science and Technology, Hong Kong
{caobin, qyang}@cs.ust.hk
[2] Microsoft Research Asia, 49 Zhichun Road, Beijing, China
{jtsun,i-jiwu,zhengc}@microsoft.com

Abstract. Memory-based collaborative filtering aims at predicting the utility of a certain item for a particular user based on the previous ratings from similar users and similar items. Previous studies in finding similar users and items are based on user-defined similarity metrics such as Pearson Correlation Coefficient or Vector Space Similarity which are not adaptive and optimized for different applications and datasets. Moreover, previous studies have treated the similarity function calculation between users and items separately. In this paper, we propose a novel *adaptive bidirectional similarity metric* for collaborative filtering. We automatically learn similarities between users and items simultaneously through matrix factorization. We show that our model naturally extends the memory based approaches. Theoretical analysis shows our model to be a novel generalization of the SVD model. We evaluate our method using three benchmark datasets, including MovieLens, EachMovie and Netflix, through which we show that our methods outperform many previous baselines.

1 Introduction

Personalized services are becoming increasingly indispensable nowadays ranging from providing searching result to product recommendation. Collaborative filtering aims at predicting the preference of items for a particular user based on the items previously rated by other users. Examples of successful applications of collaborative filtering include recommending products at Amazon.com[1], movies by Netflix[2], etc. Memory-based methods are a set of widely used approaches for collaborative filtering which are simple and effective [1]. They usually fall into two classes: user-based approaches [4,10] and item-based approaches [7,17]. To predict a rating for an item from a user, user-based methods find other similar users and leverage their ratings to the item for prediction, while item-based methods use the ratings to other similar items from the user instead.

[1] http://www.amazon.com
[2] http://www.netflix.com/

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 178–194, 2008.

Despite their success, memory-based methods suffer from several serious problems. First, missing data is a major problem in collaborative filtering, causing the so-called sparseness problem [24]. This is because there are usually millions of users and items in existence. But a single user can only rate a relatively small number of items. When the data are extremely sparse, it is difficult to find similar users or items accurately. Second, in memory based approaches, similar users and items are found by calculating a given similarity metric, including Pearson Correlation Coefficient (PCC) [16] and Vector Space Similarity (VSS) [4]. However, these metrics are not adaptive to the application domains and the data sets. Once given, they are not changeable. Third, the classical PCC and VSS have trouble in distinguishing different importance of items. To cope with these problems, many variations of similarity metrics, weighting approaches, combination measures, and rating normalization methods have been developed [9]. Although they can capture the correlation between users or items to a certain extent, for these adaptations to work, there is no consensus as to which choice of a technique is the most appropriate for a real world situation [9]. Finally, many previous studies in collaborative filtering consider the similarities between users and items separately. However, similarities between users and items in reality are interdependent and can be used to reinforce each other. Therefore, it would be more appropriate if the similarities between users and items be jointly learned automatically.

In this paper, we propose a novel model to learn both the item and user similarities together. Our model enables the similarity learning based collaborative filtering (SLCF). We show that the joint similarity learning can be formulated as a problem of matrix factorization with missing values. The learned similarities between users as well as items can be regarded as being influenced by some latent factors. Different from some previous latent factor models such as singular value decomposition (SVD) [25] and Aspect Model [11], our model provides a more flexible scheme that does not require the number of factors underling the user space and the item space to be the same. Theoretical analysis shows that our model corresponds to a novel generalization of the SVD model, thus allowing a number of nice theoretical properties to be inherited from SVD research. In addition, we provide algorithms for rating prediction with different strategies based on learned similarity. We evaluate our model using three widely used benchmark datasets, the MovieLens, EachMovie and Netflix data sets. Experiment results show that our method outperforms many of the well known baselines.

2 Related Work

In the past, many researchers have explored memory-based approaches to collaborative filtering. Many of them can be regarded as improving the definition of similarity metric [4,6,9,14]. A drawback of these methods is that these similarity metrics are not adaptive to different datasets or contain some parameters

needed to be tuned but not learned. Another set of related work consider how to utilize the user-based and item-based approaches together [14,21]. In [21], Wang et al. proposed a probabilistic fusion model to combine user-based method with item-based method. They found the fact that fusing all the ratings in the user-item matrix can help solve the data sparseness problem. However, they still estimate the user-based ratings and item-based ratings independently and omit the relationship between them. Ma et al. in [14] proposed a method to fill in the missing value first before prediction but it has the same drawback with [21]. One particular work which addressed learning similarity is in [12] where Jin et al. proposed an automatic weighting scheme for items. Their method aims at finding the optimal weights that can form a clustered distribution for user vectors in the item space by bringing similar users closer and dissimilar users far away. But they only considered the similarity weights for items, not users simultaneously.

Model based approaches do not predict ratings based on some ad-hoc heuristic rules, but rather, they are based on a *model* learned from the data using statistical and machine learning techniques. Viewed as a missing value prediction problem, collaborative filtering can also be solved through matrix factorization. SVD based approaches [3,20,25] can be regarded as latent factor models where the eigenvectors correspond to the latent factors. Users and items are mapped into a low dimensional space formed by the learned latent factors. Similar models also include [5,11]. A drawback of these models is that they all use the same latent factors to model users and items. An underlying assumption is that the numbers of latent factors that influence users and items are the same. Since a user may have diverse interests and an item may have multiple aspects, it is desirable to allow both items and users to be in a more flexible scheme. Si and Jin in [19] proposed a flexible mixture model for collaborative filtering. They are among the first to relax the restriction that users and items fall into the same classes. However, their probabilistic model regarded the ratings as discrete values. They also ignored the relation between ratings. As such, they did not consider scores of 3 and 2 to be closer to each other than scores of 5 and 1.

2.1 Memory-Based Collaborative Filtering

We review memory-based and SVD-based approaches for collaborative filtering (CF) in this and the next subsections. We construct a rating matrix R with rows representing users and columns representing movies. Since only part of the elements are known, we use X to denote the sparse matrix with elements known and use Y to denote the sparse matrix with elements we want to estimate. Both X and Y are subsets of rating matrix R. We define the problem of collaborative filtering as predicting the values in Y based on X.

User-based collaborative filtering predicts a target user u's interest in a test item m based on rating information from similar users.

$$r_{um} = \sum_{v \in C_u} s_{uv} r_{vm} \quad \text{for} \quad r_{um} \in Y \tag{1}$$

where r_{um} represents the rating for an item m from a user u and C_u is the set of nearest neighbors of the user u within which a user v has influence weight s_{uv} on u and s_{uv} can be calculated by normalizing Pearson Correlation Coefficient [16]. Hence $s_{uv} = PPC(u,v)/\sum_{w \in C_u} PPC(u,w)$, where

$$PCC(u,v) = \frac{\sum_{i \in R_u \cap R_v} (r_{ui} - \overline{r}_u) \cdot (r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i \in R_u \cap R_v} (r_{ui} - \overline{r}_u)^2} \cdot \sqrt{\sum_{i \in R_u \cap R_v} (r_{vi} - \overline{r}_v)^2}} \tag{2}$$

or Vector Space Similarity [4], so $s_{uv} = VSS(u,v)/\sum_{w \in C_u} VSS(u,w)$, where

$$VSS(u,v) = \frac{\sum_{i \in R_u \cap R_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in R_u \cap R_v} r_{ui}^2} \cdot \sqrt{\sum_{i \in R_u \cap R_v} r_{vi}^2}} \tag{3}$$

where R_u is the set of items rated by the user u.

Similar to user-based approach, we write item-based approaches as

$$r_{um} = \sum_{n \in C_m} s_{mn} r_{un} \quad \text{for} \quad r_{um} \in Y \tag{4}$$

where C_m is the set of nearest neighbors of the item m within which the item n has influence weight s_{mn} on m and s_{mn} can also be calculated using PCC or VSS as in the above equations.

2.2 SVD-Based Collaborative Filtering

Singular value decomposition (SVD)-based methods are also explored by many researchers for collaborative filtering [3,20,25]. SVD seeks a low-ranked matrix that minimizes the sum squared distance to the rating matrix R. Since most of the entries in R are missing, the sum-squared distance is minimized with respect to the partially *observed entries* of the rating matrix, which is X. So the loss function we optimize is

$$l = |I_X \odot (X - UV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2)$$

where $\odot$ stands for element-wise multiplication, $||\cdot||_F^2$ denotes the Frobenius norm, and I_X is the indicator function, with element $I_X(i,j)$ taking on value 1 if the user i rated the movie j, and 0 otherwise. U is a lower dimensional representation for users and V is a lower dimensional representation for items. The diagonal matrix Σ in traditional SVD is merged into U and V for simplicity. The last term is a regularization term which prevents the model from overfitting. *Unobserved entries* Y are then predicted by $Y = I_Y \odot (UV^T)$. The regularized SVD method has been shown to be successful in the competition of Netflix Prize [8,22].

Another adaptation of SVD-based method is using the EM algorithm to solve the missing value problem [25]. The basic idea is to iteratively estimate the

missing ratings and conduct SVD decomposition. However, since the matrix is no longer sparse in this approach, it quickly runs up against practical computational limits.

3 Learning Similarity Functions

We present our main contributions in this section. To begin with, we consider memory-based approaches in matrix form and extend it to one-directional similarity learning.

3.1 One-Directional Similarity Learning

Memory-based collaborative filtering methods are usually separated from model-based approaches and regarded as heuristic-based approaches [1]. In this paper we provide a novel way to model memory-based methods from matrix point of view.

Equation (1) can be written in a matrix form,

$$Y = \widehat{S}_1 X \tag{5}$$

where $\widehat{S}_1$ denotes the similarity matrix of row vectors corresponding to users with $\widehat{S}_1(u, v)$ defined by

$$\widehat{S}_1(u,v) = \begin{cases} s_{uv}, & v \in C_u, \quad (6) \\ 0, & \text{otherwise.} \quad (6') \end{cases}$$

Similar to the user-based approach, item-based methods can be represented in matrix form as

$$Y = X\widehat{S}_2 \tag{7}$$

where $\widehat{S}_2$ denotes the similarity matrix of the column vectors corresponding to items with $\widehat{S}_2(m, n)$ defined by

$$\widehat{S}_2(m,n) = \begin{cases} s_{mn}, & n \in C_m, \quad (8) \\ 0, & \text{otherwise.} \quad (8') \end{cases}$$

Noticing that X and Y are both subsets of the rating matrix R, Equations (5) and (7) can actually be seen as matrix reconstruction equations with respect to R. By replacing Y on the left side of the equation with R, we can obtain matrix factorization formulas for similarity matrix learning.

$$R = S_1 X \quad \text{and} \quad R = X S_2 \tag{9}$$

In the above formulas, the similarity matrices S_1 and S_2 are no longer predefined as in previous memory based approaches. Instead, they are the variables that can be learned from the data.

To reduce the number of parameters in a similarity matrix S, we can factorize S with $S = UV^T$. This means similarity matrices S_1 and S_2 can be non-symmetric since the influence between users may not be symmetric. Then we have a factorization problem with missing values,

$$R = UV^T X \tag{10}$$

If we ignore the missing values and replace R with X, this will lead to a new factorization problem

$$X = UV^T X \tag{11}$$

Matrix factorization in this form is also discussed in [23] where it is solved for document clustering.

If we assume the similarity matrices S_1 and S_2 are symmetric, we can reduce the number of parameters further and reformulate Equation (10) as

$$R = UU^T X \tag{12}$$

This is one-directional similarity learning model. In next subsection we extend it to bi-directional case.

3.2 Bi-Directional Similarity Learning

One-directional similarity learning considers users and items separately. In this section, we extend the learning problem to a bi-directional similarity learning problem that can learn the row and column similarities together. Recent studies [14,21] have found that the combination of user-based and item-based approaches can indeed boost the performance of collaborative filtering. However, these recently proposed methods still conduct user-based prediction and item-based prediction separately. In this section, we show how to integrate them together to take the advantage of both.

Based on previous subsection, a natural way to combine user-based and item-based approach can be stated as

$$r_{um} = \sum_{v,n} s_{uv} s_{mn} r_{vn} \quad \text{for} \quad r_{um} \in Y \tag{13}$$

In this formula, we extend the neighborhood to all users and all items. This indicates that all ratings are interconnected: the prediction for a target user and item can benefit from ratings of other users and items, and vice versa.

The above equation can be re-written in matrix form

$$Y = S_1 X S_2 \tag{14}$$

where S_1 and S_2 are also variables we need to learn. S_1 represents the row (user) similarity matrix and S_2 represents the column (item) similarity matrix. Similar

to one-directional similarity learning, we have a similarity learning problem in matrix factorization form.

$$R = S_1 X S_2 \tag{15}$$

With the assumption that the similarity matrices S_1 and S_2 are symmetric, the problem can be converted to

$$R = UU^T XVV^T \tag{16}$$

where U is a rank-K_U matrix and V is a rank-K_V matrix with K_U denoting the number of latent factors for users and K_V be the number of latent factors for items.

We can also extend the model to nonsymmetric similarity matrix, but in that case we have more parameters to learn. Symmetric assumption can significantly decrease the number of variables we need to learn. Another advantage of using this trick is that it guarantees the similarity matrix to be positive semi-definite naturally. Therefore, we still follow the symmetric assumption in this paper.

3.3 Algorithms for Bi-directional Similarity Learning

Now the loss function we are going to minimize is

$$l = ||I_X \odot (R - UU^T XVV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{17}$$

Since $I_X \odot R = I_X \odot X$, l can be converted to

$$l = ||I_X \odot (X - UU^T XVV^T)||_F^2 + \alpha(||U||_F^2 + ||V||_F^2)$$

The last term in l is a regularization term which prevents the model from overfitting. Let $E = I_X \odot (X - UU^T XVV^T)$, then the loss function is simplified by

$$l = ||E||_F^2 + \alpha(||U||_F^2 + ||V||_F^2) \tag{18}$$

We use gradient approaches to solve the minimization problem. We have the derivation of U and V in matrix form:

$$\frac{\partial l}{\partial U} = -2(EVV^T X^T U + XVV^T E^T U) + 2\alpha U \tag{20}$$

$$\frac{\partial l}{\partial V} = -2(E^T UU^T XV + X^T UU^T EV) + 2\alpha V \tag{20'}$$

There are a lot of gradient based algorithms which have been developed for optimization problems such as conjugate gradient [15] and SMD [18]. In this paper we use adaptive gain gradient decedent algorithm [2] to minimize the loss function. The algorithm is described in Algorithm 1. The advantage of adaptive gain gradient decedent algorithm includes easy implementation and fast convergence speed.

Algorithm 1. Bi-directional Similarity Learning using Adaptive Gain

Input: training data X, parameters μ, K_U, K_V and T
Output: U and V
Initialization: Random initialize U and V
FOR $t = 1$ **TO** T**:**
Update U**:** $U^{(t+1)} = U^{(t)} - \eta_U^{(t)} \odot \frac{\partial l}{\partial U}^{(t)}$
Update V**:** $V^{(t+1)} = V^{(t)} - \eta_V^{(t)} \odot \frac{\partial l}{\partial V}^{(t)}$
Update η_U**:**
$\eta_U^{(t)} = \eta_U^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_U^{(t)} \odot \frac{\partial l}{\partial U}^{(t-1)} \odot \frac{\partial l}{\partial U}^{(t)})$
Update η_V**:**
$\eta_V^{(t)} = \eta_V^{(t-1)} \cdot \max(\frac{1}{2}, 1 + \mu \cdot \eta_V^{(t)} \odot \frac{\partial l}{\partial V}^{(t-1)} \odot \frac{\partial l}{\partial V}^{(t)})$

Another point we should notice is that although the similarity matrices S_1 and S_2 are large and dense, we can avoid computing them in the algorithm by carefully choosing the order of matrices multiplication.

3.4 Relation to SVD

In this section, we discuss the relation between our model and SVD model.

Theorem 1. *If we disregard the missing data and require that the ranks of U and V are the same, SVD is the solution to $X = UU^TXVV^T$.*

Proof. Suppose that $X = U\Sigma V^T$. By plugging it into UU^TXVV^T, we obtain $UU^TXVV^T = UU^TU\Sigma V^TVV^T = U\Sigma V^T = X$.

The equivalence of our model and SVD models can be established under the condition that there are no missing values and U and V have equal ranks. However, when there are missing values, the two models are not equivalent anymore even when we have $K_U = K_V = K$. We can see this point in the experiment part again.

Another difference between our model and SVD is seen from the rank of approximation matrix. SVD seeks the optimal rank-K approximation to the original matrix. But in our problem, we are not explicitly given rank restriction of the reconstructed matrix. The rank of reconstructed matrix is determined by the ranks of S_1, S_2 and X itself.

From the dimension-reduction point of view, SVD seeks a K dimensional space for row vectors and column vectors. However, in our model, we look for two different ranked spaces for row vectors and column vectors. Therefore, our model can also be regarded as bi-dimension-reduction-based method for row vectors and column vectors with different dimensions. We also can find the relation between the two spaces as two basis sets satisfy the following equation

$$U \cdot B_1 = B_2 \cdot V \tag{21}$$

where the users' basis $B_1 = U^TXVV^T$ and the items' basis $B_2 = UU^TXV$.

4 Rating Prediction Based on Bidirectional Similarity Learning

Different strategies can be used for collaborative filtering based on our learned similarity. In this section, we discuss three types of similarity learning based collaborative filtering strategies.

4.1 Matrix Reconstruction Strategy

Model-based approaches keep the user profiles in a more compressed data structure than memory based methods. The prediction for a user's interests is based on the user's profile that is learned during a training process. In our model, the user u's profile corresponds to row u in matrix U denoted by U_u and the item i's profile corresponds to column i in V, i.e. V_i^T. With our learned model, we predict a rating to the item i by the user u,

$$r_{ui} = \sum_{v,j} s_{vu} s_{ij} r_{vj} = U_u U^T X V V_i^T \quad \text{for} \quad r_{ui} \in Y$$

This can be done when both u and i show up in the training data X. We refer to this prediction strategy as matrix reconstruction strategy for SLCF (R-SLCF).

Matrix reconstruction strategy for collaborative filtering has the new user and new item problem. It can only predict the rating for existing users and items during training process. A naive solution to this problem is to retrain the whole new dataset and then make prediction for the new users and items. This procedure is clearly too time-consuming and often infeasible. In the next sub-section, we will use another strategy to solve this problem.

4.2 Projection Strategy

In this section, we discuss projection based strategy P-SLCF in our new framework which can bring new users and items into the model without retraining on the whole dataset. The key issue is how to introduce new users and items into the previous model and predict ratings for these new users and items based on previous models.

Suppose that there are some new users who arrive with new rating information $\widehat{Y}$ and $\widehat{Y}$ is to be included into the previous user rating matrix X. Then we have a new rating matrix with $X' = \begin{pmatrix} X \\ \widehat{Y} \end{pmatrix}$. Let U_Y be a representation of new users. Hence we have

$$\widehat{Y} = U_Y \cdot U^T X V V^T = U_Y \cdot B_1 \tag{22}$$

By solving the above linear equation, we find U_Y with

$$U_Y = \widehat{Y} \cdot ((I_Y \odot B_1) \cdot (I_Y \odot B_1)^T + \lambda \mathbb{I})^{-1} \tag{23}$$

where $\mathbb{I}$ is identity matrix. We can regard the user as being projected to a lower dimensional space spanned by the matrix B_1. Then, all new users are projected

into this space. The last term $\lambda\mathbb{I}$ is introduced to guarantee that the inverse operation is more stable [13].

Similar to adding new users, we can consider the new items as being projected to a lower dimensional space spanned by B_2. Suppose that there are some new items that arrive with new rating information $\widehat{Y}$ and $\widehat{Y}$ is included into the previous user rating matrix X to give $X' = (X, \widehat{Y})$. We can update V_Y by

$$\widehat{Y} = UU^T XV \cdot V_Y^T = B_2 \cdot V_Y^T \tag{24}$$

$$U_Y = \widehat{Y} \cdot ((I_Y \odot B_2) \cdot (I_Y \odot B_2)^T + \lambda\mathbb{I})^{-1} \tag{25}$$

Then similar to R-SLCF, we can predict the rating by

$$R_Y = U_Y U^T XVV_Y^T$$

Although we need to calculate inverse of matrices in projection based strategy, but since the matrices are of rather small scale and can be computed efficiently.

4.3 Improved Memory-Based Strategy

Memory-based methods can also be adapted to use our learned similarity. The idea is to use the learned similarity matrices S_1 and S_2 to find the nearest neighbors. Then we can use the memory based methods for prediction. We refer to this strategy as M-SLCF. This strategy is especially helpful for comparing our learned similarity with the user-defined similarity such as PCC. We show the results of comparison in Section 5.3.

5 Experiment

In this section, we will introduce data sets, evaluation metric and experiment results of our similarity learning-based collaborative filtering. In Section 5.3, M-SLCF is used and in Section 5.4, P-SLCF is used for comparison purpose. In other parts, R-SLCF is used for experiments.

5.1 Datasets

Three benchmark datasets are used in our experiments.

- MovieLens[3] is a widely used movie recommendation dataset. It contains 100,000 ratings with scale 1-5. The ratings are given by 943 users on 1,682 movies. The public dataset only contains users who have at least 20 ratings.
- EachMovie[4] is another popular used dataset for collaborative filtering. It contains 2,811,983 ratings from 72,916 users on 1,628 movies with scale 1-6.

[3] http://www.grouplens.org/

[4] http://www.cs.cmu.edu/ lebanon/IR-lab.htm

Table 1. Optimal K_V Given K_U

K_U	5	6	7	8	9	10	11	12	13	14	15
Opt K_V	14	14	14	14	12	10	8	8	8	6	5
MAE	0.7611	0.7606	0.7606	0.7607	0.7604	0.7606	0.7605	0.7603	0.7606	0.7607	0.7608

Table 2. Optimal K_U Given K_V

K_V	5	6	7	8	9	10	11	12	13	14	15
Opt K_U	13	13	12	12	12	11	9	9	9	7	6
MAE	0.7608	0.7607	0.7606	0.7603	0.7606	0.7606	0.7606	0.7604	0.7607	0.7606	0.7610

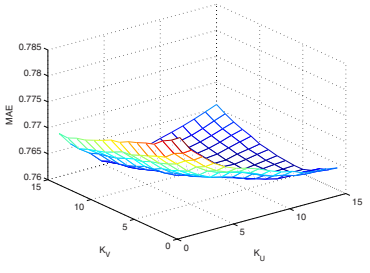


Fig. 1. MAE surface of R-SLCF on MovieLens dataset. Numbers of user factor (K_U) and item factor (K_V) are varied simultaneously.

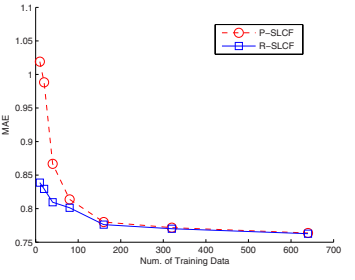


Fig. 2. Comparison of P-SLCF and R-SLCF. Evaluated by MAE on Movielens with 200 user as test data, $K_U = K_V = 10$ for SLCF.

– Netflix[5] is a pubic dataset used in Netflix Prize competition. It contains ratings from 480,000 users on nearly 18,000 movies with scale 1-5. In this paper, we use a subset of 367,348 ratings from 5,000 users and 2,000 movies for our experiments.

5.2 Evaluation Metrics

In this paper, we use Mean Absolute Error (MAE) for experiment evaluation.

$$MAE = \frac{\sum_{u,m} |r_{um} - \widehat{r}_{um}|}{N}$$

where r_{um} denotes the rating of the user u for the item m, and $\widehat{r}_{um}$ denotes the predicted rating for the item m of the user u. The denominator N is the number of tested ratings. Smaller MAE score corresponds with better prediction.

[5] http://www.netflixprize.com

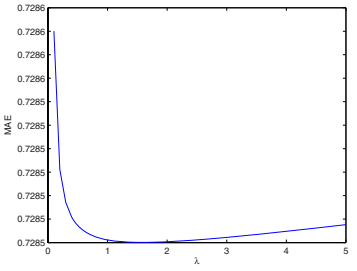


Fig. 3. Influence of parameter λ to the performance of P-SLCF

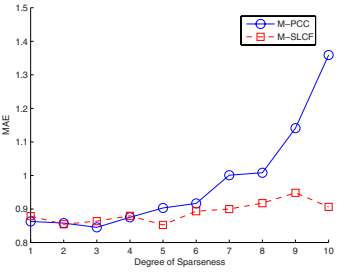


Fig. 4. Comparison of PCC and SLCF on similarity with different degree of sparseness. Evaluated using MAE on EachMovie with 50 nearest neighbors

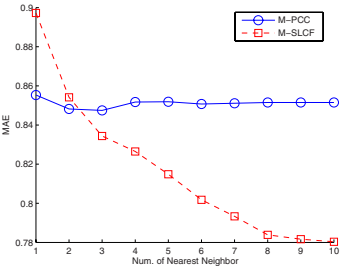


Fig. 5. Comparison of PCC and SLCF on similarity with different number of nearest neighbors. Evaluaed using MAE on EachMovie with sparseness degree 5.

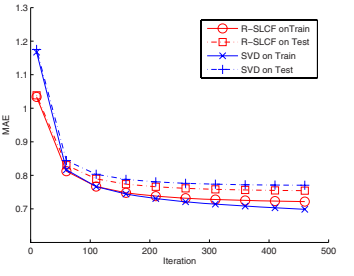


Fig. 6. Converge curves of R-SLCF and regularized SVD. Evaluated by MAE on Movielens with parameter $K = 10$ for regularized SVD, $K_U = K_V = 10$ for R-SLCF.

5.3 Empirical Study of our Approach

Impact of K_U and K_V. Two important parameters of our SLCF methods are the user similarity matrix rank K_U and the item similarity matrix rank K_V. In this experiment, we run experiments on MovieLens dataset to study the impact of K_U and K_V. Figure (1) shows the three dimensional MAE surface with K_U and K_V being changed simultaneously. We find that the best prediction result is achieved when K_U and K_V are neither too small nor too large. Table (1) shows the best K_V for given K_U and Table (2) shows the best K_U for given K_V. An interesting observation is that most of the best prediction results are achieved when $K_U + K_V \approx 20$. This means that the inherent information conveyed by latent user factors and item factors are complementary to each other. When fewer user factors are available, more item factors are required to characterize the inherent structure of rating matrix, and vice versa. From the MAE surface of Figure (1), the best result is obtained when both user and item factors are

considered ($K_U = 12$, $K_V = 8$). This verifies our motivation that user and item spaces should be modeled with different numbers of factors. Another parameter in our model is α which controls the balance between prediction error on training data and model complexity. After testing on different values, we use $\alpha = 0.0001$ in our experiments.

The Difference of R-SLCF and P-SLCF. Since it is costly to retrain the model when new users or items come, we provide the P-SLCF algorithm in Section 4.2. In this experiment, we compare the accuracy of prediction by R-SLCF and P-SLCF. Figure (2) shows the comparison results on MovieLens. In this experiment, we use 200 users as testing data. When training users are very few, P-SLCF is not as good as R-SLCF. But as the number of training users increases, the performances of P-SLCF and R-SLCF become very close.

An important parameter in P-SLCF is λ. Figure (3) shows the influence of λ to the prediction accuracy. After testing different values of λ, we find that $\lambda = 1$ to be a good choice which we use in our experiments.

Impact of Data Sparseness. In this sub-section, we show experiments on the impact of data sparseness on similarity learning using M-SLCF. For comparison purpose, we also use the predefined similarity PCC (Equation (2)) for selecting neighbors which we refer to as M-PCC. In both cases, Equation (1) with equal weights for neighbors is used for making predictions.

We first filter the EachMovie dataset by keeping the users who have rated different number of movies (from less than 50 to less than 5 in this experiment). In this way, we construct datasets with different degree of sparseness. We use user based method with neighbors found by SLCF and compare it with PCC. When the data are not that sparse, PCC can do good job in finding nearest neighbors. However, when the degree of sparseness increases, it does not work anymore. In Figure (4), we can clearly see that SLCF is able to find more accurate neighbors with the degree of sparseness increased. Figure (5) verifies our conclusion from the other side. It shows how SLCF and PCC perform with different number of nearest neighbors. We can see that PCC is good at finding the most similar users but SLCF has the advantage of finding the potentially similar users. That is, we can improve the recall of finding similar users. Therefore, when more nearest neighbors are used, our model performs much better.

5.4 Comparison with Other Approaches

The baselines we use include user-based method using PCC, item-based method using PCC and regularized SVD method. We also compare our method with another recent proposed state-of-the-art method [21] which also fusions the similarities of users as well as items. Although we also conduct experiments on Netflix dataset, our results are not comparable with the top results on the leaderboard since they are hybrid methods. We should notice regularized SVD, which is one of the best algorithms in the Netflix Prize competition [8], is also included in our baselines.

Table 3. MAE comparison of R-SLCF with SVD for different K. For R-SLCF1 we require $K_U = K_V = K$. For R-SLCF2 we require $K_U + K_V = 2K$.

Dataset	K	SVD	R-SLCF1	R-SLCF2
MovieLens	K=5	0.7665	**0.7534**	**0.7534**
	K=10	0.7676	0.7517	**0.7516**
	K=15	0.7785	0.7533	**0.7523**
	K=20	0.7906	0.7554	**0.7532**
EachMovie	K=5	0.8023	0.7902	**0.7901**
	K=10	0.8272	0.7855	**0.7845**
	K=15	0.8317	0.7920	**0.7912**
	K=20	0.8127	0.7932	**0.7920**
Netflix	K=5	0.7557	0.7505	**0.7501**
	K=10	0.7640	0.7490	**0.7480**
	K=15	0.7737	0.7498	**0.7498**
	K=20	0.7835	0.7571	**0.7569**

Table 4. MAE comparison of R-SLCF with memory-based method and item-based method. $N = 30$ means only users with ratings no larger than 30 are included.

Dataset	#Rating	I-based	U-based	R-SLCF
MovieLens	N=30	1.0936	0.8785	**0.8418**
	N=40	0.9587	0.8527	**0.8113**
	N=50	0.9144	0.8451	**0.8104**
	N=60	0.8648	0.8239	**0.8056**
EachMovie	N=30	1.7238	0.9919	**0.9347**
	N=40	1.6437	0.9908	**0.9297**
	N=50	1.7792	0.9836	**0.9338**
	N=60	1.6656	0.9886	**0.9327**
Netflix	N=30	0.9568	0.8804	**0.7974**
	N=40	0.8647	0.8390	**0.7782**
	N=50	0.8293	0.8114	**0.7672**
	N=60	0.7934	0.7774	**0.7439**

Table 5. Compare with results of SF on MovieLens

Num. of Training Users	100			200			300		
Num. of Ratings Given	5	10	20	5	10	20	5	10	20
P-SLCF	**0.838**	**0.770**	**0.771**	**0.799**	**0.768**	**0.763**	**0.787**	**0.753**	**0.739**
SF	0.847	0.774	0.792	0.827	0.773	0.783	0.804	0.761	0.769

Comparison with SVD-Based Approaches. Since our model is similar to SVD, in this section, we carefully compare our model with the regularized SVD model we introduced in Section 2.2 in different aspects. Figure (6) shows the convergence curves of our approach compared with regularized SVD. In this experiment, we use the same optimization algorithm (adaptive gain) with the same initial point[6] for U and V to run the algorithms and tune the best step length for each algorithm. We can see our approach converges faster than regularized SVD and finds better solution. It is also worthy to notice that in the last several iterations regularized SVD has smaller MAE on training data but larger MAE on test data when compared with R-SLCF. This indicates regularized SVD is more likely to be overfitting than our model. This may be due to that regularized SVD requires a strict rank-K approximation but we do not.

Table (3) shows a performance comparison of our model and regularized SVD model with various Ks. In this experiment, R-SLCF uses the same number of variables with regularized SVD for the fair of comparison. We can see our method clearly outperforms regularized SVD model. This experiment also indicates that

[6] Although the initial points are the same, the initial performance can be different.

when there are missing values our model is different from regularized SVD even $K_U = K_V$.

Comparison with Memory-Based Approaches. We compare our method with user-based(U-based) and item-based(I-based) approaches with results shown in Table (4). The experiment is carried out with different sparseness condition with $N = 30$ meaning only users who have ratings less than or equal to 30 are used. From this table we can see that our method clearly outperforms the baselines.

We also compare our method with another stat-of-the-arts algorithm Similarity Fusion (SF) [21] which also utilizes both user side and item side information. The difference between our approach and SF is that the similarities used in our algorithm is automatically learned rather than defined heuristically. To compare with their algorithm, we followed the exactly same experiment settings in the paper. Then, for the performance of their method, we quote their results from their publication. We can see that our approach outperforms SF significantly.

6 Conclusion and Future Work

We proposed a novel model learning user and item similarities simultaneously for collaborative filtering. We showed that our model can be regarded as a generalization of SVD model. We developed an efficient learning algorithm as well as three prediction strategies. The experiments showed our method could outperform baselines including memory-based approaches and SVD.

For future work, we plan to develop more efficient algorithms to learn our model in larger scale datasets. We also plan to relax the symmetry assumption. Although it brings more variables to learn, it is a more reasonable assumption. Although focused on collaborative filtering in this paper, our model is very general for sparse data which has matrix form. Therefore, we plan to apply our model to other kinds of data sets and tasks such as document clustering.

Acknowledgments

Bin Cao and Qiang Yang are supported by a grant from MSRA (MRA07/08. EG01). We thank the anonymous reviewers for their useful comments.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE TKDE 17(6), 734–749 (2005)
2. Almeida, L.B., Langlois, T., Amaral, J.D., Plakhov, A.: Parameter adaptation in stochastic optimization. On-line learning in neural networks, 111–134 (1998)

3. Brand, M.: Fast online svd revisions for lightweight recommender systems. In: Proc. of SIAM ICDM (2003)
4. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14^{th} Conf. on UAI, pp. 43–52 (1998)
5. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proc. of the 25th SIGIR, pp. 238–245. ACM, New York (2002)
6. Delgado, J.: Memory-based weightedmajority prediction for recommender systems. In: ACM SIGIR 1999 Workshop on Recommender Systems (1999)
7. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM TOIS 22(1), 143–177 (2004)
8. Funk, S.: Netflix update: Try this at home (December 2006), `http://sifter.org/~simon/journal/20061211.html`
9. Herlocker, J., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf. Retr. 5(4), 287–310 (2002)
10. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proc. of the 22nd SIGIR, pp. 230–237. ACM, New York (1999)
11. Hofmann, T.: Latent semantic models for collaborative filtering. ACM TOIS 22(1), 89–115 (2004)
12. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proc. of the 27th Annual International ACM SIGIR
13. Kirsch, A.: An introduction to the mathematical theory of inverse problems. Springer, New York (1996)
14. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: Proc. of the 30th SIGIR, pp. 39–46. ACM, New York (2007)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C, 2nd edn. The art of scientific computing. Cambridge University Press, Cambridge (1992)
16. Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proc. of ACM 1994 Conf. on CSCW (1994)
17. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. of the 10th international Conf. on WWW, pp. 285–295 (2001)
18. Schraudolph, N.N.: Local gain adaptation in stochastic gradient descent. Technical Report IDSIA-09-99, 8 (1999)
19. Si, L., Jin, R.: A flexible mixture model for collaborative filtering. In: Proc. of the Twentieth ICML (2003)
20. Vozalis, M.G., Margaritis, K.G.: Using SVD and demographic data for the enhancement of generalized collaborative filtering. Inf. Sci 177(15) (2007)
21. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proc. of the 29th SIGIR, pp. 501–508. ACM, New York (2006)
22. Wu, M.: Collaborative filtering via ensembles of matrix factorizations. In: Proc. of KDD Cup and Workshop (2007)
23. Xu, W., Gong, Y.: Document clustering by concept factorization. In: Proc. of the 27th SIGIR, pp. 202–209. ACM, New York (2004)

24. Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proc. of the 28th SIGIR, pp. 114–121. ACM, New York (2005)
25. Zhang, S., Wang, W., Ford, J., Makedon, F., Pearlman, J.: Using singular value decomposition approximation for collaborative filtering. In: Proc. of the Seventh IEEE CEC, pp. 257–264 (2005)

Bootstrapping Information Extraction from Semi-structured Web Pages

Andrew Carlson[1] and Charles Schafer[2]

[1] Machine Learning Department, Carnegie Mellon University, Pittsburgh PA 15213, USA
[2] Google, Inc., 4720 Forbes Avenue, Pittsburgh PA 15213, USA

Abstract. We consider the problem of extracting structured records from semi-structured web pages with no human supervision required for each target web site. Previous work on this problem has either required significant human effort for each target site or used brittle heuristics to identify semantic data types. Our method only requires annotation for a few pages from a few sites in the target domain. Thus, after a tiny investment of human effort, our method allows automatic extraction from potentially thousands of other sites within the same domain. Our approach extends previous methods for detecting data fields in semi-structured web pages by matching those fields to domain schema columns using robust models of data values and contexts. Annotating 2–5 pages for 4–6 web sites yields an extraction accuracy of 83.8% on job offer sites and 91.1% on vacation rental sites. These results significantly outperform a baseline approach.

1 Introduction

This work addresses the problem of extracting structured records from semi-structured web pages with no human supervision required for the target web site. Semi-structured web pages are human-readable renderings of database entries. Familiar examples of semi-structured web page domains include books for sale, properties for rent, or job offers. We develop techniques that learn extraction models applicable to an entire domain of web sites from just 2–5 annotated pages from each of 4–6 web sites within that domain. The end result is a high-accuracy system that can be applied to many web sites within a domain without any human annotation of those sites.

In this work, we extract data from *detail pages* of web sites. These are pages which correspond to a single data entity, and which render various attributes of that entity in a human-readable form. An example detail page is shown in Fig. 1. While we focus on the setting where a page contains one record in this work, our methods could be easily adapted to the case where multiple data records exist on one page through use of existing data record detection algorithms (*e.g.* [1]).

Extracting structured records from semi-structured web pages is an important problem because a great deal of information on the internet is presented in this form. Moreover, the resulting structured records are particularly valuable to

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 195–210, 2008.

Fig. 1. Part of an example detail page from the vacation rental domain. The page is about a particular house and displays attributes of that house such as its location, occupancy limit, and number of bathrooms.

downstream learning or querying systems because of their attribute/value structure and conformance to a fixed domain schema. Solving this problem without requiring human supervision for each target web site has been an understudied topic. Most previous work on this problem has required human supervision by requiring a person to either annotate example web pages from each target web site [2,3,4] or map the extracted data fields for each target site to columns in a domain schema [5,6,7]. The work that did not require supervision for each target site was not robust to typical variations on semi-structured web sites, such as cases where data fields do not have a label and do not match some unique pattern [8]. Our method only requires human supervision for a few web pages from a few sites in the target domain, allowing minimally supervised extraction from potentially hundreds or thousands of other sites within the same domain.

Our basic approach is to generalize extraction on a per-domain basis (e.g., for vacation rental sites). In taking on a new domain, first a human decides what the domain schema should be: that is, what schema columns are interesting and are present on many sites in the domain. Next, we annotate a small number of pages (2–5) for each of a few (4–6) web sites in the domain. These are provided as training data to an existing supervised extraction system[1], which learns a site-specific extraction model for each of the 4–6 sites; each model is then applied to all of the detail pages on the corresponding web site, yielding 4–6 machine-annotated sets of detail pages to be used as training data for our model. Finally, via the method presented in Sect. 3, we model both page contexts and values for each column in the domain schema, learning how to perform extraction for

[1] A supervised information extraction system making use of the partial tree alignment algorithm from DEPTA [7].

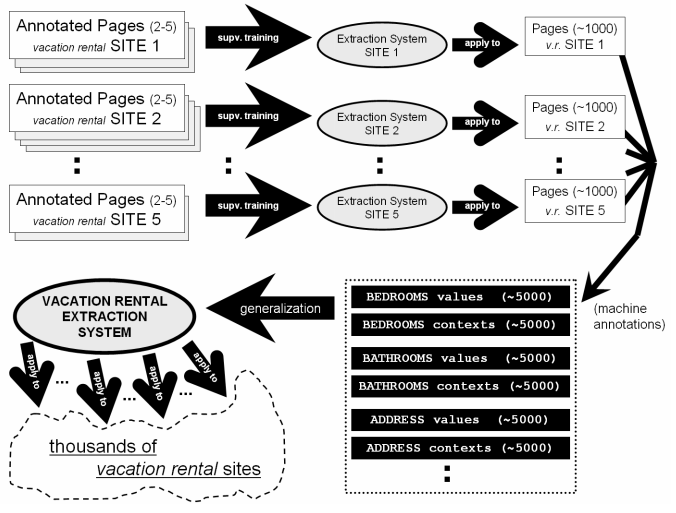


Fig. 2. A high-level view of learning under our system. By annotating 2–5 pages for 5 vacation rental sites, we train an extraction system that can be applied to thousands of other vacation rentals sites with no additional supervision required.

new vacation rental sites[2] with no additional human labeling effort. Figure 2 illustrates this learning scheme.

The experiments described in Sect. 4 measure an extraction accuracy of 83.8% on a previously unseen domain (job offer sites, which were all completely held out until evaluation time, providing a fair evaluation of how the system would perform in practice on other domains) with no direct supervision on the target sites being labeled. For previously unseen web sites in the vacation rentals domain (other vacation rental sites were used in algorithm development) we observed an accuracy of 91.1%. These results show a strong improvement over baseline accuracies of 61.8% for job sites and 65.8% for vacation rental sites, which were obtained using a logistic regression model. Other performance measures, also discussed in Sect. 4, indicate that use of our system for aiding human annotators is extremely promising, allowing a very small amount of effort to further increase accuracies.

2 Related Work

There has been much previous work on supervised learning of models for extraction of structured records from semi-structured web sites [2,3,4]. Such work

[2] Sites amenable to document object model tree alignment (implying that site-internal templates are used for automatic page generation), and making only trivial use of Javascript. We have observed that roughly 50% of vacation rental sites can be expected to meet these criteria.

requires a user to annotate training pages on a target web site, and then learns to extract data from the rest of the site. This process is labor-intensive, as it requires a new, separate labeling for every site.

Other previous work does not require a user to fully label example pages on the target web site, but does require manual labeling of the output. These approaches extract data fields by learning a common template for a page or site. Then, the user typically selects a subset of the output data fields and labels them according to the domain schema. Examples of such work are IEPAD [5], OLERA [6], and DEPTA [7]. This can be a labor-intensive process, because web sites tend to have many irrelevant fields, and thus the user must select from many more data fields than there are schema columns (web sites in our evaluation had an average of 20 data fields and 7 schema columns). Our work extends these methods, providing an automatic labeling of the extracted data by automatically mapping data fields to schema columns. This paper uses the partial tree alignment method of DEPTA to detect data fields, but many other template-finding algorithms, such as the repetitive pattern finding method of IEPAD, or the string editing distance method of OLERA, could be used in its place. Thus, this paper fills an important gap by allowing use of any one of a number of previous methods for detecting data fields on a web site, and providing minimally supervised selection of relevant fields and mapping to schema columns.

An exception to the requirement of user labeling of detected data fields is the DeLa system [8], which uses heuristics to automatically label extracted data fields. The heuristics depend on matches between the name of a schema column and labels present on a web page, or on data conforming to specific formats (e.g. dates or prices). There are many common cases where the DeLa system cannot label data fields. For example, the DeLa heuristic misses a common case where a data field does not have a label (for example, the job title field on most job sites is a heading with no prompt or label). Also, the DeLa heuristic would be confused by sites with multiple fields of the same data type (for example, the *bedrooms*, *bathrooms*, and *maximum occupancy* fields in vacation rentals). Our methods are much more robust to the variations typical on semi-structured web sites, and can handle these cases.

We label extracted data fields by comparing them to data fields on other annotated sites from the same domain. The only other previous work that we found that uses this approach to label semi-structured data is that of Golgher et al. [9]. This work bootstrapped onto new sites using examples of structured records from other web sites. These example records discarded the page context in which values appeared on the training web pages, and thus their system could not generalize across web sites based on context (our results in Table 1 demonstrate the high value of contextual features for the vacation rentals domain). Also, their model matched data values only at the token level, discarding useful textual features such as matches of short subsequences of characters.

Our method is partly inspired by techniques for schema matching in the database community. Specifically, the authors in [10] match elements in different database schemas by learning a model of schema columns which combines the

output of several base classifiers using a logistic regression meta-classifier. Our work applies a similar technique to a different problem. We match data fields on different web sites to a domain schema, rather than working with database fields. We also use different types of features, and we take advantage of the availability of distributions of data values by comparing frequencies of different values.

Freitag used a multi-strategy learning method where regression models were learned to map confidence scores to probabilities for various information extraction strategies. The probabilities were then combined in a Bayesian manner to make final predictions [11]. Our work uses a simpler, more direct method of combining models— we use a regression model for each schema column to combine confidence scores from our strategies (classifiers for each schema column that use some feature type to classify data fields). Additionally, our method classifies site-level data fields rather than individual data values, allowing better decisions to be made because more information is available at the site level.

3 Methods

In this paper, a *domain schema* is a set of attributes usually present in a record for the selected domain, while a *schema column* is a single attribute within that schema. A *detail page* of a web site is a page that corresponds to a single data record. A *data field* on a web site is a location within a template for that site (in this work, a node within a DOM tree template for that site), and a *data value* is an instance of that data field on a single detail page from that site. Data values appear in *contexts* which in this work are the text preceding a data value.

The central idea of our method is that we can bootstrap from a few training sites by building a model of data values and contexts for each schema column in our target domain schema, and then apply that model to many more sites with no further human effort. As training data, we require a collection of detail pages from a small number of web sites within the target domain that have been automatically labeled using a supervised wrapper induction algorithm according to some domain schema[3]. At test time, our method takes a collection of detail pages from a previously unseen target site[4] in the same domain. The goal is to annotate these pages according to the domain schema, identifying where schema columns are expressed in spans of text on the pages. We do this by first detecting potential data fields on the pages, and then classifying the data fields using a model learned from the training data.

[3] As mentioned in Sect. 1 acquiring this training data requires annotating 2–5 pages from 4–6 web sites to train a supervised system for each site, and then extracting records from the rest of the pages from the training sites.

[4] Our work assumes the availability of automatic methods for identifying such pages. To implement this ability we could use an approach that clusters and classifies pages, such as [12].

More specifically, we use the following method:

1. On the target site, create a template for the detail pages that identifies aligned regions across pages that look like potential data values. We consider these regions to be potential data fields. Our method for identifying these data fields is described in Sect. 3.1.
2. Based on the machine-annotated training data from a few training sites, label the detected data fields on the target site with a score that indicates how likely they are to be instances of each schema column in a domain schema. This is described in Sect. 3.2 below.
3. Using these scores, either automatically annotate the target site, or else give recommendations to aid a human annotator (these are alternate uses of our output). Details of this procedure are given in Sect. 3.3.

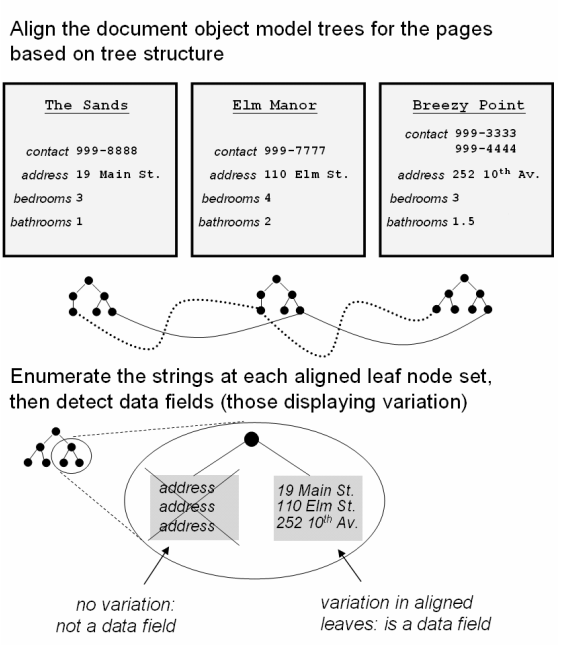


Fig. 3. Alignment of document object model trees and subsequent detection of data fields

3.1 Detecting Data Fields

We detect data fields across the pages on the target site by using the Partial Tree Alignment algorithm, which is part of the DEPTA system [7]. The data field detection is done in an unsupervised manner as follows. First, document object model (DOM) trees for the detail pages on a site are aligned. Next, the

strings occurring at aligned nodes are enumerated and inserted into a set. Any of these sets containing more than one element, then, corresponds to a region in the "template" for detail pages that exhibits variability across pages. Such regions (aligned DOM tree nodes exhibiting variability) are taken as the candidate data fields for the site. This process is illustrated in Fig. 3. Data fields that aligned to fewer than half of the pages on a site were filtered out, because these were typically not interesting data fields. Such filtered data fields tended to be different paragraphs of free text which the tree alignment algorithm occasionally decided to align. In effect, this filtering step ignores free text data fields which do not occur on all pages of the site and are not interesting data fields.

3.2 Classifying Data Fields

For each data field on the target site, we assign a score representing its correspondence to each schema column in the domain schema. A high score indicates a high degree of confidence that a data field should be mapped to that schema column. Informally, we find data fields on the target site that have data values that are similar to the values observed in the training data for a schema column, and we also want to find data fields that appear in page contexts that are similar to contexts from the training data.

To compute the score for a data field f and a schema column c, an obvious method to try is to extract a set of features for the data field, and use a classifier to map data fields to schema columns, where we train the classifier with our training sites. However, we discuss below in the 'Motivation' subsection that this method is impractical given the number of different textual features we must use (tens of thousands) and the number of training examples we have (roughly one hundred), so that a single classifier will tend to overfit and give irrelevant features high weights. Instead, we use a model that computes, for K different *feature types*, how similar the feature values observed for that data field are to the feature values observed in the training data for that schema column. This yields K different subscores, one for each feature type. We then combine those scores using a regression model that outputs a final score. The intuition is that for each schema column, different types of features matter, and that comparing distributions of observed features is much less noisy than singling out individual features in our model.

In the rest of this subsection, we describe the types of features that we use to represent data fields and schema columns. We then give details about our method for comparing different distributions of values of these features, describe the regression model that combines the similarity scores for each feature type, and give details on how we train the model. We end the subsection with further discussion of the motivation behind our model.

Feature Types. Our method uses four different *feature types*. A feature type is a family of features, such as 'lowercase tokens occurring in data values of a

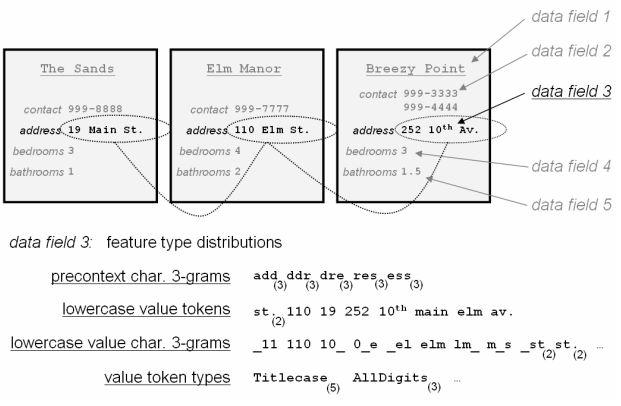


Fig. 4. Feature type distributions are created for data fields in aligned document object model trees. This is done for both training and test sites, so that the resulting distributions can be compared. In the figure, counts exceeding 1 are indicated by subscripts.

data field.' Our method combines different types of features because there can be different indicators of the correct label for a data field. We might sometimes have strong clues from the text before a data field (e.g. it might always say 'Bedrooms:', indicating that the data field holds the value for *bedrooms*). We might also have strong clues from the text inside a data field (e.g. 'Software Engineer Level I' indicating a *job title*). See Fig. 4 for an illustration of the feature type distributions for an example data field. We use the following feature types:

- **Precontext character 3-grams**: We extract character 3-grams from the non-tag text preceding a data field in HTML. Web sites often abbreviate or use different forms of labels, such as "Bedrooms", "Beds", "Bedrms". Character 3-grams can capture these variations.
- **Lowercase value tokens**: We tokenize data values by whitespace, and convert them to lowercase. Many of the schema columns that occur in semi-structured data have values that come from a limited vocabulary.
- **Lowercase value character 3-grams**: We extract character 3-grams of the data values. Many types of data can be abbreviated, or use similar but not identical words, or use special symbols or codes, all of which can be better captured using character 3-grams.
- **Value token types**: We categorize tokens in data values into general types (e.g. all caps, all digits, title case). These features are helpful for addresses, unique IDs, or other schema column types with a mix of token types.

Comparing Distributions of Feature Values. We compare distributions of features observed in training data to distributions observed in data fields to be labeled on the target site. This approach uses all of the data, including relative frequencies of features, to make its predictions. This is an advantage in cases such as the vacation rental domain, which includes schema columns *bedrooms*

and *bathrooms* which have very similar data values but, typically, different distributions of these values. Additionally, comparing distributions of features helps us avoid overfitting when dealing with high-dimensional feature spaces and small numbers of training examples.

A common method of comparing two distributions is Kullback-Leibler Divergence, or KL-Divergence. For the kth feature type, we have a distribution P_{kc} for the schema column c in the training data, and P_{kf} for the data field f in the target site. The KL-Divergence from the training data distribution to the data field distribution for feature type k is:

$$KL_k(c||f) = \sum_i P_{kc}(i) \log \frac{P_{kc}(i)}{P_{kf}(i)} \tag{1}$$

An issue with KL-Divergence is that when $P_{kc}(i) > 0$ and $P_{kf}(i) = 0$ for some feature value i, the KL-Divergence is undefined. To counter this, we use Skew Divergence, a smoothed version of KL-Divergence [13]:

$$SD_k(c||f) = KL_k(c||\alpha * f + (1 - \alpha) * c) \tag{2}$$

Note that $\alpha = 1$ gives the original KL-divergence. A value close to 1 gives a slightly smoothed KL-Divergence. We use $\alpha = 0.8$ in this work.

We alter the Skew Divergence with a simple transformation to create the Skew Similarity score, which has value 0 for the data field f most dissimilar from the schema column c, and highest value for the data field which is most similar to the schema column in the training data.

$$SS_k(c, f) = [\max_f SD_k(c||f)] - SD_k(c||f) \tag{3}$$

Our choice of Skew Divergence as the method of comparing distributions is one of many reasonable choices. We chose to use a smoothed version of the KL-Divergence. Other measures of the distance between two distributions, such as the Jensen-Shannon divergence, would also be reasonable.

Combining Skew Similarity Scores. Skew Similarity scores for different feature types will have different scales, and in some cases might actually be misleading (for instance, if contextual features are not helpful for some schema column in a domain, we want to ignore the Skew Similarity score for these features). Thus, we cannot simply average or sum the scores. Instead, we combine Skew Similarity scores for the different feature types using a linear regression model. A data field f on a target site is given a score $LR_c(f)$ for a schema column c using a linear regression model which combines the K different similarity scores (one for each feature type). The final score is a weighted combination of the Skew Similarity scores plus a constant:

$$LR_c(f) = \beta_{0c} + \sum_{k=1}^{K} \beta_{kc} SS_k(c, f) \tag{4}$$

If we view the Skew Similarity scores as different classifiers, each making predictions based on models over features learned from training data, then this overall model can be viewed as a stacked classifier model. Stacking, which generally refers to using the outputs of classifiers as inputs to another classifier, has been shown to often be an effective way of combining the predictions of multiple classifiers [14]. Our choice of linear regression as the 'top-level' classifier was motivated by the good empirical performance observed across a wide variety of data sets in previous work on stacking [14], and by ease of implementation. A number of other choices could be made for the top-level classifier, including logistic regression if posterior probability estimates were desired.

Training the Model. Training the stacked classifier regression model involves learning the weights β. This is done by holding out each training site, and generating data points for the linear regression for each data field on the held-out site with Skew Similarity scores computed relative to the other training sites. Using such a held-out approach to training the model is important, because otherwise the training examples will have very biased estimates of the accuracies of the Skew Similarity scores (they will appear to be much better than one would expect on held-out data). For every data field f on a training site and every schema column c in the domain, we generate an example: $(\delta(c,f), SS_1(c,f), SS_2(c,f), \ldots, SS_K(c,f))$ where $\delta(c,f) = 1$ if the data field f is annotated with schema column c in the training data and 0 otherwise. The coefficient β_{kc} controls how strong an indicator feature type k is for schema column c. The coefficients allow our method to learn which feature types are reliable indicators on a per-domain, per-schema column basis, which is essential for robust performance across a wide variety of domains and sites.

Motivation for the Model. One might question why we did not use a model for each schema column that learned a classifier over a feature vector that held all of the features that describe a data field. The key reason is that each annotated training site yields only as many training examples as there are data fields on that site. For example, with 5 training sites and an average of 20 data fields detected per site (typical values in our setting), we would have 100 examples, most of which would be negative examples. With the various character n-gram features and token features that we need in order to robustly recognize variations of data values and contexts, the dimensionality of the feature vectors would reach tens of thousands of features. It is difficult to train a reliable classifier in such a setting. Even with appropriate feature selection and regularization methods, with so few training examples and so many features, overfitting seems inevitable.

We instead chose a stacked model that combines a small number of similarity scores. Our choice was motivated by several reasons. First, a similar model design that combined a set of base learners with a regression-based classifier has been shown to be useful for a related problem, matching database schemas [10]. Second, such a model has to learn only $k + 1$ parameters for each schema column, where k is the number of base learners to combine. In our setting, where $k = 4$, we can expect to obtain sensible estimates of these parameters. Third,

similarity measures like Skew Divergence are effective ways to compare distributions of values and summarize them with a score. Finally, we desired a model that could be easily extended with additional sources of information, which our model facilitates. The new information can be simply added to the regression model.

In our evaluation, we compare these two approaches, using a regularized logistic regression classifier as a baseline approach that uses all features at once. Our results suggest that overfitting is a significant issue for the baseline method, and that our model yields superior performance.

3.3 Labeling the Target Site

After we have computed a score for each possible mapping of a data field on the target site to a schema column, we must output a labeling of the target site. The output of our method depends on the application setting. If we are operating in a fully-automated setting, the system chooses the data field f which maximizes the score $LR_c(f)$ for each schema column c, and annotates all pages on which those data fields occur with the appropriate labels (a data field does not always occur on every page). Not all schema columns occur on every target site, so we choose no data field if the maximum linear regression score is below a threshold θ for a schema column[5].

If we are aiding a human annotator, we recommend the top N data fields on the target site for each schema column. This dramatically reduces the annotation effort required, as the annotator only has to look at a handful of data fields.

We consider both scenarios in our evaluation.

4 Evaluation

The evaluation of our method was designed to measure the accuracy of automatically labeling new sites, and also to measure how well we could make recommendations to human annotators. Given a collection of annotated sites for a domain, we performed our evaluation in a cross-validated fashion, training the system on all but one of the sites, and testing on the held-out site. We used as our gold standard the nearly-perfect machine annotations from our supervised extraction system (hand-checking a large sample of annotations failed to find any errors in many dozens of pages, implying that they were over 99% accurate). In the results below, we refer to our method as the 'Stacked Skews Model.'

4.1 Baseline Method

As discussed in Sect. 3.2, an obvious approach to the problem we consider is to extract a feature vector describing each schema column from each training site, and train a multiclass classifier to classify data fields with which schema column they match (and to decide which data fields are irrelevant). To evaluate the

[5] We used $\theta = 0.1$ in this work, selected based on the development data.

performance of our system relative to this baseline, we implemented a regularized logistic regression classifier as our baseline method. The output of the classifier is used just as the output of the linear regression model is in our system, with the highest scoring data field for each schema column being mapped to that schema column. We hand-tuned the threshold below which the baseline decides that a schema column is not present on a target web site (we used a value of 0.2, which optimized the test results for the baseline). In the remainder of this section, we refer to this baseline method as 'Logistic Regression.'

4.2 Metrics

The metrics we used to evaluate our system are:

- Accuracy: On each page of the test site, we create an annotation for each schema column in the domain schema, or assert that the schema column is not present on the page. We measure the accuracy of these decisions[6].
- Recall in Top 3: For a test site, we select the top 3 data fields for each schema column. On each page of that test site, we check to see if each correct annotation on that page is in the top 3 ranked data fields for its label.

4.3 Domains

We evaluated our system using two different domains: vacation rentals and job sites. Vacation rental sites list properties offered for rent, with attributes such as number of bedrooms, maximum occupancy, and a text description of the property. Job listing sites describe open job positions, and include attributes like the company offering the job, the date it was posted, and the location of the job. Refer to Fig. 6 for the complete list of schema columns for each domain schema. We developed our methods using other sites in the vacation rentals domain than the ones we ultimately trained and tested on. The vacation rental sites used in our evaluation were not seen until the system development was finished, nor were any sites in the jobs domain.

4.4 Web Sites

We selected web sites for each domain that had most of the schema columns listed in Fig. 6 present and that were amenable to tree alignment. We were unable to find publicly available data suitable for our evaluation; while there was limited annotated data available publicly for job sites, our method needed at least 5 different sites in a domain annotated with a similar schema. Job sites had 240 pages and 17 detected data fields on average. Vacation rental sites had 151 pages and 25 detected data fields on average.

[6] Some schema columns appear on pages in multiple locations. Any correct location is acceptable.

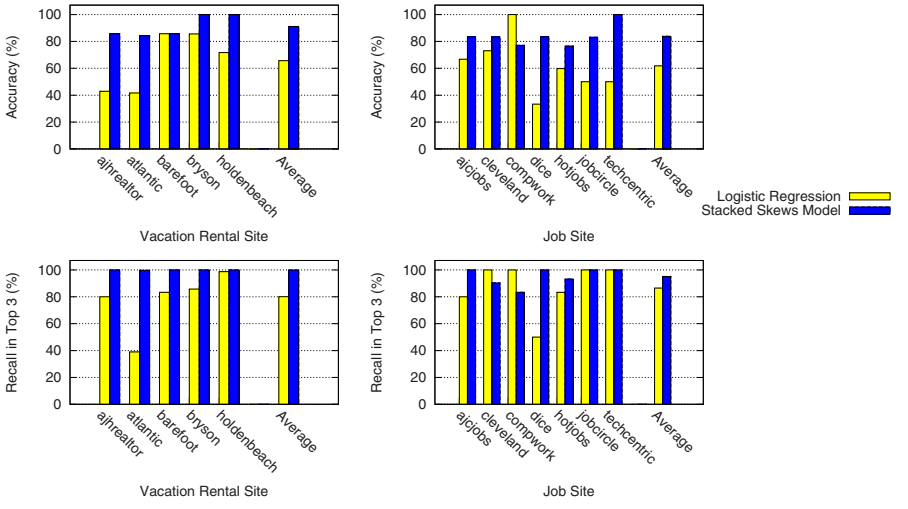


Fig. 5. Results for Logistic Regression and Stacked Skews Model when holding out each site from the training data and testing on it, averaged across schema columns. The 'Average' columns give results averaged across sites and columns. *(Top and Bottom Left)* Accuracy and Recall in Top 3, respectively, for each site in the jobs domain. *(Top and Bottom Right)* Accuracy and Recall in Top 3 for the vacation rentals domain.

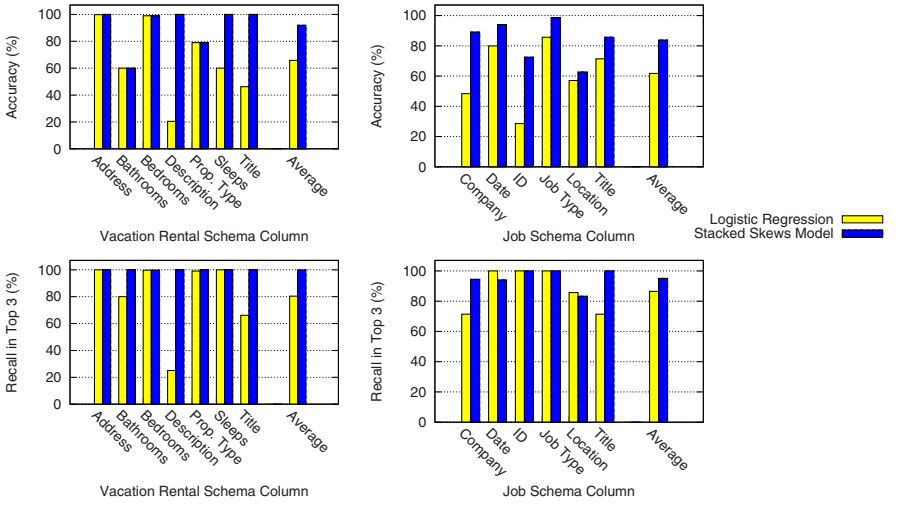


Fig. 6. Results for Logistic Regression and Stacked Skews Model by schema column, averaged across using each held-out web site as a test set. The 'Average' columns give results averaged across sites and schema columns. *(Top and Bottom Left)* Accuracy and Recall in Top 3, respectively, for each schema column in the jobs domain. *(Top and Bottom Right)* Accuracy and Recall in Top 3 for the vacation rentals domain.

Table 1. Results from feature type ablation experiments with the Stacked Skews Model. The top 4 rows give results for leaving out one feature type, and the rows below those give results with just one feature type.

	Jobs		Vacation Rentals	
Features	**Accuracy**	**Top 3**	**Accuracy**	**Top 3**
Excl. Context n-grams	85.3	95.1	54.4	86.5
Excl. Value n-grams	75.1	89.7	91.1	99.9
Excl. Value Token Types	83.8	95.1	91.1	99.8
Excl. Value Tokens	73.7	89.7	85.4	99.9
Only Context n-grams	41.8	61.1	77.0	89.9
Only Value n-grams	68.0	97.8	62.8	99.8
Only Value Token Types	30.2	48.2	31.3	46.5
Only Value Tokens	71.4	81.6	54.3	89.9
All	**83.8**	**95.1**	**91.1**	**99.9**

4.5 Results

Results by Site. Figure 5 shows results for each site, averaged across the different schema columns in the domain, for both the Logistic Regression baseline and our Stacked Skews Model. Averaged across all sites and schema columns (indicated by the 'All' column in the figure), our method achieved an accuracy of 91.1% for vacation rental sites, and 83.8% for job sites, significantly higher than the baseline accuracies of 65.8% and 61.8%. The results are reasonably consistent across different sites. Additionally, the correct data fields for vacation rental sites were present in the top 3 recommendations 99.9% of the time, and 95.1% for job sites. The baseline classifications of data fields had the correct answers in the top 3 only 80.3% and 86.5% of the time. As an additional comparison, random assignment of data fields to schema columns would have an expected accuracy of 5.9% for job sites, and 4.0% for vacation rentals, and an expected top 3 performance of 17.6% for job sites, and 12.0% for vacation rentals.

Results by Schema Column. Figure 6 gives results for each schema column for our methods, averaged across web sites. We see that our method is generally accurate for a wide range of data types, and nearly always exceeds the baseline results. Our method is particularly better suited to schema columns which tend to have free text (for example, the *description* and *title* schema columns for vacation rentals, and the *company* and *ID* schema columns for jobs). We believe that this is due to the logistic regression's tendency to overfit when many different features can perfectly classify the training data. The baseline method fares much better for schema columns with more limited vocabularies.

Referring to the Stacked Skews Model results for individual schema columns, most of the schema columns have high accuracy. The cases where performance suffered were the *job type* schema column, which has high variation in both contexts and values (some sites even use special codes as values for this schema

column), *bathrooms*, which had trouble when there was also a 'Half Bathrooms' item on a site, and *property type*, where one site incorrectly labeled a data field as *property type* because it was preceded by the text 'Property Type' on the site, but it was a different sense of property type from the intended meaning from our domain schema[7].

Identifying Missing Schema Columns. Most of the schema columns in each domain were present on our evaluation sites. In the case where a column was not present on a site, the accuracy metric required us to correctly identify that column as missing, or else it was considered an incorrect answer. Identification of such missing columns was described in Sect. 3.3. We evaluated the accuracy of our model for these cases, to see if they were a significant source of error in our evaluation. For each domain, there were 5 cases where a schema column was missing from a site. The Stacked Skews Model identified missing schema columns for vacation rentals with an accuracy of 80.0%, and a lower accuracy of 49.3% for job sites. This is because most job listing sites had some unstructured text describing the job, in addition to well-formatted sections which typically held our extraction targets. Often when a schema column was not present in the semi-structured text of a site, one of the data fields corresponding to the free text was chosen.

Feature Type Ablation Study. To assess the contributions and relative importance of each of the feature types, we ran ablation experiments where we used subsets of the four feature types. We considered combinations where we held out each feature type, and also where we used each feature type alone. Table 1 gives results for these experiments. We see that context features are very informative for the vacation rentals domain, but not informative for the jobs domain (in fact, excluding them improves average accuracy). The value token type features do not appear to be useful for either domain. In general, we see that using multiple feature types in combination allows the system to achieve higher accuracies than any single feature type, indicating that the different feature types provide complementary information, and that our stacking method effectively combines these sources of information.

5 Conclusions

This work addressed the problem of extracting structured records from semi-structured web pages. The system we described demonstrated a way to learn high-quality automated extraction systems for large numbers of semi-structured web sites, by exploiting a tiny, fixed amount of human effort per domain of interest. Starting from manual annotation of 2–5 pages each on 4–6 sites in a domain, we bootstrapped a system that achieves high accuracies even on a domain, jobs, that was not considered during development of our model. This

[7] Our domain schema included a notion of property type as a house, townhouse, condominium, etc. The site's notion of property type was beachfront vs. not.

performance is encouraging for use either as a standalone extraction system for certain applications, or as an aid to human annotators. The performance significantly exceeds the performance of a competitive baseline method.

Acknowledgments. Most of this work was done during an internship at Google in Pittsburgh. The authors wish to acknowledge Kamal Nigam, William Cohen, Gideon Mann, and the anonymous reviewers for their helpful comments, William Morris and Dominic Widdows for assistance with annotating data, and Haakan Younes for his reimplementation of the Partial Tree Alignment algorithm.

References

1. Liu, B., Grossman, R.L., Zhai, Y.: Mining data records in web pages. In: KDD, pp. 601–606 (2003)
2. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine Learning 34(1-3), 233–272 (1999)
3. Kushmerick, N., Weld, D.S., Doorenbos, R.B.: Wrapper induction for information extraction. In: IJCAI, pp. 729–737 (1997)
4. Muslea, I., Minton, S., Knoblock, C.: A hierarchical approach to wrapper induction. In: AGENTS, pp. 190–197 (1999)
5. Chang, C.H., Lui, S.C.: IEPAD: information extraction based on pattern discovery. In: WWW, pp. 681–688 (2001)
6. Chang, C.-H., Kuo, S.-C.: OLERA: Semisupervised web-data extraction with visual support. IEEE Intelligent Systems 19(6), 56–64 (2004)
7. Zhai, Y., Liu, B.: Web data extraction based on partial tree alignment. In: WWW, pp. 76–85 (2005)
8. Wang, J., Lochovsky, F.H.: Data extraction and label assignment for web databases. In: WWW, pp. 187–196 (2003)
9. Golgher, P.B., da Silva, A.S., Laender, A.H.F., Ribeiro-Neto, B.A.: Bootstrapping for example-based data extraction. In: CIKM, pp. 371–378 (2001)
10. Madhavan, J., Bernstein, P.A., Doan, A., Halevy, A.: Corpus-based schema matching. In: ICDE, pp. 57–68 (2005)
11. Freitag, D.: Multistrategy learning for information extraction. In: ICML, pp. 161–169 (1998)
12. Crescenzi, V., Mecca, G., Merialdo, P.: Wrapping-oriented classification of web pages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 1108–1112. Springer, Heidelberg (2003)
13. Lee, L.: Measures of distributional similarity. In: ACL, pp. 25–32 (1999)
14. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)

Online Multiagent Learning against Memory Bounded Adversaries

Doran Chakraborty and Peter Stone

Department of Computer Sciences
University of Texas
Austin, Texas, USA
{chakrado,pstone}@cs.utexas.edu

Abstract. The traditional agenda in Multiagent Learning (MAL) has been to develop learners that guarantee convergence to an equilibrium in self-play or that converge to playing the best response against an opponent using one of a *fixed set* of known targeted strategies. This paper introduces an algorithm called *L*earn *o*r *E*xploit for *A*dversary *I*nduced *M*arkov *D*ecision *P*rocess (*LoE-AIM*) that targets optimality against any learning opponent that can be treated as a memory bounded adversary. *LoE-AIM* makes no prior assumptions about the opponent and is tailored to optimally exploit any adversary which induces a Markov decision process in the state space of joint histories. *LoE-AIM* either explores and gathers new information about the opponent or converges to the best response to the partially learned opponent strategy in repeated play. We further extend *LoE-AIM* to account for online repeated interactions against the same adversary with plays against other adversaries interleaved in between. *LoE-AIM-repeated* stores learned knowledge about an adversary, identifies the adversary in case of repeated interaction, and reuses the stored knowledge about the behavior of the adversary to enhance learning in the current epoch of play. LoE-AIM and LoE-AIM-repeated are fully implemented, with results demonstrating their superiority over other existing MAL algorithms.

1 Introduction

The aim of many adversarial strategic interactions is to learn a model of the opponent(s) and to respond accordingly [1,3,13]. If the opponents execute static policies, then the learning agent is faced with a stationary environment, thus reducing the problem to effectively a single-agent decision problem. However when in the presence of other learning agents, there is an inherent non-stationarity in the environment which makes the learning problem for an individual agent much harder [11]. The most popular solution concept in such multiagent settings has been the Nash equilibrium [12] and most multiagent learning (MAL) algorithms proposed to date aim at convergence to such an equilibrium in self-play [5,8,14].

Their popularity notwithstanding, the ability to find Nash equilibria does not solve all multiagent problems. For one thing, there can be multiple Nash

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 211–226, 2008.

equilibria in general sum games: MAL algorithms provide no guarantee that the Nash equilibrium attained at convergence will be the one maximizing social welfare. Furthermore, an algorithm that converges to such an equilibrium in self-play may perform poorly when faced with an adversary that behaves differently.

Motivated in part by this observation, Powers and Shoham recently proposed an alternate set of evaluation criteria for MAL algorithms, focusing on *Targeted Optimality*, *Auto Compatibility* and *Safety* [13]. In their setting, the goal is to converge to within ϵ of the best response if the opponent uses one of a set of known targeted strategies, to within ϵ of a Pareto-dominant Nash equilibrium in self-play, and to within ϵ of the safety value against any unknown opponent. The authors further proposed an algorithm that meets these criteria against a set of target opponents [13,16]. The optimal responses to the stored set of target strategies are pre-computed, such that when an opponent is recognized to be using such a strategy, the matching response can be played. While their approach is effective for a fixed set of opponents, no prior learning algorithm guarantees outcomes greater than the safety value against arbitrary opponents. This paper introduces the first algorithm capable of meeting the Powers and Shoham criteria against adversaries of a finite memory size. We show that a large class of existing algorithms are actually memory bounded and can be exploited by our approach. To the best of our knowledge, this learning algorithm is the first that targets optimality against a mixture of opponents with different properties and goals. Rather than fixing the set of target opponents, we instead focus our algorithm on any adversary that induces a Markov Decision Process (MDP) according to the *A*dversary *I*nduced *M*DP (*AIM*) model [1]. By this model, it can be shown that for a large class of opponents, the learner finds itself in an MDP whose states are determined by bounded histories of joint actions and whose transition function is determined by the opponent's strategy. Specifically, we introduce an algorithm *L*earn *o*r *E*xploit for *AIM* (*LoE-AIM*) that either explores and gathers new information about the opponent or converges to the best response to the partially learned opponent strategy.

To demonstrate *LoE-AIM*'s effectiveness, we first test it against opponents (both deterministic and stochastic) drawn from the literature of MAL research. Our results show that in most cases, *LoE-AIM* converges to playing the optimal policy against the opponent without knowing the opponent's identity.

Unfortunately it is infeasible to develop a learning algorithm that plays optimally against *every possible* memory bounded opponent of a fixed memory size without the ability to restart play (i.e. erase the history and start over), e.g., consider the following opponents in the Prisoner's Dilemma (PD) game (Table 1(a)): (1) one which always plays cooperate, (2) one which starts playing cooperate, but defects forever if the opponent ever defects once (known as "grim-trigger"). It is not possible to develop a learner which can learn to play optimally against both the opponents without having a restart. Just to differentiate between them, the learner must play defect, and once it does so, it loses the chance of attaining the (cooperate, cooperate) payoff against the grim-trigger opponent.

On the other hand, in online learning it is not uncommon to face the same *type* of adversary in multiple well-defined "epochs" of several plays, possibly with epochs against other types of adversaries interleaved in between. In such situations an effective restart is possible: each time a new opponent of the same type appears, the history starts over, but the experience from past epochs remains. Specifically, we consider the case in which the learning agent plays against multiple adversaries that it knows are drawn from the same population and therefore use the same (or similar) strategy. It plays against each individual for a finite time before playing against the next. This scenario is representative of common cases such as online auctions in which an auctioneer repeatedly sells goods to a pool of bidders. Bids in each auction are irrevocable, but the process restarts when the next good is introduced to the market. In such a setting, we propose a mechanism *LoE-AIM-repeated* that leverages such repeated interactions to learn a model of the opponent and store it in its repository of learned models. When playing a new adversary, it tries to map the model of the new adversary to one of the stored models and uses the knowledge it gathered before about the adversary to further enhance learning in the current epoch.

The remainder of this paper is organized as follows. Section 2 presents the background necessary for our work. Section 3 summarizes possible adversaries in the existing MAL literature and introduces the class of opponents targeted by *LoE-AIM*. Sections 4 and 5 introduce the *LoE-AIM* algorithm and *LoE-AIM-repeated* respectively, including results achieved against memory bounded adversaries, and Section 6 concludes.

2 Background and Definitions

In this section we introduce the definitions and concepts necessary for our work. We focus on bimatrix stage games because they are general enough to fully explore the concepts we propose and simple enough to implement, study and relate to the existing MAL literature.

Definition 1 (Bimatrix Game:). *A bimatrix game is defined by a pair of matrices* $\{M_i, M_o\}$ *where each* $M_{x|x\in\{i,o\}}$ *is of size* $|A_i| \times |A_o|$ *and* $M_x : A_i \times A_o \mapsto \Re$ *maps every possible joint-action to a reward received by agent* x. A_i *and* A_o *are the sets of actions available to agents* i *and* o *respectively.*

For the rest of the paper we consider agent i to be the learner under our control and agent o to be the opponent.

Definition 2 (History(h^k):). *A history* $h^k = (a_i, a_o)^k$ *where* $a_i \in A_i, a_o \in A_o$ *is the sequence of the last* k *joint actions played by the agents. In other words, a history is a vector of length* k *consisting of the past* k *joint actions played by the agents. Often* k *is referred to as the window size or length of the history.* $h^k(j)$ *is the* j*th joint-action in the sequence* h^k *where* $0 \leq j < k, k \in \mathbb{N}$ *with* $h^k(0)$ *being the most recent joint action. Similarly* $h^k_o(j)$ *is the* j*th action played by agent* o *in the sequence* h^k *with* $h^k_o(0)$ *being the most recent action played by* o. *The*

history at time t is denoted $h^{k,t}$; thus the action played by agent o, j steps before time t is denoted $h_o^{k,t}(j)$.

For the rest of the paper, we refer to the memory size of the adversary as k.

Definition 3 (Policy (π_o):). *A policy π_o of o maps the history to a probability distribution over o's action set, i.e., $\pi_o : h^k \mapsto \Delta A_o$ where k is the memory size of agent o. The probability of playing action j following the policy $\pi_o(\cdot)$ is given by $\pi_o(\cdot)(j)$.*

Definition 4 (Memory Bounded Opponent:). *An opponent is said to be memory bounded if it follows a policy as specified above.*

Now we briefly review some definitions related to Markov Decision Processes (MDPs).

Definition 5 (Markov Decision Process (MDP) :). *An MDP M on a set of states S and with action set $A = \{a_1, \ldots, a_k, \ldots, a_{|A|}\}$ consists of*

Transition Probabilities: For each state-action pair (s, a), a next-state distribution $P_{s,a}(s')$ gives the probability of moving to state s' when action a is taken in state s.

Reward Distribution: For each state-action pair (s, a), a reward distribution $R(s, a)$ specifies the probability distribution on a set of real numbers that can be achieved as reward given action a is taken in state s.

2.1 Adversary Induced Markov Decision Process (AIM)

The key insight enabling this research is that in the setting of a repeated game where the adversary is a memory bounded opponent, the dynamics of the system can be modeled as a MDP whose transition probabilities and reward functions are determined by the model of the opponent. For a history of play (a "state") $h^{k,t}$, the next state $h^{k,t+1}$ and the reward received are determined by the current state $h^{k,t}$, the adversary's policy in that state $\pi_o(h^{k,t})$, and the action a_i chosen by agent i.

Definition 6 (Adversary Induced Markov Decision Process:). *An Adversary Induced MDP (AIM) $\mathcal{M}$ is defined as follows,*

State Space ($\mathcal{S}$): The state space $\mathcal{S}$ of $\mathcal{M}$ is given by $\boldsymbol{a^k}$ where $a \in A_i \times A_o$, i.e, set of all possible joint histories of length k. From now onwards we will use the word state and history interchangeably.

Action Space ($\mathcal{A}$): The action space $\mathcal{A}$ of $\mathcal{M}$ is given by A_i. The action space is just the set of actions available to agent i.

Transition Probabilities ($\mathcal{P}$): Intuitively, the history is updated as a sliding window. Transitioning from a history $h^{k,t}$ to a history $h^{k,t+1}$ is just keeping the last $k-1$ joint actions (each shifted one time step backwards) and including the latest pair at index 0 of the vector. The transition probability of transitioning from a history $h^{k,t}$ to a history $h^{k,t+1}$ given the action taken being a_i is,

$$\mathcal{P}_{h^{k,t},a_i}(h^{k,t+1}) = \pi_o(h^{k,t})(h_o^{k,t+1}(0)) \text{ where } h_i^{k,t+1}(0) = a_i.$$
$$= 0 \text{ o.w}$$

Note that, there is a non-zero probability to transitioning to only those histories which end in action a_i as they are the possible histories for this transition. For all other histories, the transition probability is 0. If $\pi_o(h^{k,t})$ is stochastic, then $\mathcal{P}_{h^{k,t},a_i}$ is stochastic as well. Whether the AIM is ergodic [1] depends on π_o. For example, against an opponent playing grim-trigger in PD, once learner play a defect action, it can never transition to a state where the opponent has recently played cooperate.

Reward Function ($\mathcal{R}$): The reward function $\mathcal{R}$ of $\mathcal{M}$ is given by $\mathcal{R}(h^{k,t}, a_i) = E_{a_o \sim \pi_o(h^{k,t})} M_i(a_i, a_o)$.

3 A Taxonomy of Possible Adversaries

The algorithms introduced in this paper target adversaries whose action at time t depend on at most the past k joint actions ($h^{k,t}$). In this section, we show that this apparently restrictive class of adversaries actually captures a large class of opponents from the literature. In order to do so, we present a taxonomy of possible adversaries, along with how several existing strategies can be classified within it. This taxonomy is summarized in Figure 1.

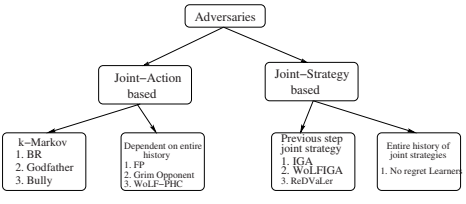


Fig. 1. A taxonomy of possible adversaries

First, an adversary can be broadly classified as either joint-action based or joint-strategy based. A joint-action based adversary bases its current action on the joint-actions played in the past. They can be further classified as k-Markov opponents whose policies depend only on the past k joint-actions, or opponents whose current action depend on the entire history of play. Examples of k-Markov opponents include Bully, Godfather [15] and Best Response (BR), while Fictitious Play (FP) [9], Joint-action learner (JAL) [7], and the family of Q-learners (e.g, PHC and WoLF-PHC [5]) depend on the entire history. A BR opponent plays the best reponse to the empirical distribution of the opponent's play captured by the current history. If memory size is unbounded, BR is equivalent to FP.

[1] A MDP is ergodic if there is a non-zero probability of eventually transiting from every state to every other state (possibly via some number of intermediate states).

In contrast to a joint-action based adversary, a joint-strategy based adversary bases its current step strategy on the past history of joint *strategies*: not just the actual plays, but the probability distributions from which they were drawn. In practice, it is unnatural to assume that the opponent strategy is ever known. Thus in this paper, the past step opponent strategy is estimated based on the recent history. In this paper we estimate the opponent strategy by the frequency of each action played by the opponent in the captured history at that time instant. As a result, joint-strategy opponents are in effect also joint-action opponents. Nonetheless, we classify them differently since in the literature they are presented and analyzed as acting based on past joint-startegies.

Similar to the joint-action case, joint-strategy based adversaries can be further classified based on whether the current step strategy depends either on just the past step joint strategy or the entire history of joint strategies. Examples of the former are MAL algorithms which converge to a single stage Nash Equilibrium in a repeated setting (e.g. IGA [14], WoLF-IGA [4] and ReDVaLer [2]) while examples of the latter are no-regret learners which attempt to minimze the cost of online learning [10].

As our targeted opponents in this paper, we consider the k-Markov joint-action opponents and single-step joint-strategy adversaries. Though our results are against a sample of such opponents drawn from the literature, our claims hold for any opponent which induces an AIM in a joint-action space of bounded length.

4 LoE-AIM

This section introduces the *LoE-AIM* algorithm which is the heart of our overall learning mechanism. We present two versions of this algorithm, one for opponents which play deterministically (e.g. Bully, Godfather, and BR) and another for opponents who play stochastic stratgies (e.g. MAL algorithms). We start by assuming that the player[2] knows whether the opponent is playing deterministically or stochastically. In Section 5 we present a more general framework which enables the player to learn this attribute of the opponent well.

Algorithm 1 presents the version of the learning algorithm for deterministic opponents. Due to space constraints, we only present the high level algorithm and for all called methods, we give a textual explanation. The algorithm takes as input the current opponent-model ($\hat{\pi_o}$), the current start state (history) and the number of episodes for which it should continue learning. Note $\hat{\pi_o}$ refers to some partially learned model if it exists. If the algorithm has no prior information about the opponent it is playing, opponent-model is null. All the results presented in this section assume that there exists no such partial model and the learner learns from scratch. In Section 5 when we talk about repeated interactions with an opponent, then the $\hat{\pi_o}$ fed as input can be a partially learned model from past interaction(s) with the same opponent. The algorithm outputs the final $\hat{\pi_o}$ and the solved AIM strategy (π_i) governing the model. π_i is explained

[2] From this point onwards we will refer to the learner as the player.

Algorithm 1. LoE-AIM-DETERMINISTIC

```
begin
    input  : episodes, π̂_o, history
    output: π̂_o, π_i
1   episode ← 0
2   π_i ← SOLVE-AIM-MODEL(π̂_o)
3   for episode++ < episodes do
4       opponent-action ← action taken by opponent
5       player-action ← action as per π_i
6       if {history, opponent-action} ∉ π̂_o then
7           π̂_o ← π̂_o ∪ {history, opponent-action}
8           π_i ← SOLVE-AIM-MODEL(π̂_o)
            history ← UPDATE-HISTORY(history, {player-action, opponent-action})
end
```

below. Since the opponent is deterministic, just one visit is needed to a state to know what the opponent's policy is for that state. The SOLVE-AIM-MODEL function finds a control policy (π_i) for the underlying AIM by assuming that for all known histories h_t of play, the opponent plays $\hat{\pi_o}$ and for all *unknown* histories, the opponent plays the maximax strategy for the player (the strategy that maximizes the maximum pay-off for the player). The assumption for unknown histories causes π_i to explore towards histories of play not visited before. The UPDATE-HISTORY method updates the history by prepending the most recent joint action and removing the oldest joint-action.

Algorithm 2 is similar to Algorithm 1 except now that opponent can play stochastic strategies. In this case, the player maintains a stochastic model of the opponent. UPADATE-OPPONENT-MODEL updates $\hat{\pi_o}$ with the latest decision taken by the opponent. Note, "updating" here means updating the percentage of times an action has been played for that state and then normalizing over all possible actions. The HAS-CHANGED-OPPONENT-MODEL? returns true if for any state the probability of taking an action is η gretear than that of the same action in the previous solved model and the number of visits to that state is at least κ. All results in this paper use values for η and κ that led to the best results in informal preliminary testing, namely $\eta = 0.1$ and $\kappa = 20$.

Lemma 1. *In repeated infinite play LoE-AIM either converges to the optimal policy for the partially learned opponent model or keeps expanding the learned model.*

Proof. Let $\bar{\pi_o}$ be the remainder of π_o that needs to be learnt at a particular time instant. $\hat{\pi_o}$ refers to the part of the opponent strategy that the player knows while $\bar{\pi_o}$ being the part that still needs to be explored. By solving for a control policy for $\hat{\pi_o}$ where for every state in $\bar{\pi_o}$ the player believes that it could get the best possible reward (since it assumes that the opponent playes the maximax strategy for the player at those states and the value of maximum possible achievable reward is known), the algorithm generates π_i that will always

Algorithm 2. LoE-AIM-STOCHASTIC

```
begin
    input  : episodes, π̂_o, history
    output: π̂_o,π_i
1   episode ← 0
2   π_i ← SOLVE-AIM-MODEL(π̂_o)
3   for episode++ < episodes do
4       opponent-action ← action taken by opponent
5       player-action ← action as per π_i
6       π̂_o ← UPADATE-OPPONENT-MODEL(π̂_o, {history, opponent-action})
7       if HAS-CHANGED-OPPONENT-MODEL?(π̂_o) then
8           π_i ← SOLVE-AIM-MODEL(π̂_o)
            history ← UPDATE-HISTORY(history, {player-action, opponent-action})
end
```

promote exploring states in $\bar{\pi_o}$. However if $\hat{\pi_o}$ is non-ergodic, then there are chances that the current state may prohibit transition to newer states, i.e, the strategy of the opponent is such that it prevents further expanding of the model. Then the algorithm converges to the optimal policy given the partially learned model.

Corollary 1. *If π_o is ergodic, then LoE-AIM converges to the optimal policy in infinite repeated play.*

Note that this exploratory aspect of *LoE-AIM* is motivated in part by the R-Max algorithm [6] which also deliberately balances exploitation with exploration of unvisited states. The main difference is that R-Max is designed for single agent MDP's and hence the exploration depends only on the action of the agent, whereas in AIMs, the agent and its adversary jointly determine the state space explored.

4.1 Results against Deterministic Opponents

This section presents the results achieved by *LoE-AIM* against the deterministic opponents mentioned in Section 3, namely k-Markov adversaries such as Godfather, BR and Bully.

Figure 2 shows the results achieved by *LoE-AIM* in the game of Prisoner's Dilemma (PD) (Table 1(a)) against a couple of variations of the Godfather, Bully [15] and BR strategies.

Godfather is a finite-state strategy that makes its opponent an offer that it cannot refuse. Godfather chooses a targetable pair [3]. From then on, if the opponent keeps playing its half of targetable pair in one stage, Godfather plays its half in the next stage. Otherwise it plays a strategy (threat) that forces the

[3] A pair of deterministic policies is a targetable pair if playing them results in each player getting more than the safety value and plays its half of the pair.

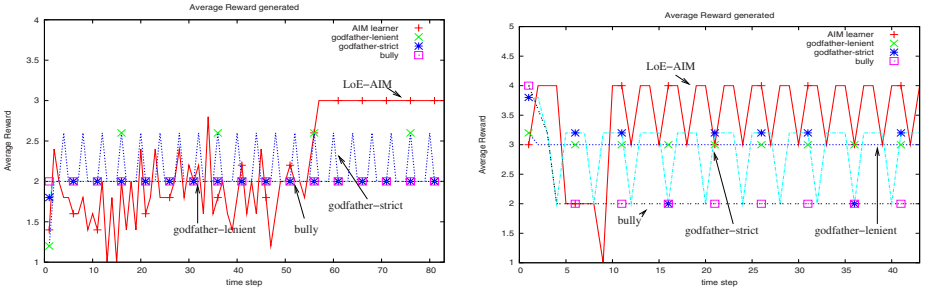
Table 1. Payoff matrices

(a) Prisoner's Dilemma

	cooperate	defect
cooperate	(3,3)	(1,4)
defect	(4,1)	(2,2)

(b) Chicken

	$A1$	$A2$
$A1$	(3,3)	(2,4)
$A2$	(4,2)	(1,1)



(a) Against Godfather-strict (b) Against Godfather-lenient

Fig. 2. Results against Godfather opponents in PD

opponent to achieve at most its safety value. Hence Godfather is a memory-bounded adversary with $k = 1$. We now introduce a couple of variations of the Godfather strategy that are tailored for $k > 1$.

- Godfather-lenient plays its part of a targetable pair if the opponent at least once played its own half of the pair (within the last k actions). Otherwise Godfather-lenient punishes its opponent by playing the threat strategy that reduces the opponent's best outcome to its safety value.
- Godfather-strict is a stricter version that always punishes its opponent if the opponent ever deviated from the targetable pair during the observable history.

Note that in case of PD, the Godfather players target the {cooperate, cooperate} pair and use defect as the threat strategy.

Bully is a deterministic strategy given by $\text{argmax}_{a_o \in A_o} M_o(a_i^*, a_o)$ where $a_i^* = \text{argmax}_{a_i \in A_i} M_i(a_i, a_o)$. The opponent optimizes its payoff by assuming that the Bully remains fixed while the Bully optimzes its payoff by assumimg that the opponent is the follower and would adapt accordingly.

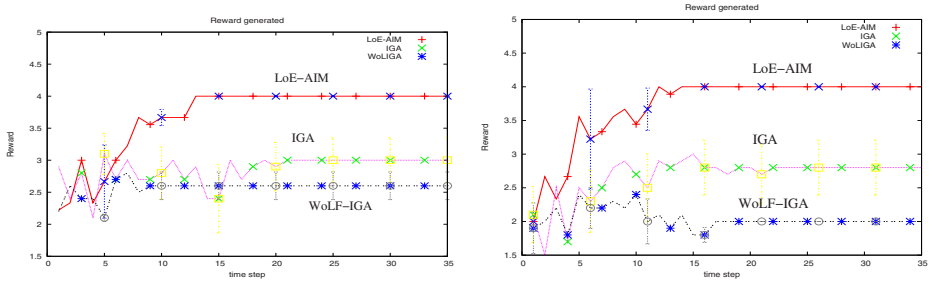
Now we present results which show that *LoE-AIM* exploits all of the above opponents without knowing their identity. For benchmarking purposes we also present results had the player chosen any of the deterministic strategies as its strategy instead of *LoE-AIM*. The results presented in Figure 2 are for $k = 3$ and averaged over 10 random instantiations of the start state (e.g. the assumed "history" of the opponent when it makes its first decision). However, each run is independent and the learner starts learning from scratch with each restart. In

the spirit of online learning, *LoE-AIM* converges to the optimal policy in each of the occasions without requiring a restart. Against Godfather-strict, the *LoE-AIM* algorithm eventually learns (after about 55 episodes of learning) that it should play cooperate (its half of the targetable pair) and hence converges to a payoff of 3 (Figure 2(a)). The results show that none except the Godfather-strict [4] strategy converge to the optimal payoff. For Godfather-lenient, *LoE-AIM* learns to optimally exploit by playing cooperate frequently enough so that the history always contains one cooperate action for the player. At convergence, the *LoE-AIM* player plays defect twice followed by a cooperate ensuring two consecutive payoffs of 4 followed by a payoff of 3 (Figure 2(b) shows that the average converged payoff (after about 10 episodes) oscillates between 3 and 4). In case of PD, both the Bully strategy and BR strategy is to play defect deterministically. Against both of these opponents, the learner eventually learns to play defect and converges to a payoff of 2 (for space constraints, we omit the graphs).

4.2 Results against Stochastic Opponents

We now present results of *LoE-AIM* learning against popular MAL algorithms that converge to single-stage Nash equilibrium in repeated play. Due to space constraints we only present results against IGA [14] and WoLF-IGA [3], but the algorithm also works against all other MAL algorithms that decide their next step strategy based on the past step joint-strategy (e.g. ReDVaLer [2]). We assume that the opponent cannot observe the player's past step strategy and hence approximates it by the proportion of each action played by the player in the current state (history). A point to note is that the opponent knows its own strategy for sure and uses it to compute its next step strategy. This makes the process non-Markovian in the space of the k-history. However if k is large enough, the proportion of each action played by the opponent will be close to its real strategy and hence will make the process approximately Markovian. Though it seems that larger the value of k the better, our results show that even for $k = 4$, *LoE-AIM* can efficiently model the opponent and exploit it to the optimum. Once again all our results are averaged over 10 different instantiations of the start state and learning at each restart starts from scratch. The learning rate used for IGA is 0.1 and the learning rates for WoLF-IGA are 0.1 and 0.2. Figure 3 gives evidence that the *LoE-AIM* learner was successful in reaching its optimal payoff in the game of chicken(Table 1(b)) by exploiting the MAL opponents on both the occasions. The reason we choose Chicken game is because the game has three Nash equilibria: two in pure strategies, sustaining the outcomes (4,2) and (2,4), and one in mixed strategies where the players play each of their actions with equal probability with the corresponding expected payoff of 2.5 for each agent. Neither IGA, nor WoLF-IGA guarantees the possible final converged Nash pay-off in self-play, e.g, in both Figure 3(a) and Figure 3(b), self-play generates outcomes much less than 4 showing that on numerous occasions the final converged Nash payoff was not (4,2), the one most coveted by the player. In contrast, in all of its runs, *LoE-AIM* converged to the outcome (4,2).

[4] Note, Godfather-strict strategy in self play always converge to the targetable pair.



(a) Against IGA (b) Against WoLF-IGA

Fig. 3. Results against MAL opponents in Chicken game

	A1	A2
A1	(3,4)	(2,1)
A2	(1,2)	(4,3)

((1,1) (0,0) (0.25,0.5))

	A1	A2
A1	(2,4)	(3,1)
A2	(1,2)	(4,3)

((0,0), (1,1) (0.25,0.5))

	A1	A2
A1	(2,3)	(3,4)
A2	(4,2)	(1,1)

((1,0) (0,1) (0.5,0.5))

	A1	A2
A1	(2,2)	(4,3)
A2	(3,4)	(1,1)

((1,0) (0,1) (0.75,0.75)

	A1	A2
A1	(3,3)	(2,4)
A2	(4,2)	(1,1)

((1,0) (0,1) (0.5,0.5))

	A1	A2
A1	(2,2)	(3,4)
A2	(4,3)	(1,1)

((1,0) (0,1) (0.5,0.5))

Fig. 4. Payoff matrices of 6 games with multiple Nash Equilibria. ((1,1) (0,0) (0.25,0.5)) means that the game has 3 Nash equilibria where the probabilities of playing action A1 by both players are respectively (1,1), (0,0) and (0.25,0.5).)

Figure 5 gives a summary of a head to head comparison among the various opponents discussed in this section together with results achieved by *LoE-AIM*. There are 78 structurally distinct 2×2 strict ordinal games in which the two players can strictly rank the four payoffs from best to worst. Of the 78 games, only 6 games (shown in Figure 4) have multiple Nash equilibria with each player favoring a different one. We present results from these games because in self-play none of the MAL algorithms guarantee the final converged Nash pay-off (the algorithms can converge to any one of the Nash equilibria depending on the learning rates and start states). Each point in the plot has been averaged over results achieved from all the 6 games, with the results in each game first averaged over 10 runs with different initial start states. For benchmark comparisons, we show head to head results achieved by various other algorithms that the player could have used as its default strategy instead of *LoE-AIM*. Figure 5 shows that against the MAL algorithms (IGA, WoLF-IGA) and BR, *LoE-AIM* successfully converged to its best outcome of 4 on all occasions thereby demonstrating its ability to exploit its opponent to the optimum. All the benchmarks generate lower average payoffs when played against these opponents. Against the other opponents, *LoE-AIM* still did better than all other benchmarks though the average outcome was lower than 4 in these cases. Note, that against certain opponents

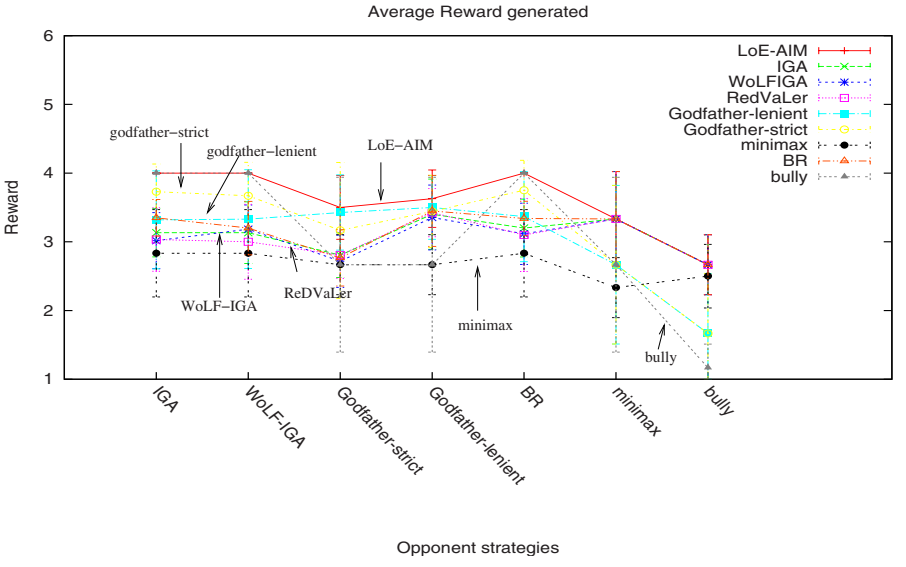


Fig. 5. Result against the 6 games with multiple Nash equilibria

it is never possible to achieve the 4 outcome because the opponent won't allow that, e.g, against Godfather-strict in PD (see Figure 2(a)).

5 LoE-AIM-Repeated

In online learning, repeated interaction with multiple opponents is quite common. For example, the player may plays opponent 1 for 10 rounds, opponent 2 for the next 10 rounds, and then again opponent 1 for another 10 rounds. One such scenario is a market with multiple sellers where the buyer is interested in learning an optimal negotiation strategy for buying items. The buyer negotiates in turn with different sellers and learns from these experiences.

Figure 6 presents *LoE-AIM-repeated*, which such a buyer can employ to maximize her return. We assume that the buyer has a fixed set of interactions (episodes) with an opponent in one run (EPOCH). For the first STORE-EPISODES number of plays, the buyer tries to build an approximate model of the opponent. The *LoE-AIM* method called with opponent-model set to null, outputs an approximate model and an AIM-strategy for that model. In these STORE-EPISODES number of plays, if ever the opponent took different actions for the same state, H_0 is set as stochastic, else H_0 is set as deterministic. Once the model has been built, the framework searches for the closest-model in the pool of stored models. The method Get-Closest-Model takes as arguments the pool of stored models and model, and returns the closest model that matches the model. In the deterministic case, closest-model is computed by iterating over all the stored deterministic models and returning the one which has the maximum number

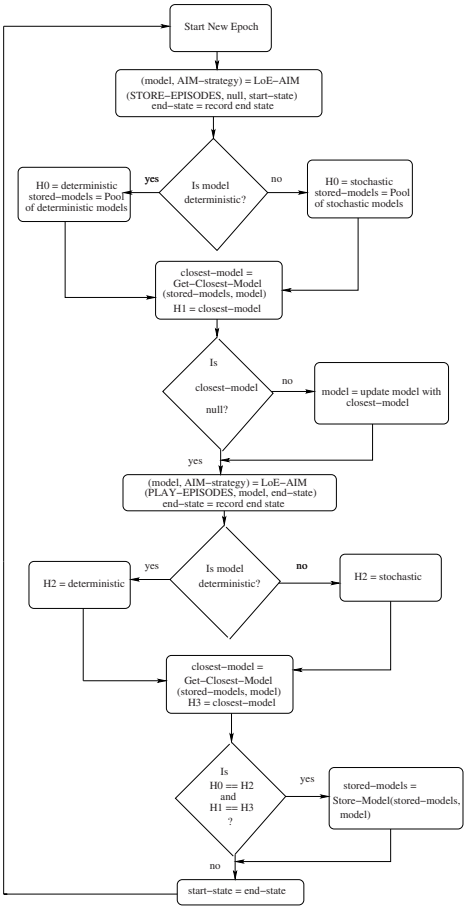
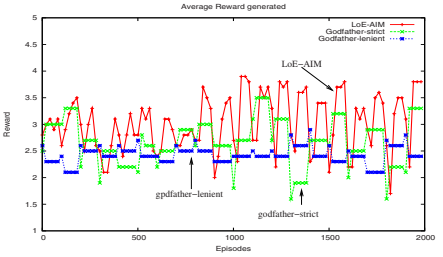


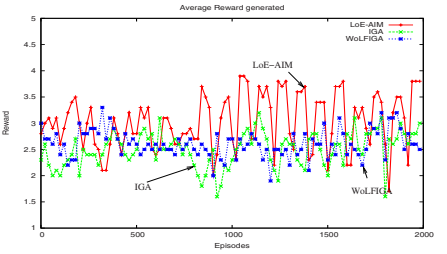
Fig. 6. *LoE-AIM-repeated*

of states such that the same decision would be taken. In the stochastic case, closest-model is selected by iterating over all the stored stochastic models and returning the one which has the minimum Max Norm distance from model. If a convincing closest-model exists (if the distance is smaller than a fixed threshold for the stochastic case), the model is updated with the closest-model. The player then calls the *LoE-AIM* method with the updated opponent model and runs it for PLAY-EPISODES.

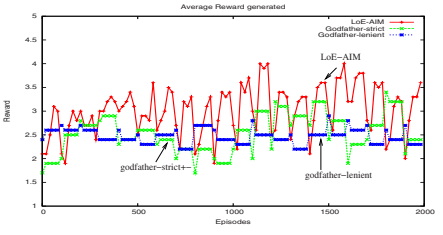
Next, the framework recomputes the closest-model (H_3) based on the newly updated model returned by the earlier call to the *LoE-AIM* method. In these PLAY-EPISODES number of plays, if ever the opponent took different actions for the same state, H_2 is set as stochastic, else H_2 is set as deterministic. Finally the framework makes a conservative check to see whether the assumptions it made after the first STORE-EPISODES number of plays also hold after the next



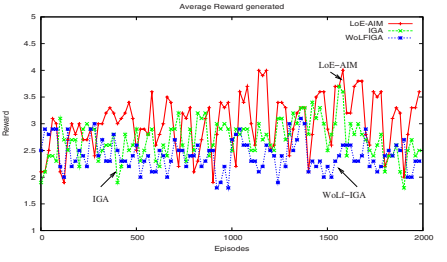
(a) Comparison with the two version of Godfather in PD



(b) Comparison with IGA and WoLF-IGA in PD



(c) Comparison with the two version of Godfather in Chicken



(d) Comparison with IGA and WoLF-IGA in Chicken

Fig. 7. *LoE-AIM-repeated* Results

PLAY-EPISODES number of plays. If the assumptions hold, it stores (replaces, if it updated a former stored model) the newly generated model in the pool. The whole process repeats with every new EPOCH of play.

For experimental evalauation of *LoE-AIM-repeated*, we restrict the set of opponents to the two versions of the Godfather together with IGA and WoLF-IGA. The opponents we choose give a fair representative mix of the targeted class that *LoE-AIM-repeated* is designed to exploit. Figure 7 provides a comparative picture of the results achieved by the *LoE-AIM-repeated* in the game of PD and Chicken respectively. As base case results, we also provide the results achieved by each of the opponent approaches had they been the approach employed by the player. We break the results in each game in two individual plots for clarity of expression, one comparing the performance of *LoE-AIM-repeated* with the deterministic opponents (two versions of Godfather) and the other comparing the same with the MAL opponents (IGA nd WoLF-IGA). We tested our approach for different values of STORE-EPISODES and PLAY-EPISODES, and finally decided to fix them at 20 and 80 respectively. As part of our future work, we would like to have a theoritical bound on the number of episodes we need to explore (STORE-EPISODES) to get a reasonable approximation of the model. The simulation has been run for 20 EPOCHs thereby resulting in a run of 2000 episodes in total. After every EPOCH a new opponent is chosen randomly. The results have been averaged over 10 instantiations of the start state. In both the plots, *LoE-AIM* does better than the benchmark opponents. An interesting thing to

note is that the *LoE-AIM* learning plot has spikes after every 100 episodes. After every 100 episodes, the learner explores for 20 episodes to build an approximate model of the new opponent. But once it builds the model, it matches it with a stored model and starts using the knowledge it learned from past interactions with the opponent.

6 Conclusion

In this paper we introduced a general mechanism for learning against memory-bounded adversaries. Our algorithm *LoE-AIM* either explores to gather new information about the opponent or converges to the best response to the partially learned opponent strategy. We showed detailed results in the games of PD and Chicken and further backed our claims with results averaged over 6 games with ordinal payoffs and multiple Nash equilibria and each player favoring a different Nash equilibrium. Our results show that *LoE-AIM* generates higher average rewards than existing MAL approaches against the same set of opponents. We then introduced a mechanism that enables online learning based on epochs of play against similar opponents by mining of learned knowledge about the opponent and using it to seed learning when faced against the same opponent in future interactions.

This research suggests several possible directions for future work. First, the algorithms presented are limited to targeting opponents with bounded memory. It would be natural to try to extend the results to opponents that fall in other parts of the taxonomy shown in Figure 1. For example, it would be interesting to see how *LoE-AIM* can be generalized to account for opponents whose next step strategy depends on the entire history of play (not just k-Markov as assumed in this paper). An important challenge in that direction would be to devise a compact finite state representation that captures the history of play.

Acknowledgement. This research was supported in part by NSF CAREER award IIS-0237699.

References

1. Banerjee, B., Peng, J.: Efficient learning of multi-step best response. In: AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 60–66. ACM Press, New York (2005)
2. Banerjee, B., Peng, J.: Rv&#963(t): a unifying approach to performance and convergence in online multiagent learning. In: AAMAS 2006: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 798–800. ACM Press, New York (2006)
3. Bowling, M.: Convergence and no-regret in multiagent learning. In: Neural Information Processing Systems, vol. 17. MIT Press, Cambridge (2005)
4. Bowling, M., Veloso, M.: Convergence of gradient dynamics with a variable learning rate. In: Proc. 18th International Conf. on Machine Learning, pp. 27–34. Morgan Kaufmann, San Francisco (2001)

5. Bowling, M.H., Veloso, M.M.: Rational and convergent learning in stochastic games. In: IJCAI, pp. 1021–1026 (2001)
6. Brafman, R.I., Tennenholtz, M.: R-max - a general polynomial time algorithm for near-optimal reinforcement learning. J. Mach. Learn. Res. 3, 213–231 (2003)
7. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI/IAAI, pp. 746–752 (1998)
8. Conitzer, V., Sandholm, T.: Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents, pp. 83–90 (2003)
9. Fudenberg, D., Levine, D.K.: The theory of learning in games. MIT Press, Cambridge (1999)
10. Greenwald, A., Jafari, A., Ercal, G., Gondek, D.: On no-regret learning, fictitious play, and nash equilibrium
11. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the 11th International Conference on Machine Learning (ML 1994), New Brunswick, NJ, pp. 157–163. Morgan Kaufmann, San Francisco (1994)
12. Nash Jr., J.F.: Equilibrium points in n-person games. In: Classics in game theory (1997)
13. Powers, R., Shoham, Y.: Learning against opponents with bounded memory. In: IJCAI, pp. 817–822 (2005)
14. Singh, S., Kearns, M., Mansour, Y.: Nash convergence of gradient dynamics in general-sum games, pp. 541–548
15. Stone, P., Littman, M.L.: Implicit negotiation in repeated games. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 96–105. Springer, Heidelberg (2002)
16. Vu, T., Powers, R., Shoham, Y.: Learning against multiple opponents. In: AAMAS 2006: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 752–759. ACM Press, New York (2006)

Scalable Feature Selection for Multi-class Problems

Boris Chidlovskii and Loïc Lecerf

Xerox Research Centre Europe
6, chemin de Maupertuis, F–38240 Meylan, France

Abstract. Scalable feature selection algorithms should remove irrelevant and redundant features and scale well on very large datasets. We identify that the currently best state-of-art methods perform well on binary classification tasks but often underperform on multi-class tasks. We suggest that they suffer from the so-called accumulative effect which becomes more visible with the growing number of classes and results in removing relevant and unredundant features. To remedy the problem, we propose two new feature filtering methods which are both scalable and well adapted for the multi-class cases. We report the evaluation results on 17 different datasets which include both binary and multi-class cases.

1 Introduction

Feature selection is the technique of selecting a subset of relevant features for building robust learning models[1,4,5,7,8,10]. The feature selection targets removing most *irrelevant* and *redundant* features from the data, by which it helps improve the performance of learning models by

1. alleviating the effect of the curse of dimensionality,
2. enhancing generalization capability,
3. speeding up learning process and
4. improving model interpretability. [1]

Existing supervised feature selection algorithms fall into one of the three following groups: filter models, wrapper models or hybrid models [1,7,9,10]. The *filter model* relies on general characteristics of the training data to select some features without involving any learning or mining algorithm, therefore it does not inherit any bias of a learning algorithm. The *wrapper model* requires one predetermined learning algorithm and uses its performance to determine which features to select. The wrapper model needs to learn a classifier and may suffer from the bias problem [13]. It tends to give superior performance as it finds features better suited to the predetermined learning algorithm, but it also tends to be more computationally expensive. When the number of features becomes very large, the filter model is usually a choice due to its computational efficiency. Algorithms in a *hybrid model* try to combine the advantages of both models [11,1,14].

[1] http://en.wikipedia.org/wiki/Feature_selection

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 227–240, 2008.

In this work we focus on the filter model and propose new feature selection algorithms which effectively remove irrelevant and redundant features and are computationally efficient. The entropy-based algorithms we propose are well suited for a variety of problems in different domains where data instances are characterized by a mixture of nominal, categorical and numeric features. Moreover, they play the central role in the *automatic feature extraction* from massive datasets where the automatic analyzer can generate thousands or billions of features. A part of the generated features is irrelevant to the classification task; even a larger part of features is redundant.

Our work is motivated by the automatic feature extraction from the image, OCR- and layout-oriented documents where a very large number of features are produced in order to capture different characteristics of visual and textual data elements. The problem we face is the selection from the very large feature set to build robust and accurate learning models.

Within the filter model, one can distinguish between feature weighting algorithms and subset selection algorithms, based on whether they evaluate the goodness of features individually or through feature subsets. *Feature weighting algorithms* assign weights to features individually and rank them using their relevance to the target concept. There are a number of different definitions on feature relevance in machine learning literature [7]. A feature is good and thus will be selected if its weight of relevance is greater than a threshold value. Relief [6] is one of the well known algorithms relying on relevance evaluation. Its key idea is to estimate the relevance of features according to how well their values distinguish between the instances of the same and different classes.

Subset selection algorithms search through candidate feature subsets guided by a certain evaluation measure which captures the goodness of each subset. An optimal (or near optimal) subset is selected when the search stops. Evaluation measures that have been shown effective in removing both irrelevant and redundant features include the consistency measure [2] and the correlation measure [5]. Combined with different search strategies, such as exhaustive, heuristic, and random search these evaluation measures form different algorithms [2,5,10]. The time complexity is exponential in terms of data dimensionality for exhaustive search and polynomial for heuristic search. However, with the quadratic or higher time complexity in terms of dimensionality, existing subset selection algorithms do not scale well when dealing with very large datasets.

To overcome the problems and to meet the demand for feature selection for high dimensional data, a novel *class of scalable algorithms* has recently appeared [11,10,15]; these algorithms can effectively identify both irrelevant and redundant features with less time complexity than subset selection algorithms. The scalability requires that the feature filtering algorithm allows a little overhead beyond the minimal cost of scanning the dataset with N features and M data items. Since a dataset may be very large in both N and M, scalable algorithms have a complexity sub-quadratic in NM.

Existing scalable algorithms show up a good performance in binary classification tasks but may cause the performance loss when the classification problem

addresses multiple classes. In the following sections, we try to analyze the reasons of this loss and propose two new methods for the scalable feature selection. The algorithms allow us to achieve the two goals: they keep the algorithms scalable and perform well on both binary and multi-class tasks.

2 Scalable Feature Selection

One common approach to measure the correlation between two random variables is based on the linear correlation coefficients. However, linear measures are unable to capture correlations that are not linear in nature; such is the case of categorical and nominal features. Another limitation is that the calculation requires all features contain numerical values. To overcome these shortcomings, the other approach is to use entropy-based measures of the uncertainty of a random variable, where the entropy of a variable Y is defined as $H(Y) = -\sum_y P(y)\ \log_2 P(y)$.

Given two random variables Y and X, we are interested in measuring the information that one variable has about another. The so-called *mutual information* $I(Y;X)$ is given by the Kullback-Leibler (KL) divergence between a joint distribution $P(Y,X)$ and the product of its marginal distributions $P(Y)P(X)$:

$$\begin{aligned} I(Y;X) &= D_{KL}(P(Y,X)||P(Y)P(X)) \\ &= \textstyle\sum_{x,y} P(y,x)\ \log_2 \frac{P(y,x)}{P(y)P(x)} \\ &= \textstyle\sum_{x,y} P(y,x)\ \log_2 P(y|x) - \sum_{x,y} P(y,x)\ \log_2 P(y) \\ &= \textstyle\sum_{x,y} P(y|x)P(x)\ \log_2 P(y|x) - \sum_y P(y)\ \log_2 P(y) \\ &= H(Y) - H(Y|X), \end{aligned} \tag{1}$$

where $P(x)$ is the prior probabilities for $x \in X$ values, $P(y|x)$ is the posterior probabilities of $y \in Y$ given the values of $x \in X$ and the conditional entropy $H(Y|X)$ is the entropy of Y after observing values of X:

$$H(Y|X) = \sum_x P(x) \sum_y P(y|x)\ \log_2 P(y|x). \tag{2}$$

When applied to the feature selection, the mutual information is the amount by which the entropy of one variable decreases from the knowledge of another variable. The mutual information (called also *information gain* [8]) is symmetrical for two variables Y and X. Since it is often biased in favor of features with more values, the values have to be normalized to ensure they are comparable and have the same affect. Therefore, *symmetrical uncertainty* is used instead. Defined as

$$SU(Y,X) = \frac{2\ I(Y;X)}{H(Y) + H(X)}, \tag{3}$$

it does compensate for the bias and normalizes its values to the range [0, 1], where the value 0 indicates that Y and X are independent and the value 1 indicates that the value of either one completely predicts the value of the other.

2.1 Markov Blankets for Redundant Features

An efficient feature selection method should cope with irrelevant and redundant features. After years of intensive research, a consensus has been achieved on determining irrelevant features [7,3,8,15]. Instead, the major difficulty remains around the redundant features. Indeed, unlike irrelevant features, not all of them should be removed while keeping all of them often causes the accuracy loss and overfitting. Therefore, this poses the problem of selecting an optimal subset among redundant features.

Let we dispose a dataset S with feature set F and class set Y. A relevant feature $F_i \in F$ is *redundant* if it has a Markov blanket in F [15], where a *Markov blanket* for feature F_i is a feature subset $\mathcal{M}_i \in F$ which subsumes the information feature F_i has about target Y and all other features in $F - \mathcal{M}_i - \{F_i\}$:

$$P(F - \mathcal{M}_i - \{F_i\}, Y | F_i, \mathcal{M}_i) = P(F - M_i - \{F_i\}, Y | \mathcal{M}_i). \tag{4}$$

The *Markov blanket filtering* [3] is a backward elimination procedure, which at any step removes F_i if there exists a Markov blanket for F_i among the features remaining in F. The process guarantees that a feature removed at previous steps will be still redundant later and removing a feature at later steps will not render the previously removed features necessarily to be included in the optimal subset F_{opt}. Finding the exact Markov blanket for a feature requires an exhaustive enumeration of feature subsets which makes the exact Markov blanket filtering computationally unacceptable for any important feature set.

Scalable filtering algorithms do approximate the Markov blanket filtering. Similarly to the exact feature subset selection, where only relevant features having no Markov blanket are selected, in the *approximate feature subset selection*, one selects the relevant features having no approximate Markov blanket.

The scalable algorithms essentially include two steps where the first step determines and removes irrelevant features and the second step copes with redundant features. Below, we present a generic filtering-based feature selection algorithm. Without loss of generality, we assume using SU measures for removing irrelevant features in F. All methods discussed in this section and developed in the following sections are instances of Algorithm 1.

The algorithm initially calculates SU values for each feature $F_i \in F$, then selects those which are superior to the irrelevance threshold δ and ranks the selected ones in the decreasing order. Steps (4)-(10) take $O(N\ M + N \log N)$ time, where N is the number of features, $N = |F|$, and M is the number of data items.

In the algorithm, the goodness criteria GOOD() guides the (greedy) process of selecting redundant features among relevant ones. Different algorithms may use different goodness criteria which in turn may results in different computational complexities. The complexity of the second part (steps (11)-(21)) depends on the number of feature pairs the GOOD () criteria is applied on. In the worst case, the criteria removes no features from F_{cand} which leads to the $O(N^2\ G)$ worst case complexity, where G is the complexity of the GOOD criteria for a feature pair, which is $O(M)$ in the general case. Moreover, if, at each iteration (12)-(19)

Algorithm 1. Scalable Feature Filtering Algorithm

INPUT: training dataset S with class set Y and feature set $F = \{F_i\}, i = 1, \ldots, N$, irrelevance threshold δ, goodness criteria GOOD()
OUTPUT: optimal feature subset F_{opt}
for $i = 1, \ldots, N$ **do**
  calculate $SU(Y, F_i)$
  if $SU(Y, F_i) > \delta$ **then**
    append F_i to F_{rel}
  end if
end for
Order features F_i in F_{rel} in the decreasing order of $SU(Y, F_i)$
F_{pivot} = getFirst (F_{rel})
while F_{pivot} is not null **do**
  add F_{pivot} to F_{opt}
  F_{cand} = getNext (F_{rel},F_{pivot})
  while F_{cand} is not null **do**
    if not GOOD(F_{pivot}, F_{cand}) **then**
      remove F_{cand} from F_{rel}
    end if
  end while
  F_{pivot} = getNext (F_{rel},F_{pivot})
end while
return F_{opt}

of Algorithm 1, F_{pivot} removes αN features F_{cand} where $0 < \alpha < 1$, then it can be shown that the expected complexity is $O(MGN \log N)$ [15].

2.2 FCBF and Accumulation Effect

The *Fast Correlation-Based Filtering* (FCBF) is currently the best scalable algorithm for feature selection [15,10]. Among various methods for *approximated Markov blankets* developed so far, the FCBF offers the best trade-off between the efficiently and effectiveness. The FCBF is based on the following Markov blanket approximation rule:

Definition 1. *Feature F_1 is an approximate Markov blanket for feature F_2 if $SU(Y, F_1) \geq SU(Y, F_2)$ and $SU(F_1, F_2) \geq SU(Y, F_1)$.*

The FCBF is an instance of Algorithm 1 where the GOOD () verifies the approximation condition given by Definition 1. Since GOOD () takes $O(1)$ time for a feature pair, the FCBF expected complexity is $O(MN \log N)$ which ensures its high scalability.

When we deployed the FCBF algorithm on a number of various datasets, we empirically identified that the FCBF works often poorly in multi-class cases. Indeed, for all datasets (presented in detail in Section 5), we evaluated the impact of FCBF on the classification accuracy. In each case, we trained (using the cross validation with 5 folds) a classifier, first with the initial feature set F, and

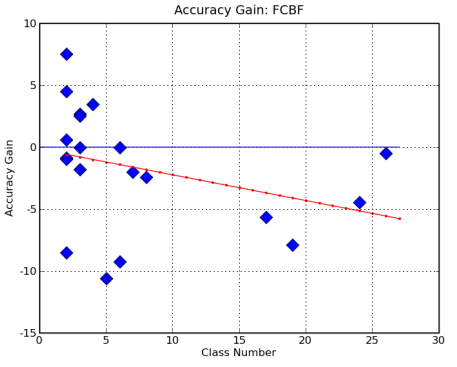


Fig. 1. The FCBF for different class numbers

then with the feature subset F_{opt} selected by the FCBF. For all datasets, we measured the accuracy gain as the difference between the second and first measures. Figure 1 plots the accuracy gain against the number of classes, denoted as $|Y|$, as well as the linear least squares fitting. The figure shows a statistically important correlation between the accuracy loss and the class number. All cases where FCBF does improve the accuracy correspond to small $|Y|$, with 2, 3 or 4 classes. And vice versa, for the large $|Y|$ cases (5 and more), the accuracy loss may achieve 10%.

We have carefully analyzed all cases where the FCBF causes the accuracy loss. Table 1 presents an artificial example which is abstracted from our analysis. The dataset includes 10 items with two features, F_1 and F_2, and three classes, y_0, y_1 and y_2. Feature F_1 correlates with class value y_1; F_2 correlates with y_2 and none correlates with y_0. It is easy to verify that both F_1 and F_2 are relevant to Y and moreover $SU(Y, F_1) = SU(Y, F_2)$. None of the two features is a Markov blanket of another because the information subsumption holds for some y and not for entire Y in (4). However, FCBF approximation rule given by Definition 1 would mistakenly eliminate one of the two features as redundant one.

The problem seems to be in the way the FCBF approximates the Markov blankets. The uncertainty reduction may vary from one class to another, but the FCBF uses the uncertainty value for entire class set Y to make a selection decision. Actually, our hypothesis is the FCBF suffers from the *accumulation effect* when the uncertainty reduction for different classes are summed up when verifying the Markov blanket condition. The effect is hardly visible for binary classification cases, but it becomes important for multi-class cases where the FCBF tends to remove features which are not redundant at all.

This analysis does suggest that the redundancy of one feature F_i with respect to another feature F_j should be verified for each class $y \in Y$ and we should consider the redundancy correlation on the per-class basis. In the following section, we propose two new methods to remedy the problem, without compromising the efficiency and scalability of the filtering algorithm.

Table 1. Artificial example where the FCBF fails

Y	F_1	F_2
y_0	0	0
y_0	0	0
y_0	0	0
y_0	1	1
y_0	1	1
y_0	1	1
y_1	0	0
y_1	0	1
y_2	0	0
y_2	1	0

3 Fast Targeted Correlation-Based Filter

In order to keep low the computational complexity and to take into account the per-class uncertainty, we first propose the *Fast Targeted Correlation Based Filtering* (FtCBF) method. It modifies the FCBF algorithm by adding an extra condition to the goodness criteria in order to avoid, or at least to minimize, the accumulation effect.

Class $y \in Y$ is called *targeted* by a feature F_i if there exist at least two items in S with different values of F_i with the class y. In terms of the conditional probability, y is targeted by feature $F_i \in F$ if $H(F_i|Y = y) > 0$. The set of classes $y \in Y$ targeted by feature F_i is denoted $S_Y(F_i)$, $S_Y(F_i) = \{y|H(F_i|Y = y) > 0\}$.

We modify the FCBF in order accommodate it to the multi-class cases. We relax the too strong condition imposed by FCBF, by verifying that the target set of the pivot feature subsumes the target set of the candidate, $S_Y(F_{pivot}) \supseteq S_Y(F_{cand})$. Thus the FtCBF applies on the following Markov blanket approximation rule:

Definition 2. *(FtCBF rule). Feature F_1 is an approximate Markov blanket for feature F_2 if $SU(Y, F_1) \geq SU(Y, F_2)$, $SU(F_1, F_2) \geq SU(Y, F_1)$ and $S_Y(F_1) \supseteq S_Y(F_2)$.*

In the artificial example in Table 1, we have $S_Y(F_1) = \{y_0, y_1\}$ and $S_Y(F_2) = \{y_0, y_2\}$. Since none of the two target sets subsumes another, the new approximation rule would retain both features. We will see in the evaluation section, how this modification allows to resist to the accumulation effect and to filter out redundant features without the accuracy loss.

The algorithm for FtCBF is obtained by setting accordingly the GOOD () criteria in Algorithm 1. Since the first term of the FtCBF rule is explicitly realized by the first part of the algorithm, the criteria for FtCBF includes the second and third terms, GOOD(F_{pivot}, F_{cand}) $=SU(F_{pivot}, F_{cand}) \geq SU(Y, F_{cand})$ *and* $S_Y(F_{pivot}) \supseteq S_Y(F_{cand})$.

4 Fast Class Correlation Filter

Another method to avoid the accumulation effect is to analyze the contribution of each class to the conditional entropy $H(Y|X)$ and the symmetric uncertainty $SU(Y,X)$ and to take them in consideration when building the Markov blanket approximation. Here we propose the *Fast Class Correlation Filtering* (FCCF) method which use both per-class uncertainty and feature correlation to build the Markov blanket approximation. We first rewrite the information gain in (1) on the per-class basis as follows:

$$\begin{aligned} I(Y;X) &= H(Y) - H(Y|X) \\ &= \sum_y \sum_x P(x)P(y|x)\ \log_2 P(y|x) - \sum_y P(y)\ \log_2 P(y) \\ &= \sum_{y\in Y} I(Y=y;X), \end{aligned} \tag{5}$$

where $I(Y=y;X)$ is the contribution of class $y \in Y$ to the aggregated information gain $I(Y;X)$, $I(Y=y;X) = \sum_x P(x)P(y|x)\ \log_2 P(y|x) - P(y)\ \log_2 P(y)$. After the normalization, the symmetric uncertainty SU may be equally decomposed on the per-class principle. For two random variables Y and X, we have $SU(y,X) = \frac{H(Y=y)-I(Y=y;X)}{H(Y)+H(X)}$ where $H(Y=y) = p(y)\log_2(y)$ and therefore

$$SU(Y,X) = \sum_{y\in Y} SU(y,X). \tag{6}$$

A relevant feature F_i *strictly subsumes* a relevant feature F_j if $SU(y,F_i) \geq SU(y,F_j)$, for all $y \in Y$. By adding the constraint on the feature correlation $SU(F_i,F_j)$, we obtain the following Markov blanket approximation rule:

Definition 3. *(FCCF rule). Feature F_1 is an approximate Markov blanket for feature F_2 if for any $y \in Y$, $SU(Y=y,F_1) \geq SU(Y=y,F_2)$ and $SU(F_1,F_2) \geq SU(Y,F_1)$.*

To apply the FCCF approximation rule and to preserve the scalability, we have to make some minor changes in Algorithm 1. In step (4)-(8), we extend the calculation of $SU(Y,F_i)$ by calculation of per-class uncertainty vector $[SU(y_1,F_i),\ldots, SU(y_{|Y|},F_i)]$. The ordering of values in step (9) would be then done by the decreasing values of one class, say y_1. The second part of Algorithm 1 is modified accordingly, in order to compare uncertainty vectors $[SU(y_1,F_i),\ldots,SU(y_{|Y|},F_i)]$ and $[SU(y_1,F_j),\ldots,SU(y_{|Y|},F_j)]$ for a pair of features $F_i,F_j \in F$. This need to compare uncertainty vectors leads to the change in the method complexities with respect to FCBF. The FCCF worst case becomes $O(M|Y|N^2)$ and the average case is $O(M|Y|N\log N)$.

This class-by-class uncertainty comparison is naive and quite satisfactory for dozens and hundreds of classes. If the number of classes $|Y|$ accounts for thousands, some sophisticated structures might implemented to reduce the average case to $O(M\log|Y|N\ \log N)$ [12].

5 Evaluation Results

In this section we report results of evaluation tests aimed at verifying how good new feature selection methods are in cases of large numbers of features, instances

and classes. All evaluations are performed in terms of classification accuracy and dimensionality degree.

For evaluation tests, 17 different datasets have been proposed. First, we selected 15 datasets available from UCI Machine Learning repository [2]. Among existing UCI datasets, we prefered ones covering different application domains and, importantly, representing a high variability of class numbers. Since the UCI Machine Learning repository is essentially dominated by 2- and 3-class datasets, we additionally included two multi-class datasets from another source. These two datasets have been created in the framework of the VIKEF European Integrated Project [3] for enabling the integrated development of semantic-based information, content, and knowledge management systems. The first, CPO dataset is a collection of data items extracted from PDF documents and annotated with 8 metadata classes, including `title, author, organization` and `address`. The second, bizCard dataset is composed of data items extracted from personal business cards where each paper-based business card was scanned, OCR-ed and annotated with different metadata classes. For all 17 datasets included in the evaluation tests, Table 2 reports the number of data items, classes and features.

Table 2. Feature selection results for 17 test datasets

Collection			**Initial Set**			**FCBF**			**FtCBF**			**FCCF**		
UCI	$\|Y\|$	M	F	**ME**	**DT**	F_{opt}	**ME**	**DT**	F_{opt}	**ME**	**DT**	F_{opt}	**ME**	**DT**
breast	2	699	9	94.7	**94.98**	8	**95.28**	94.85	8	**95.28**	94.85	8	**95.28**	94.85
credit-a	2	690	15	83.48	85.55	3	82.61	83.51	10	83.91	85	11	**84.57**	**85.51**
heart-c	2	303	13	74.97	76.24	5	**82.51**	**80.37**	10	79.89	**80.37**	11	81.11	**80.37**
hepatitis	2	155	19	79.35	78.52	3	**83.87**	**80.52**	8	80.64	**80.52**	8	82.45	**80.52**
labor	2	57	16	94.54	**80.53**	3	86.01	77.48	9	94.54	77.48	9	**94.71**	77.48
mushroom	2	8124	22	**100**	**100**	5	99.03	99.02	5	99.03	99.02	5	99.03	99.02
balance-sc	3	625	4	**88.32**	78.11	4	**88.32**	**77.78**	4	**88.32**	**77.78**	4	**88.32**	**77.78**
iris	3	150	4	91.33	94.53	1	**94**	**94.67**	4	91.33	94.33	4	91.33	**94.67**
splice	3	3190	61	92.98	93.88	22	**95.52**	**94.21**	28	93.98	94.16	31	94.13	93.26
waveform	3	5000	40	76.12	75.24	1	54.33	57.01	4	73.86	74.1	5	**76.44**	**76.19**
lymph	4	148	18	77.76	76.11	8	81.21	75.01	13	**85.28**	**76.43**	13	84.8	**76.43**
anneal	5	898	38	98.55	**98.62**	5	87.97	**98.62**	19	**99.00**	98.33	20	97.66	98.48
autos	6	205	25	58.05	**79.66**	3	58.05	70.59	24	55.12	69.82	22	**59.13**	72.59
glass	6	214	9	48.79	67.45	1	39.56	54.44	8	47.31	64.11	9	**48.79**	64.58
zoo	7	101	17	95.0	93.29	7	93.0	91.39	10	**95.0**	**94.96**	10	**95.0**	**94.96**
soybean	19	683	35	85.35	**90.57**	9	77.46	80.13	21	**86.09**	87.92	22	85.87	89.91
audiology	24	226	69	**78.22**	77.57	24	73.77	75.9	27	77.12	**78.68**	27	77.12	**78.68**
letter	26	20000	16	**82.58**	87.25	10	80.16	86.77	10	80.16	86.77	11	80.21	**87.66**
CPO	8	7612	42	93.83	90.54	11	91.43	92.04	19	94.18	**93.45**	23	**95.2**	93.22
Bizcard	17	3620	135	**71.48**	**71.65**	21	65.85	62.42	45	68.29	69.63	38	71.24	69.97
Average	6.3		31.1	83.31	**84.37**	7.6	81.57	81.05	14.5	83.57	83.79	14.7	**84.32**	84.02

[2] http://mlearn.ics.uci.edu/MLRepository.html
[3] http://www.vikef.net/

We evaluated the performance of different filter-based feature selection methods by using two well-known classification methods, C4.5 decision tree (DT) and maximum entropy (ME) ones. For the former, we used the Weka package [4] which has its version of C4.5 known as `weka.classifiers.trees.J48`. In all tests, the C4.5 confidence factor was set by default to 0.25. For the maximum entropy classifier, we used the Maximum Entropy modeling toolkit for Python language [5].

Feature selection evaluation. In the first series of tests, we compare the accuracy of classification models trained with initial feature set F, as well as with the feature subsets selected by three methods, FCBF, FtCBF and FCCF. All tests are run using the cross validation protocol. We used 5 folds for all selected datasets. For each folding, a feature selection method is applied to the training set; it returns a feature subset which is used to train a model from the training set. [6] The average over all foldings is reported as the model accuracy for the dataset.

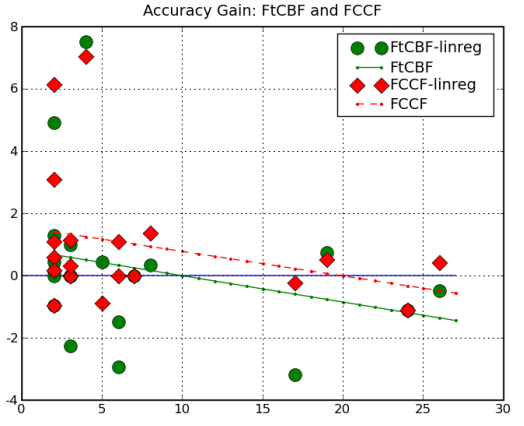


Fig. 2. Accuracy gain versus the class number: FtCBF and FCCF methods

Table 2 reports the evaluation results for the initial feature sets and the feature subsets selected by FCBF, FtCBF and FCCF. For each dataset [7] and each method, the table reports the size of optimal feature subset $|F_{opt}|$ and the accuracy for both C4.5 and maximum entropy classification models. Both

[4] http://www.cs.waikato.ac.nz/m̃l/weka/

[5] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

[6] Using the same training data in both feature selection and classifier training may cause the so-called *feature selection bias* [9]. We initially intended to avoid this bias. Unfortunately, a further split of training data would severely penalize the small datasets. Thus we accepted the proposed schema as far as all feature selection algorithms are evaluated in the same conditions.

[7] The UCI datasets are ordered by the increasing number of classes $|Y|$.

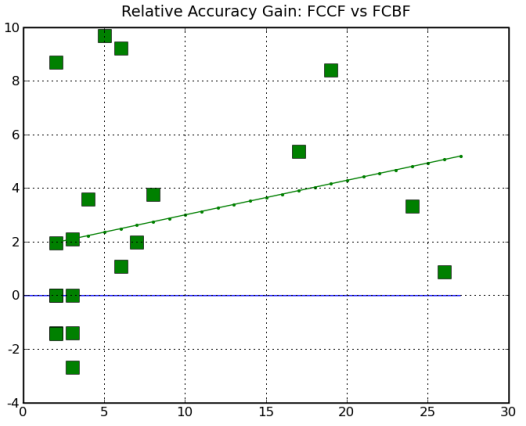


Fig. 3. Relative accuracy gain: FCCF versus FCBF

FtCBF and FCCF behave well on all the datasets, using either of two classification methods. They both appear less sensible to the number of classes and show no significant difference between collections with 2, 3, 4 and more classes.

By analogy with Figure 1, Figure 2 plots the accuracy gain and the linear minimar square fitting for all datasets, using FtCBF and FCCF methods [8]. First, the performance of new methods are comparable to the FCBF on 2- and 3-class datasets. Instead, they take an advantage over the FCBF on multi-class datasets.

To show it explicitly, Figure 3 combines the results presented in Figures 1 and 2. It shows the *relative accuracy gain* against the class number. Here, the relative gain is given by the difference in accuracy between FCCF and FCBF. The advantage of FCCF over FCBF grows as the class number increases.

Finally, we mention another aspect of test results reported in Table 2. Figure 4 plots the accuracy gain against the *feature reduction ratio* given by the fraction of features from the initial feature set F selected by a given method, $|F_{opt}|/N$. As one can observe, the FCBF conducts an aggressive policy of redundant feature removal which might be not justified and thus leading to the accuracy loss on some datasets, in particular, the multi-class ones. Instead, FtCBF and FCCF impose additional conditions for removing a feature as redundant one. As result, they are more modest at removing features from the initial set, this however permits to avoid the accuracy loss.

Class shrinkage. In order to extend the comparison of feature selection methods in multi-class tasks, we have undertaken the second series of experiments. The *class shrinkage* procedure was invented in order to test the feature selection methods over the varying number of classes.

The class shrinkage is implemented in the following way. In a dataset with $|Y|$ classes, a pair of distinct classes is randomly selected and all items of one

[8] The plot shows the accuracy values obtained with the maximum entropy models. Results for C4.5 models are not presented here but have a very similar shape.

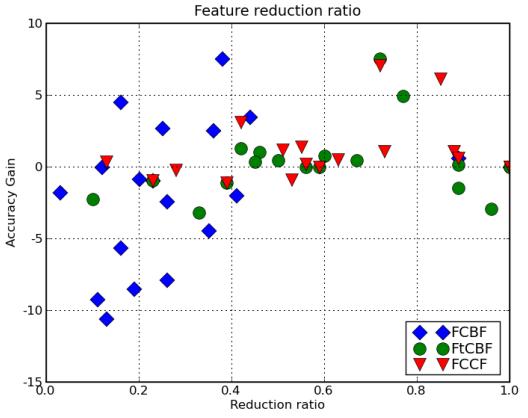


Fig. 4. Accuracy gain versus the feature reduction ratio

class are relabeled with another one. This step reduces the class number by one. The random shrinkage step is repeated $|Y| - 2$ more times, thus producing a sequence of datasets with diminishing number of classes, $k = |Y|, |Y| - 1, \ldots, 2$. For each value k in the sequence, we run all feature selection methods and measure the accuracy of classification models with the initial (Basic) feature set and feature subsets selected by FCBF, FtCBF and FCCF. The shrinkage experiment is repeated 10 times and the average values over all runs are reported as the model accuracy together with the standard deviation.

Below we present results of class shrinkage experiments for multi-class datasets in Table 2. Figures 5 shows Basic, FCBF, FtCBF and FCCF plots for the **soybean** dataset. The right extreme of all plots corresponds to the initial dataset; it was analyzed in Table 2. All three feature selection methods get close to the Basic accuracy on the left extreme, when $k = 2$. Among the three feature selection

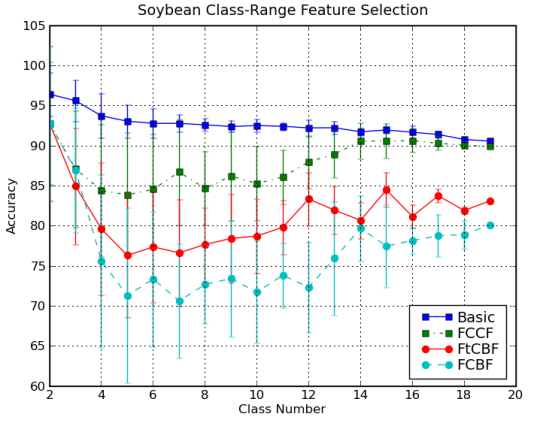


Fig. 5. Class shrinkage with the soybean dataset

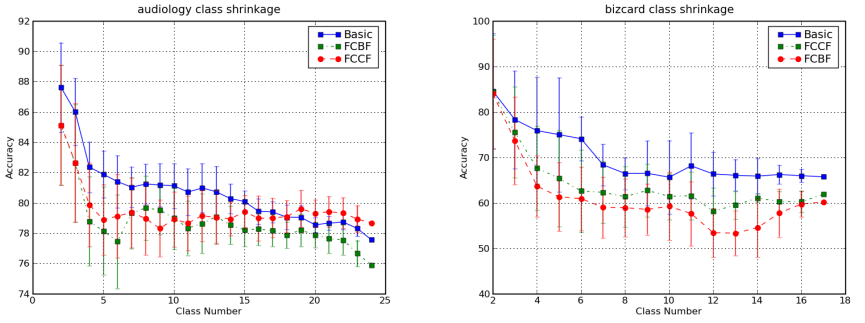


Fig. 6. Class shrinkage with the a) audiology and b) bizcard datasets

methods, both FtCBF and FCCF outperform FCBF. We note very important standard deviation values for all plots, in particular for intermediate class numbers. This is explained by the randomness of class replacement and a large variety of datasets obtained by such a replacement.

Figure 5 reports the **audiology** and **bizcard** for the initial feature set and FCBF and FCCF methods [9]. Both new methods tend to behave well in class shrinkage experiments on other multi-class datasets. The only exception is the *letter* dataset, where all feature selection methods show no significant difference.

6 Conclusion

We have proposed two new scalable feature selection methods which guarantee a good tradoff between efficiency and effectiveness for multi-class cases. Both new methods outperform the state-of-art FCBF method which may suffer from the accumulation effect. Either method proposes an approximate Markov blanket rule which relaxes the FCBF's aggressive criteria for removing redundant features. The evaluation on 17 datasets demonstrate that this relaxation pays off when the number of classes becomes important.

All scalable filtering algorithms remove as irrelevant any feature whose uncertainty value with respect to the class variable falls under the irrelevance threshold. This unfortunately ignores all possible interactions between the features [16]. Improving the filtering algorithms that takes into consideration the feature interaction but does not hurt the scalability represents a challenging problem.

References

1. Das, S.: Filters, wrappers and a boosting-based hybrid for feature selection. In: Proc. 18th Intern. Conf. Machine Learning, pp. 74–81 (2001)
2. Dash, M., Liu, H.: Feature selection for clustering. In: Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 110–121 (2000)

[9] The FtCBF plots are not reported here; they are close to FCCF plots for both datasets.

3. Koller, D., Sahami, M.: Toward optimal feature selection. In: ICML 1996: Proc. 13th International Conference on Machine Learning, pp. 284–292. Morgan Kaufmann Publishers Inc., San Francisco (1996)
4. Geng, X., Liu, T.-Y., Qin, T., Li, H.: Feature selection for ranking. In: SIGIR 2007: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 407–414. ACM, New York (2007)
5. Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In: ICML 2000: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 359–366 (2000)
6. Kira, L., Rendell, L.: The feature selection problem: Traditional methods and a new algorithm. In: Proc. 10th National Conf. Artificial Intelligence, pp. 129–134 (1992)
7. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–323 (1997)
8. Kononenko, I.: Estimating attributes: Analysis and extensions of relief. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
9. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Chapman and Hall/CRC, Boca Raton (2007)
10. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering 17(4), 491–502 (2005)
11. Ruiz, R., Aguilar-Ruiz, J.S., Riquelme, J.C.: Efficient incremental-ranking feature selection in massive data. In: Liu, H., Motoda, H. (eds.) Computational Methods of Feature Selection, pp. 147–166. Chapman and Hall/CRC, Boca Raton (2007)
12. Samet, H.: Foundations of Multidimensional And Metric Data Structure. Morgan Kaufmann Publishers, Reading (2006)
13. Singhi, S.K., Liu, H.: Feature subset selection bias for classification learning. In: ICML 2006: Proceedings of the 23rd international conference on Machine learning, pp. 849–856. ACM, New York (2006)
14. Xing, E.P., Jordan, M.I., Karp, R.M.: Feature selection for high-dimensional genomic microarray data. In: ICML 2001: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 601–608. Morgan Kaufmann Publishers Inc., San Francisco (2001)
15. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. 5, 1205–1224 (2004)
16. Zhao, Z., Liu, H.: Searching for interacting features. In: Proc. Intern. Joint Conf. Artificial Intelligence, IJCAI, pp. 1156–1161 (2007)

Learning Decision Trees for Unbalanced Data

David A. Cieslak and Nitesh V. Chawla

University of Notre Dame, Notre Dame IN 46556, USA
{dcieslak,nchawla}@cse.nd.edu

Abstract. Learning from unbalanced datasets presents a convoluted problem in which traditional learning algorithms may perform poorly. The objective functions used for learning the classifiers typically tend to favor the larger, less important classes in such problems. This paper compares the performance of several popular decision tree splitting criteria – information gain, Gini measure, and DKM – and identifies a new skew insensitive measure in Hellinger distance. We outline the strengths of Hellinger distance in class imbalance, proposes its application in forming decision trees, and performs a comprehensive comparative analysis between each decision tree construction method. In addition, we consider the performance of each tree within a powerful sampling wrapper framework to capture the interaction of the splitting metric and sampling. We evaluate over this wide range of datasets and determine which operate best under class imbalance.

1 Introduction

Data sets in which one class is particularly rare, but more important – termed unbalanced or unbalanced datasets – continue to be a pervasive problem in a large variety of supervised learning applications, ranging from telecommunications to finance to medicine to web categorization to biology. Typically sampling methods [1,2,3,4,5] are used for countering class imbalance.

Decision trees, particularly C4.5 [6], have been among the more popular algorithms that have been significantly helped by sampling methods for countering the high imbalance in class distributions [3,4,7]. In fact, the vast majority of papers in the ICML'03 Workshop on unbalanced Data included C4.5 as the base classifier. While it is understood that sampling generally improves decision tree induction, what is undetermined is the interaction between sampling and how those decision trees are formed. C4.5 [6] and CART [8] are two popular algorithms for decision tree induction; however, their corresponding splitting criteria — information gain and the Gini measure — are considered to be skew sensitive [9]. It is because of this specific sensitivity to class imbalance that use of sampling methods prior to decision tree induction has become a de facto standard in the literature. The sampling methods alter the original class distribution, driving the bias towards the minority or positive class[1]. Dietterich, Kearns, and

[1] Without loss of generality, we will assume that positive and minority class is the same.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 241–256, 2008.

Mansour [10] suggested an improved splitting criterion for a top down decision tree induction, now known as DKM. Various authors have implemented DKM as a decision tree splitting criterion and shown its improved performance on unbalanced datasets [9,11,12]. However, DKM has also been shown to be (weakly) skew-insensitive [9,11].

We posit that class imbalance is also a characteristic of sparseness in feature space, in addition to the skewed class distributions. Thus, it becomes important to design a decision tree splitting criterion that captures the divergence in distributions without being dominated by the class priors. To that end, we consider the Hellinger distance [13,14] as a decision tree splitting criterion, which we show to be skew-insensitive. We also demonstrate similarities between DKM and Hellinger distance, albeit Hellinger distance offers a stronger skew insensitivity. Finally, we consider the popular sampling methods and study their impact on the decision tree splitting criteria. *Does having a skew insensitive splitting criterion mitigate the need of sampling?*

Contributions: Our key contributions include the following: 1) Characterization of the Hellinger distance metric in data mining context as a skew insensitive decision tree splitting metric. 2) Analytical demonstration of the utility of proposed formulation of Hellinger distance using isometrics graphs. 3) A theoretical comparison between Hellinger distance and DKM. 4) A decision tree algorithm called HDDT incorporating the Hellinger distance as the tree splitting criterion. 5) Comparison of the effect of sampling on the decision tree splitting methods. We have used a total of 19 datasets, from UCI and real-world domains, with varying properties and skew in this study. We have also used statistical measures suggested by Demsar [15] to robustly compare the classifiers across multiple datasets. Note that we only used unpruned decision trees for all our experiments, irrespective of the splitting criterion used, as the previous work has pointed to the limitations of pruning for unbalanced datasets [16,17].

2 Hellinger Distance

Hellinger distance is a measure of distributional divergence [13,14]. Let (Θ, λ) denote a measurable space with P and Q as two continuous distributions with respect to the parameter λ. Let p and q be the densities of the measures P and Q with respect to λ. The definition of Hellinger distance can be given as:

$$d_H(P,Q) = \sqrt{\int_{\Omega} (\sqrt{P} - \sqrt{Q})^2 d\lambda} \tag{1}$$

This is equivalent to:

$$d_H(P,Q) = \sqrt{2\left(1 - \int_{\Omega} \sqrt{PQ}\, d\lambda\right)} \tag{2}$$

where $\int_{\Omega} \sqrt{pq}\, d\lambda$ is the Hellinger integral. Note the Hellinger distance does not depend on the choice of the dominating parameter λ. It can also be defined for

a countable space Φ, as $d_H(P,Q) = \sqrt{\Sigma_{\phi\in\Phi}(\sqrt{P(\phi)} - \sqrt{Q(\phi)})^2}$. The Hellinger distance carries the following properties: 1) $d_H(P,Q)$ is in $[0,\sqrt{2}]$. 2) d_H is symmetric and non-negative, implying $d_H(P,Q) = d_H(Q,P)$. Moreover, squared Hellinger distance is the lower bound of KL divergence.

In this paper, the P and Q in Equations 1 & 2 are assumed to be the normalized frequencies of feature values across classes. This allows us to capture the notion of "affinity" between the probability measures on a finite event space. If P = Q, then distance = 0 (maximal affinity) and if P and Q are completely disjoint then distance = $\sqrt{(2)}$ (zero affinity). This dictates the decision tree splitting criterion for separability between classes. We want to select a feature that carries the minimal affinity between the classes. Thus, the Hellinger distance can be used to capture the propensity of a feature to separate class distributions.

For application as a decision tree splitting criterion, we assume a countable space, so we discretize all continuous features into p partitions or bins. Assuming a two-class problem (class + and class −), let X_+ be class + and X_- be class −. Then, we are essentially interested in calculating the "distance" in the normalized frequencies aggregated over all the partitions of the two class distributions X_+ and X_-. The Hellinger distance between X_+ and X_- is:

$$d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{p} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}} \right)^2} \tag{3}$$

We postulate that this formulation is strongly skew insensitive as the prior does not influence the distance calculation. It essentially captures the divergence between the feature value distributions given the two different classes. There is no factor of class prior. We will show the effectiveness of this enumeration isometric plots.

2.1 Comparing Isometrics

Vilalta & Oblinger [18] proposed the use of isometric lines to define the bias of an evaluation metric by plotting contours for a given metric over the range of possible values. They presented a case study on information gain. While they did not produce isometrics under class skew, they note that "A highly skewed distribution may lead to the conclusion that two metrics yield similar generalization effects, when in fact a significant difference could be detected under equal class distribution. [18]" Subsequently, Flach [9] connected the isometric plots to ROC analysis, demonstrating the effects of true and false positives on several common evaluation metrics: accuracy, precision, and f-measure. In addition, he also presented isometrics for three major decision tree splitting criteria: entropy (used in information gain) [6], Gini index [8], and DKM [10]. Flach also established the effect on class skew on the shape of these isometrics [9].

We adopted the formulation of Flach in this paper, where the isometric plots show the contour lines in 2D ROC space, representative of the performance of different decision tree splitting criteria with respect to their estimated true

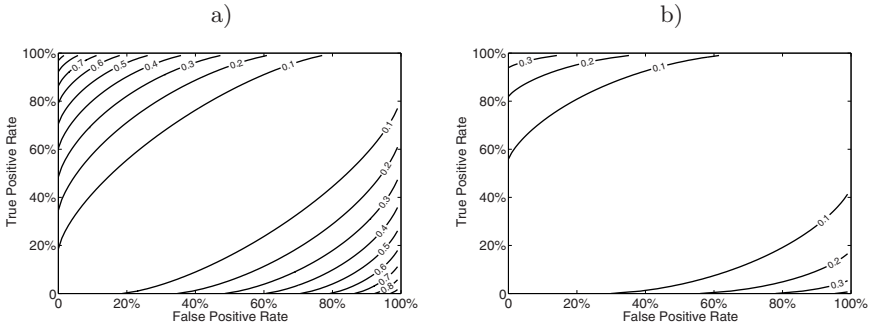


Fig. 1. Information gain isometrics for (+:-)=(1:1) in (a) and (+:-)=(1:10) in (b)

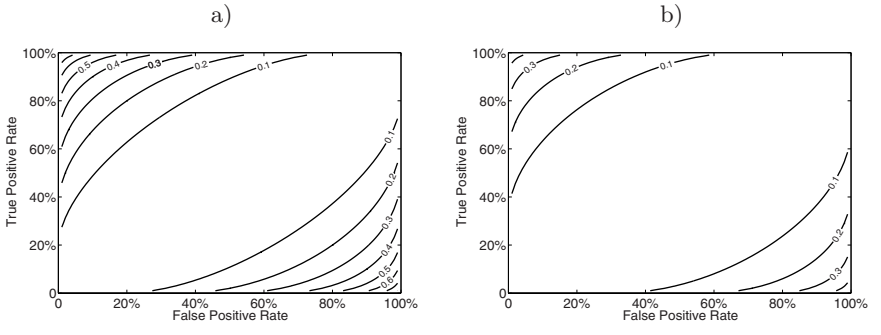


Fig. 2. DKM isometrics for (+:-)=(1:1) in (a) and (+:-)=(1:10) in (b)

and false positive rates, conditioned on the skew ratio ($c = \frac{neg}{pos}$). A decision tree split, for a binary class problem, can be defined by a confusion matrix as follows. A parent node will have POS positive examples and NEG negative examples. Assuming a binary split, one child will carry the true and false positive instances, and the other child will carry the true and false negative instances. The different decision tree splitting criteria, as considered in this paper, can then be modeled after this impurity (distribution of positives and negatives). Thus, in the isometric plots, each contour represents the combinations of true positives and false negatives that will generate a particular value for a given decision tree splitting criterion. For example, the 0.1 contour in Figure 1 (a) indicates that the value of information gain is 0.1 at (fpr, tpr) of approximately $(0\%, 20\%)$, $(20\%, 60\%)$, $(80\%, 100\%)$, $(20\%, 0\%)$, $(60\%, 20\%)$, $(100\%, 80\%)$, and all other combinations along the contour. In Figures 1 (a) & (b), information gain is observed as contours formed in ROC space under a $(+ : -)$ skew of $(1 : 1)$ and $(1 : 10)$, respectively. As the skewness increases, the isometrics become flatter and information gain will operate more poorly as a splitting criterion. Vilalta & Oblinger [18] and Flach [9] observed similar trends. Additionally, Flach [9] notes that DKM is (weakly) skew-insensitive. It is affected like information gain (and

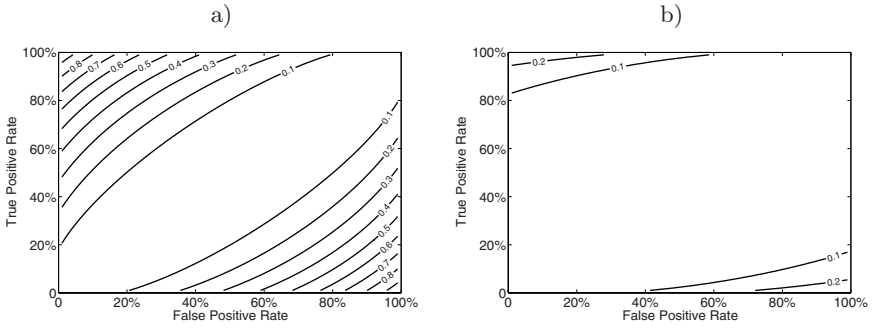


Fig. 3. Gini isometrics for (+:-)=(1:1) in (a) and (+:-)=(1:10) in (b)

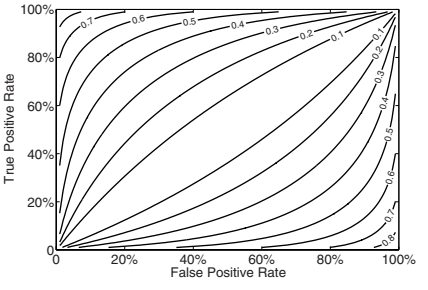


Fig. 4. Hellinger distance isometric for any (+:-)

therefore C4.5) and Gini (and therefore CART) which are highly skew dependent, but not to the same degree. Additionally, its contours do not "twist" – there is some a for which each contour intersects $(0, a)$ and $(1 - a, 0)$ – under skew. Gini is by far the most skew sensitive metric of this group. We only considered two class proportions of $(1 : 1)$ and $(1 : 10)$ to highlight the impact of even a marginal class skew. We point the interested reader to the paper by Flach for a more elaborate analysis of class skew using isometrics on these three metrics [9].

On the other hand, an important observation may be drawn from an isometric of Hellinger distance. First, using Flach's model of relative impurity, we derive the following for Hellinger distance: $\sqrt{(\sqrt{tpr}-\sqrt{fpr})^2+(\sqrt{1-tpr}-\sqrt{1-fpr})^2}$. Figure 4 contains Hellinger distance contours. The Hellinger distance isometrics will not deviate from the contours with varying class skew (c), as there is no factor of c in the relative impurity formulation. This result follows from the previous section and the independence of the Hellinger distance to the parameter λ, which in our case is the respective class priors. The isometric contours for Hellinger distance are unaffected by an increase in the class skew rate.

2.2 Comparing DKM and Hellinger Distance

We posit that DKM and Hellinger distance have similar properties, albeit Hellinger distance has stronger skew insensitivity than DKM. We consider both

DKM and Hellinger distance within the canonical two class, binary split problem. $P(L)$ and $P(R)$ designate the weight of examples falling under the left and right branches respectively. $P(+)$ and $P(-)$ represent the probability of belonging to class $+$ and $-$. In these terms, we may state DKM as follows.

$$d_{DKM} = 2 \quad \overline{P(+)P(-)} - 2P(L) \quad \overline{P(L|+)P(L|-)} - 2P(R) \quad \overline{P(R|+)P(R|-)} \tag{4}$$

Applying these terms to Equation 3, the same terms, Hellinger distance maybe be stated as follows.

$$d_H = \overline{\quad \overline{P(L|+)} - \quad \overline{P(L|-)}^{\,2} + \quad \overline{P(R|+)} - \quad \overline{P(R|-)}^{\,2}} \tag{5}$$

$$d_H = \overline{\quad 2 - 2 \quad \overline{P(L|+)P(L|-)} - 2 \quad \overline{P(R|+)P(R|-)}} \tag{6}$$

Representing the equations in this form demonstrate a clear similarity between Hellinger and DKM. Both are capturing the divergence in conditionals at a split point, albeit with some differences. DKM places the notion of "branch strength" in terms of $P(L)$ and $P(R)$ for each of the corresponding left and right branch conditionals. Moreover, DKM also takes into account the class priors as $2\sqrt{P(+)P(-)}$, which can also be considered as the upper bound for pure splits. On the other hand, Hellinger is upper bounded by $\sqrt{2}$, and does not take into account the notion of class skew in the calculation. It is simply capturing the deviation between $P(x|y)$ at a split node, without factoring in the relative class distributions at the parent node, which DKM does. This also highlights why DKM may be less skew insensitive than Hellinger distance.

Hellinger distance aims for "more pure" leaves as it aims for partitioning the space by capturing the deviations in the class conditionals at the node. However, this can result in smaller coverage, which may be damaging for more balanced class rates, but could prove helpful for highly unbalanced datasets as it tries to form purer leaves that are minority class specific. Nevertheless, it depends on the relative distribution of features with respect to the classes. DKM, on the other hand, may not be as greedy and stop the splitting for the sake of larger coverage.

We demonstrate this property using value surfaces in Figure 5, which display the full range of possible split values for both metrics. Figure 5(a) shows the Hellinger distance throughout all possible class skew ratios, while Figure 5(b),(c), & (d) display DKM values for the $(+:-)$ class ratios of (1:1), (1:10), & (1:100), respectively. As skew increases, the DKM surface flattens and potentially reduces the set of usable values, as it gets dominated by the skew factor in the favor of majority class. We do note that in a vast number of data sets such a scenario may not arise, and Hellinger and DKM may converge to similar performances. Nevertheless, it is important to consider this difference as we may want to grow a completely unpruned tree for unbalanced datasets, and at lower nodes in the tree the class skew may get to the very extreme. This is obviously conditioned on the property of the data, but theoretically it is possible. At this point, Hellinger

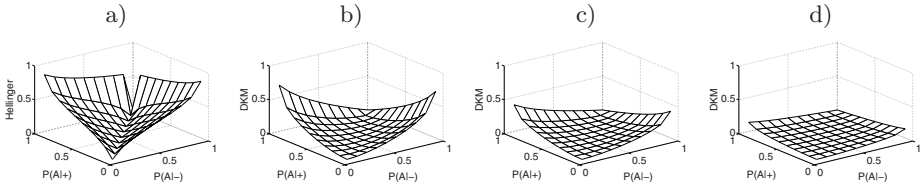


Fig. 5. Full value surfaces for the total range of Hellinger distances and DKM. (a) The Hellinger distance remains unchanged over all possible class skews. The range of DKM values vary with class skew: we note the (+:-) ratios of (b) (1:1), (c) (1:10), & (d) (1:100).

distance may prove more amenable. Thus, we suggest use of Hellinger over DKM given its stronger skew insensitivity, albeit with the caveat that at the general case both Hellinger and DKM will converge to similar performances. But, we want to be prepared for the worst case.

2.3 HDDT: Hellinger Distance Decision Tree

The following algorithm outlines the approach to incorporating Hellinger distance in learning decision trees. We will refer to Hellinger distance and Hellinger distance based decision trees as HDDT for the rest of the paper. In our algorithm, $T_{y=i}$ indicates the subset of training set T that has class i, $T_{x_k=j}$ specifies the subset with value j for feature k, and $T_{x_k=j,y=i}$ identifies the subset with class i and has value j for feature k .

Algorithm 1. *Calc_Hellinger*

Input: Training set T, Feature f

1: **for** each value v of f **do**

2: $Hellinger+ = \left(\sqrt{|T_{x_f=v,y=+}|/|T_{y=+}|} - \sqrt{|T_{x_f=v,y=+}|/|T_{y=+}|}\right)^2$

3: **end for**

4: **return** $\sqrt{Hellinger}$

In the case that a given feature is continuous, a slight variant to Algorithm 1 is used in which *Calc_Hellinger* sorts based on the feature value, finds all meaningful splits, calculates the binary Hellinger distance at each split, and returns the highest distance. This is identical to the methodology used by C4.5. With this practical distance calculator, Algorithm 2 outlines the procedure for inducing HDDT trees.

We do not consider any pruning with HDDT and smoothed the leaf frequencies by the Laplace estimate. This was primarily motivated by the observations of Provost & Domingos [16]. We likewise considered only the unpruned decision trees for C4.5, CART, and DKM, and smoothed the leaf frequencies by the Laplace estimate.

Algorithm 2. *HDDT*

Input: Training set T, Cut-off size C

```
if |T| < C then
    return
end if
for each feature f of T do
    H_f = Calc_Hellinger(T, f)
end for
b = max(H)
for each value v of b do
    HDDT(T_{x_b=v}, C)
end for
```

3 Sampling Methods

Treatment of class imbalance by under- and/or over-sampling, including variants of the same, has resulted in improvement in true positives without significantly increasing false positives [3,19]. However, we believe it is important to understand the interaction of the sampling methods with different decision tree splitting metrics with different skew sensitivities. This study examines combining two samplings methods: random undersampling and SMOTE [5]. While seemingly primitive, randomly removing majority class examples has been shown to improve performance in class imbalance problems. Some training information is lost, but this is counterbalanced by the improvement in minority class accuracy and rank-order. SMOTE is an advanced oversampling method which generates synthetic examples at random intervals between known positive examples.

Elkan discusses the interaction of cost and class imbalance [20], proposing a simple method to calculate optimal sampling levels. However, our evaluation occurs without explicit costs. In this case, Elkan's calculation simply indicates sampling the classes to a balanced proportion. In addition, this approach leaves open much to interpretation: should the negative class be undersampled, the positive class be oversampled, or should a combination be used to reach the balance point? To address this, we search a larger sampling space (which includes several potential balance points) via wrapper to determine optimal class proportions [19]. Testing for each pair of undersampling and SMOTE percentages will result in an intractable search space. The wrapper framework first explores the amounts of undersampling that result in an improvement in performance over the baseline, where baseline is defined as the decision tree classifier learned on the original distribution of data. Subsequently, once the majority class is undersampled to the point where the performance does not deteriorate anymore, the wrapper searches for the appropriate levels of SMOTE. This strategy removes the "excess" negative examples, thereby reducing the size of the training dataset and making learning time more tractable. Then SMOTE adds synthetic positive examples and generalizes performance of the classifier over the positive class. AUROC is the primary metric for considering performance in unbalanced

datasets, so it will be used both as a wrapper objective function and the final performance metric. We point the reader to the paper by Chawla et al. [19] for further details on the wrapper framework.

We perform experiments using each base decision tree classifier in combination with the sampling wrapper. We note that both undersampling and SMOTE contain elements of randomization. Therefore, we first constructed an exhaustive sets of sampled datasets at different amounts of undersampling and different amounts of SMOTE. We let the wrapper search only on these prior constructed undersampled datasets and SMOTE datasets to determine the appropriate levels of sampling for different splitting criteria. For example, each splitting metric considers the removal of exactly the same majority class examples in the first comparison to the baseline. Of course, each splitting criterion may converge to different amounts of undersampling. But this ensures uniformity of results and that the potential performance differences stem from the bias of the decision tree metrics themselves, rather than possible variance due to randomness in the applied sampling methods.

4 Experimental Evaluation

In this section, we provide experimental results to determine performance compares the characteristics of HDDT, C4.5, CART, and DKM and the combination of each with the sampling wrapper. We use a variety of unbalanced, binary-class, mixed feature type real-world and UCI datasets. Such a wide variety should comprehensively outline the strengths and weaknesses of using more skew insensitive metrics such as DKM or Hellinger versus information gain and Gini. We used the 5x2-fold cross-validation (cv) over 10-fold cv as that is more appropriate for unbalanced data sets, as the latter can result in an elevated Type 1 error [21], which is particularly punishing for unbalanced datasets because of the trade-off between false positives and false negatives. Demsar [15] also encourages use of 5x2 cross-validation for statistical comparisons among classifiers across datasets. We statistically evaluate and compare classifiers using the Holm procedure of the Friedman test – a procedure to determine the statistical significance of performance rankings across multiple datasets [15].

4.1 Datasets

Table 1 describes the characteristics of the datasets used in our experiments. We have a number of real-world and UCI datasets. We will briefly describe the real-world datasets used in this paper. E-state contains electrotopological state descriptors for a series of compounds from the National Cancer Institute's Yeast AntiCancer drug screen. Mammography is highly unbalanced and records information on calcification in a mammogram. Oil dataset contains information about oil spills; it is relatively small and very noisy [22]. The Phoneme dataset originates from the ELENA project and is used to distinguish between nasal and oral sounds. Boundary, Calmodoulin, and PhosS are various biological

Table 1. All the datasets used in this paper

No.	Dataset	Examples	Features	MinClass %
1	Boundary	3,505	174	4%
2	Breast-W	569	32	37%
3	Calmodoulin	18,916	131	5%
4	E-State	5,322	12	12%
5	Forest Cover	38,500	10	7.1%
6	FourClass	862	2	36%
7	German.Numer	1,000	24	30%
8	Letter	20,000	16	19%
9	Mammography	11,183	6	2%
10	Oil	937	49	4%
11	Page	5,473	10	10%
12	Pendigits	10,992	16	10%
13	Phoneme	5,404	5	21%
14	PhosS	11,411	479	5%
15	Pima	768	8	35%
16	Satimage	6,435	36	10%
17	Segment	2,310	19	14%
18	Splice	1,000	60	4.8%
19	SVMGuide1	3,089	4	35%

datasets [23]. FourClass, German.Numer, Splice, and SVMGuide1 all are available from *LIBSVM* [24]. The remaining datasets all originate from the UCI repository [25]. Some of these are originally multiple class datasets and were converted into 2-class problems by keeping the smallest class as minority and the rest as majority. The exception is Letter, for which each vowel became a member of the minority class, against all of the consonants as the majority class. Aside from stated modifications, each dataset is used "as is."

4.2 Experimental Results

Baseline Comparisons. We first compare all the baseline decision tree algorithms. In Table 2, we report the average *AUROC* over the 5x2 cv experimental framework. The relative ranking for each classifier is indicated parenthetically. Using the Holm procedure of the Friedman test [15] for comparing the ranking across all the 19 datasets and 4 classifiers, we determine HDDT and DKM are statistically significantly better than C4.5 and CART decision trees at 95% confidence interval. Thus, when applying decision trees to unbalanced data, selecting HDDT or DKM will typically yield a significant edge over C4.5 and CART. In general, DKM and HDDT converge towards similar trees and therefore the final performance. This is reflected by ties on 15 of the 19 datasets, with a marginal improvement in HDDT average ranks over DKM. Thus, these empirical observations agree with the isometric analyses and discussion in the previous Section that as the splitting criterion becomes relatively more skew-insensitive, decision trees tend to perform more strongly on unbalanced data.

Interaction with Sampling. We now consider the effect of sampling on each of the decision tree splitting criterion. We used a wrapper approach, as described in Section 3, to determine the potentially optimal levels of sampling for each

Table 2. Baseline decision tree *AUROC* results with relative ranks in parentheses, with the average rank applied for all ties. HDDT achieves the best over-all ranking. We use the Friedman test to compare the ranks at 95% confidence interval as per the recommendation of Demsar [15] that it is more appropriate to compare classifiers' ranks when using multiple classifiers and multiple datasets. A ✓ in the bottom row indicates that HDDT statistically significantly improved over that classifier.

Dataset	C4.5	DKM	CART	HDDT
Boundary	0.554 ± 0.037 (4)	0.606 ± 0.044 (1)	0.558 ± 0.029 (3)	0.594 ± 0.039 (2)
Breast-w	0.948 ± 0.011 (3)	0.952 ± 0.010 (1.5)	0.937 ± 0.017 (4)	0.952 ± 0.010 (1.5)
Calmodoulin	0.668 ± 0.009 (3)	0.670 ± 0.012 (2)	0.621 ± 0.012 (4)	0.680 ± 0.010 (1)
E-State	0.554 ± 0.035 (3)	0.579 ± 0.014 (2)	0.547 ± 0.020 (4)	0.580 ± 0.014 (1)
Forest Cover	0.978 ± 0.002 (3)	0.982 ± 0.002 (1.5)	0.963 ± 0.004 (4)	0.982 ± 0.002 (1.5)
Fourclass	0.969 ± 0.011 (3)	0.975 ± 0.013 (1.5)	0.946 ± 0.023 (4)	0.975 ± 0.013 (1.5)
German.numer	0.705 ± 0.016 (1)	0.692 ± 0.028 (2.5)	0.629 ± 0.027 (4)	0.692 ± 0.028 (2.5)
Letter	0.990 ± 0.004 (2)	0.990 ± 0.004 (2)	0.962 ± 0.006 (4)	0.990 ± 0.004 (2)
Mammography	0.889 ± 0.008 (3)	0.912 ± 0.013 (1.5)	0.858 ± 0.017 (4)	0.912 ± 0.013 (1.5)
Oil	0.787 ± 0.074 (4)	0.799 ± 0.042 (2.5)	0.815 ± 0.052 (1)	0.799 ± 0.042 (2.5)
Page	0.971 ± 0.004 (3)	0.974 ± 0.005 (1.5)	0.964 ± 0.010 (4)	0.974 ± 0.005 (1.5)
Pendigits	0.985 ± 0.005 (3)	0.992 ± 0.002 (1.5)	0.976 ± 0.007 (4)	0.992 ± 0.002 (1.5)
Phoneme	0.892 ± 0.010 (3)	0.905 ± 0.006 (2)	0.887 ± 0.007 (4)	0.906 ± 0.005 (1)
PhosS	0.638 ± 0.025 (4)	0.677 ± 0.009 (1.5)	0.648 ± 0.017 (3)	0.677 ± 0.009 (1.5)
Pima	0.753 ± 0.013 (3)	0.760 ± 0.019 (1.5)	0.724 ± 0.019 (4)	0.760 ± 0.019 (1.5)
Satimage	0.906 ± 0.009 (3)	0.911 ± 0.008 (1.5)	0.862 ± 0.011 (4)	0.911 ± 0.007 (1.5)
Segment	0.982 ± 0.006 (3)	0.984 ± 0.007 (1.5)	0.977 ± 0.007 (4)	0.984 ± 0.007 (1.5)
Splice	0.954 ± 0.016 (1)	0.950 ± 0.014 (2.5)	0.806 ± 0.035 (4)	0.950 ± 0.014 (2.5)
SVMguide1	0.985 ± 0.005 (3.5)	0.989 ± 0.002 (1.5)	0.985 ± 0.002 (3.5)	0.989 ± 0.002 (1.5)
Avg. Rank	2.92	1.74	3.71	1.63
Friedman $\alpha = .05$	✓		✓	—

of the decision tree algorithms. The wrapper optimized on AUROC. Note that the wrapper uses a separate validation framework to determine the sampling levels. Each decision tree algorithm used 5-fold cross-validation on the training set of the 5x2 cv to determine the optimal sampling levels by optimizing on AUROC. Once these were determined, the entire training set was resampled by that amount and evaluated on the corresponding 5x2 cv testing set. This approach is outlined in the paper by Chawla et al. [19]. The performances that we report are on the testing set of the 5x2 cv.

The results on the 5x2 cv are shown in Table 3. These results show a compelling trend. The benefits of DKM and HDDT over C4.5 are clearly eroded. CART still remains the worst performing classifier, and statistically significantly so. However, there are a couple of exceptions, such as Breast-w and Oil, in which CART and sampling produces the best classifier. We note that these datasets are very small (Breast-w being the smallest and Oil being the fourth smallest) and have the lowest feature-to-example ratio, suggestive of the curse of dimensionality. Moreover, Oil is also very noisy.

One question at this stage is: *how much do the different decision trees benefit from sampling when compared to their respective baseline performances?* Figure 6 depicts the percentage improvement in AUROC across all the datasets after applying sampling for each of the different decision tree splitting criteria. This figure shows a very compelling trend: C4.5 and CART are the biggest gainers from sampling, while DKM and HDDT, being skew insensitive, do not achieve

Table 3. *AUROC* values produced by each decision tree in combination with the sampling wrapper. Relative ranking is noted parenthetically. A ✓ in the bottom row indicates a 95% significant improvement over CART. There was no statistically significant difference among the other three decision tree classifiers – C4.5, HDDT, and DKM.

Dataset	C4.5	DKM	CART	HDDT
Boundary	0.616 ± 0.033 (1)	0.602 ± 0.023 (3)	0.582 ± 0.026 (4)	0.604 ± 0.029 (2)
Breast-w	0.953 ± 0.008 (3)	0.953 ± 0.008 (3)	0.955 ± 0.007 (1)	0.953 ± 0.008 (3)
Calmodoulin	0.676 ± 0.007 (1)	0.660 ± 0.011 (3.5)	0.660 ± 0.007 (3.5)	0.669 ± 0.010 (2)
E-State	0.580 ± 0.016 (1)	0.575 ± 0.013 (2.5)	0.560 ± 0.012 (4)	0.575 ± 0.013 (2.5)
Forest Cover	0.980 ± 0.002 (3)	0.983 ± 0.001 (1.5)	0.974 ± 0.001 (4)	0.983 ± 0.001 (1.5)
Fourclass	0.965 ± 0.012 (3)	0.971 ± 0.009 (1.5)	0.943 ± 0.010 (4)	0.971 ± 0.009 (1.5)
German.numer	0.690 ± 0.015 (1)	0.687 ± 0.015 (3)	0.668 ± 0.016 (4)	0.688 ± 0.014 (2)
Letter	0.989 ± 0.002 (2)	0.989 ± 0.003 (2)	0.977 ± 0.004 (4)	0.989 ± 0.003 (2)
Mammography	0.909 ± 0.011 (1)	0.906 ± 0.013 (2.5)	0.905 ± 0.008 (4)	0.906 ± 0.013 (2.5)
Oil	0.789 ± 0.029 (4)	0.803 ± 0.028 (3)	0.806 ± 0.041 (1)	0.804 ± 0.029 (2)
Page	0.978 ± 0.004 (1)	0.976 ± 0.004 (2.5)	0.970 ± 0.006 (4)	0.976 ± 0.004 (2.5)
Pendigits	0.987 ± 0.003 (3)	0.991 ± 0.002 (1.5)	0.982 ± 0.003 (4)	0.991 ± 0.002 (1.5)
Phoneme	0.894 ± 0.005 (3)	0.902 ± 0.006 (1.5)	0.890 ± 0.006 (4)	0.902 ± 0.006 (1.5)
PhosS	0.670 ± 0.013 (1)	0.666 ± 0.015 (2)	0.665 ± 0.019 (3.5)	0.665 ± 0.015 (3.5)
Pima	0.755 ± 0.013 (3)	0.759 ± 0.014 (1)	0.742 ± 0.014 (4)	0.758 ± 0.013 (2)
Satimage	0.904 ± 0.004 (3)	0.910 ± 0.005 (1.5)	0.887 ± 0.006 (4)	0.910 ± 0.005 (1.5)
Segment	0.982 ± 0.006 (3)	0.984 ± 0.007 (1.5)	0.980 ± 0.006 (4)	0.984 ± 0.007 (1.5)
Splice	0.942 ± 0.013 (1)	0.933 ± 0.010 (2)	0.829 ± 0.015 (4)	0.932 ± 0.009 (3)
SVMguide1	0.987 ± 0.002 (3)	0.988 ± 0.001 (1.5)	0.985 ± 0.001 (4)	0.988 ± 0.001 (1.5)
Avg. Rank	2.16	2.13	3.71	2.08
Friedman $\alpha = .05$	✓	✓	—	✓

significant gains from sampling. In fact, we note that DKM and HDDT often experience a reduction in performance when sampling is applied. 14 out of 19 datasets show a reduction in AUROC for HDDT, and 15 out of the 19 datasets show a reduction in AUROC for DKM. This also points out that the wrapper overfits on the sampling amounts over the validation set, and diminishes generalization capability of HDDT or DKM. Thus, using the natural distribution and letting one of the skew insensitive splitting criteria work the way through the data can be potentially more beneficial over using the computationally expensive step of sampling with DKM and HDDT.

We consider a comparison of all eight potential classifiers: each baseline decision tree and its wrapper-enhanced counterpart. We re-ranked all the 8 classifiers across the 19 datasets. Note that exactly the same training and testing sets were used for all decision tree classifiers, albeit the training sets were modified by sampling when used with the wrapper. These comparative rankings across each dataset are presented in Table 4. There are some interesting observations from this table. The baseline HDDT still achieves the best rank. Wrapper has a positive effect on C4.5 and CART, as pointed out earlier, but a negative effect on both DKM and HDDT. Thus, there is merit to using skew insensitive metrics over sampling. The statistical significance test establishes that HDDT is better than C4.5, CART, and Wrapper + CART.

Finally, we point out that the amounts of sampling determined for each of the decision trees varied. We elucidate that we had first generated the various levels of samplings and then let the wrapper search from that space for each decision tree metric. This ensured that the differences were not due to randomness in

Table 4. Comparative *AUROC* ranks across the entire set of tested classifiers. A ✓ in the bottom row indicates that using the Friedman test HDDT is statistically significantly better (at 95%) than the respective classifier.

	Baseline				Wrapper			
Dataset	C4.5	DKM	CART	HDDT	C4.5	DKM	CART	HDDT
Boundary	8	2	7	5	1	4	6	3
Breast-w	7	5.5	8	5.5	3	3	1	3
Cam	5	3	8	1	2	6.5	6.5	4
Covtype	6	3.5	8	3.5	5	1.5	7	1.5
Estate	7	3	8	1.5	1.5	4.5	6	4.5
Fourclass	5	1.5	7	1.5	6	3.5	8	3.5
German.numer	1	2.5	8	2.5	4	5	6.5	6.5
ism	7	1.5	8	1.5	3	4.5	6	4.5
Letter	2	2	8	2	4	4	7	4
Oil	8	5.5	1	5.5	7	4	2	3
Page	6	4.5	8	4.5	1	2.5	7	2.5
Pendigits	6	1.5	8	1.5	5	3.5	7	3.5
Phoneme	6	2	8	1	5	3.5	7	3.5
PhosS	8	1.5	7	1.5	3	4	5.5	5.5
Pima	6	1.5	8	1.5	5	3	7	4
Satimage	5	1.5	8	1.5	6	3.5	7	3.5
Segment	5.5	2.5	8	2.5	5.5	2.5	7	2.5
Splice	1	2.5	8	2.5	4	5	7	6
SVMguide1	7	1.5	7	1.5	5	3.5	7	3.5
Avg. Rank	5.61	2.58	7.42	2.5	4	3.76	6.13	3.79
Friedman $\alpha = .05$	✓		✓	—			✓	

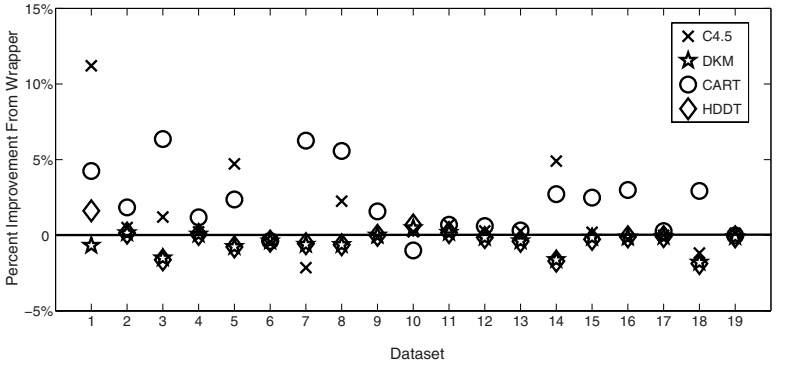


Fig. 6. Percent improvement in *AUROC* from sampling for each decision tree type, with relative rankings. We note that CART generally exhibits the highest improvement yielded from the wrapper. Position on the x-axis corresponds to the dataset number in Table 1.

sampling, and were more intrinsic to the base decision tree splitting metric. Table 5 in the Appendix shows the different sampling levels. HDDT and DKM continued to share similarities in the amounts of sampling level as well. C4.5 and CART generally required higher sampling levels than DKM and HDDT. While there were variations in the amounts of sampling, Friedman test's Holm procedure shows that there is no statistically significant difference in the ranks of levels of sampling for each decision tree.

5 Conclusions

The primary focus of this paper is learning decision trees on unbalanced datasets. We first propose Hellinger distance as a decision tree splitting criterion. We then thoroughly compare four different decision tree splitting criteria with different reactions to the skew in data distribution. We also considered an evaluation of the effect of sampling and how it impacts the different metrics differently. We draw the following conclusions.

Hellinger distance and DKM share similar properties. The isometric in Section 2.1 show Hellinger distance to be skew-invariant while the isometric plot for DKM varies with class skew ratio. However, in Section 2.2, we go on to demonstrate that although there are divergent components of both metrics.This carries over into our experimental results where we note frequent convergence to identical performance.

HDDT and DKM produce superior decision trees under class imbalance. Without using any sampling, both DKM and HDDT statistically significantly outperformed C4.5 and CART.

Sampling generally benefits C4.5 and CART, and hurts DKM and HDDT. We believe this is a compelling observation of this study. We can avoid the use of sampling when using more appropriate decision tree splitting criteria, as those remain superior even after considering sampling. In general, we can recommend the use of HDDT as a decision tree methodology given its skew insensitive properties and the best ranks (no statistical significance over DKM).

As part of future work, we are expanding this study to include balanced and multi-class datasets. We also want to explore the effect of pruning and what pruning methods may be more appropriate for DKM and HDDT. While the focus of this study has been largely on decision trees, we believe rule-based classifiers can also consider Hellinger distance to separate and conquer the instance space.

References

1. Japkowicz, N.: Class Imbalance Problem: Significance & Strategies. In: International Conference on Artificial Intelligence (ICAI), pp. 111–117 (2000)
2. Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: International Conference on Machine Learning (ICML), pp. 179–186 (1997)

3. Batista, G., Prati, R., Monard, M.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. SIGKDD Explorations 6(1), 20–29 (2004)
4. Van Hulse, J., Khoshgoftaar, T., Napolitano, A.: Experimental perspectives on learning from imbalanced data. In: ICML, pp. 935–942 (2007)
5. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
6. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1, 81–106 (1986)
7. Chawla, N.V., Japkowicz, N., Kołcz, A. (eds.): Proceedings of the ICML 2003 Workshop on Learning from Imbalanced Data Sets II (2003)
8. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman and Hall, Boca Raton (1984)
9. Flach, P.A.: The Geometry of ROC Space: Understanding Machine Learning Metrics through ROC Isometrics. In: ICML, pp. 194–201 (2003)
10. Dietterich, T., Kearns, M., Mansour, Y.: Applying the weak learning framework to understand and improve C4.5. In: Proc. 13th International Conference on Machine Learning, pp. 96–104. Morgan Kaufmann, San Francisco (1996)
11. Drummond, C., Holte, R.: Exploiting the cost (in)sensitivity of decision tree splitting criteria. In: ICML, pp. 239–246 (2000)
12. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Proc. 18th International Conf. on Machine Learning, pp. 609–616. Morgan Kaufmann, San Francisco (2001)
13. Kailath, T.: The Divergence and Bhattacharyya Distance Measures in Signal Selection. IEEE Transactions on Communications 15(1), 52–60 (1967)
14. Rao, C.: A Review of Canonical Coordinates and an Alternative to Corresponence Analysis using Hellinger Distance. Questiio (Quaderns d'Estadistica i Investigacio Operativa) 19, 23–63 (1995)
15. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)
16. Provost, F., Domingos, P.: Tree Induction for Probability-Based Ranking. Machine Learning 52(3), 199–215 (September 2003)
17. Chawla, N.V.: C4.5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. In: ICML Workshop on Learning from Imbalanced Data Sets II (2003)
18. Vilalta, R., Oblinger, D.: A Quantification of Distance-Bias Between Evaluation Metrics In Classification. In: ICML, pp. 1087–1094 (2000)
19. Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically countering imbalance and its empirical relationship to cost. Utility-Based Data Mining: A Special issue of the International Journal Data Mining and Knowledge Discovery (2008)
20. Elkan, C.: The Foundations of Cost-Sensitive Learning. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 973–978 (2001)
21. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
22. Kubat, M., Holte, R.C., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30, 195–215 (1998)
23. Radivojac, P., Chawla, N.V., Dunker, A.K., Obradovic, Z.: Classification and knowledge discovery in protein databases. Journal of Biomedical Informatics 37, 224–239 (2004)

24. Chang, C., Lin, C.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
25. Asuncion, A., Newman, D.: UCI Machine Learning Repository (2007)

A Appendix: Wrapper Selected Sampling Levels

Table 5. Optimized $(Undersample, SMOTE)$ levels for each respective splitting metric, along with relative ranking for sampling level among all classifiers. The noted undersample level reflects the percentage of negative class examples removed while the SMOTE level represents the percent of synthetic examples added to the training data relative to the original positive class size.

Dataset	C4.5	DKM	CART	HDDT
Boundary	(76,320) (1,1)	(44,160) (4,4)	(54,210) (3,3)	(59,230) (2,2)
Breast-w	(7,270) (4,4)	(26,350) (2.5,2.5)	(28,280) (1,1)	(26,350) (2.5,2.5)
Calmodoulin	(35,210) (2,1)	(29,90) (4,3)	(57,170) (1,2)	(33,40) (3,4)
E-State	(44,280) (2,1)	(41,120) (3.5,3.5)	(57,250) (1,2)	(41,120) (3.5,3.5)
Forest Cover	(11,440) (3,1.5)	(6,420) (3.5,3.5)	(13,440) (1,1.5)	(6,420) (3.5,3.5)
Fourclass	(14,270) (2.5,3)	(14,320) (2.5,1.5)	(14,100) (2.5,4)	(14,320) (2.5,1.5)
German.numer	(41,250) (1,4)	(39,280) (2.5,3)	(32,330) (4,1)	(39,300) (2.5,2)
Letter	(45,200) (2,4)	(38,210) (3.5,2.5)	(54,410) (1,1)	(38,210) (3.5,2.5)
Mammography	(32,370) (4,2)	(62,360) (1.5,3.5)	(58,420) (3,1)	(62,360) (1.5,3.5)
Oil	(44,330) (1,1)	(39,280) (3,3)	(38,180) (4,4)	(41,310) (2,2)
Page	(9,350) (4,4)	(19,370) (2,2.5)	(19,430) (2,1)	(19,370) (2,2.5)
Pendigits	(38,420) (1,1.5)	(33,320) (2.5,3.5)	(28,420) (4,1.5)	(33,320) (2.5,3.5)
Phoneme	(5,340) (2,4)	(4,370) (2.5,1.5)	(9,350) (1,3)	(4,370) (2.5,1.5)
PhosS	(64,180) (1,1)	(21,0) (3,3.5)	(32,50) (2,2)	(20,0) (4,3.5)
Pima	(15,180) (4,4)	(38,220) (2.5,3)	(40,360) (1,1)	(38,270) (2.5,2)
Satimage	(32,280) (2,2)	(22,240) (3.5,3.5)	(56,370) (1,1)	(22,240) (3.5,3.5)
Segment	(34,260) (1,1)	(23,140) (3.5,3.5)	(30,250) (2,2)	(23,140) (3.5,3.5)
Splice	(10,80) (4,3)	(12,100) (3,1.5)	(25,20) (1,4)	(13,100) (2,1.5)
SVMguide1	(18,300) (1,1)	(12,210) (2.5,2.5)	(5,140) (4,4)	(12,210) (2.5,2.5)
Avg. Undersample Rank	2.24	2.97	2.08	2.76
Avg. SMOTE Rank	2.32	2.89	2.11	2.68

Credal Model Averaging: An Extension of Bayesian Model Averaging to Imprecise Probabilities

Giorgio Corani and Marco Zaffalon

IDSIA
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Manno, Switzerland
{giorgio,zaffalon}@idsia.ch

Abstract. We deal with the arbitrariness in the choice of the prior over the models in *Bayesian model averaging* (BMA), by modelling prior knowledge by a set of priors (i.e., a prior *credal set*). We consider Dash and Cooper's BMA applied to *naive Bayesian networks*, replacing the single prior over the naive models by a credal set; this models a condition close to prior ignorance about the models, which leads to *credal model averaging* (CMA). CMA returns an *indeterminate* classification, i.e., multiple classes, on the instances for which the learning set is not informative enough to smooth the effect of the choice of the prior. We give an algorithm to compute exact credal model averaging for naive networks. Extensive experiments show that indeterminate classifications preserve the reliability of CMA on the instances which are classified in a prior-dependent way by BMA.

Keywords: Credal model averaging, Bayesian model averaging, imprecise probabilities, naive Bayes, classification, naive Bayesian networks.

1 Introduction

In the last ten years, data mining and statistical research has been paying increasing attention to the question of model uncertainty. Loosely speaking, *model uncertainty* refers to a situation where more than one model is consistent with the available data. Many researchers have argued, both theoretically and empirically, that taking such an uncertainty into account leads to improved inference (see [1] for a recent overview). In this context, *Bayesian model averaging* (BMA) [2] has proven to be an effective way to deal with model uncertainty.

BMA is based on a very simple observation: that the posterior probability for an event of interest given the data, say $P(X = x|\mathbf{d})$, can be re-written (in the case of finitely many models) as

$$P(X = x|\mathbf{d}) = \sum_{j=1}^{l} P(X = x|M_j, \mathbf{d})P(M_j|\mathbf{d}), \tag{1}$$

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 257–271, 2008.

thus making explicit its dependency on the possible model M_j; in particular, $P(M_j|\mathbf{d}) = P(\mathbf{d}|M_j)P(M_j)/P(\mathbf{d})$ formalizes how much one should trust each model after having observed the data if the prior beliefs were $P(M_j)$.[1]

Another important issue concerns the arbitrariness inherent in the choice of the prior over the models; in fact, the results produced by BMA can be sensitive to such a choice. Traditionally, a very common choice is to adopt a uniform prior over the models; this, however, can be criticized from different standpoints (see for instance the discussions in the rejoinder of [2]). Alternatively, in [3] a prior is adopted which favors simple models over complex ones. Although all these choices are reasonable in some situation, it is more difficult to justify their use in general. The problem is that the specification of any single prior implies some arbitrariness, which entails the risk of drawing prior-dependent conclusions that may be fragile. In fact, the way the prior over the models should be specified is a serious open problem of BMA.

In this paper we focus in particular on pattern *classification*, where BMA is often related to *feature selection*. In fact, given a set of N feature variables, one can design 2^N different subsets of feature variables; *feature selection* is indeed concerned with selecting the supposedly best subset of the feature variables, which corresponds to a supposedly best classifier. An appealing alternative to the selection of a single classifier is to use BMA to average over all the 2^N classifiers.

This avenue has been taken by Dash and Cooper, who focused in particular on *Bayesian networks* [4]. In case of *naive* networks, in particular, their approach allows one to compute BMA *exactly* and *efficiently*, as their algorithm does not introduce any approximation and has complexity $\mathcal{O}(N)$. Moreover, Dash and Cooper show that exact BMA over the 2^N naive networks can be implemented by a single summary naive network. Yet, they do not discuss the problem of the sensitivity to the prior, nor does so a subsequent approach that implements another form of averaging for naive nets [3].

Our standpoint is that solving the problem of the prior in BMA may require to drop the idea of specifying a unique, precise prior, and to model instead prior knowledge by a set of priors; such a set is referred to as the *credal set*. In Section 2.3 we extend Dash and Cooper's BMA to *imprecise probability* [5], substituting the single prior by a credal set. We call the resulting approach *credal model averaging* (CMA). While traditional non-informative priors model a condition of indifference between the different models, the prior that we define for CMA models a condition close to *prior ignorance* by expressing very weak beliefs a priori about the relative credibility of the 2^N naive nets. Then we use Dash and Cooper's algorithm to efficiently turn each precise prior in the credal set into a posterior. The set of posteriors obtained in this way is referred to as the *posterior credal set*.

Having multiple posteriors instead of one leads to a generalized form of classification that we have called *credal classification* in some previous work [6,7]. In

[1] The more traditional approach to inference that considers only one model $M_{\bar{j}}$ as possible is indeed recovered when $P(M_{\bar{j}}) = 1$.

particular, CMA returns a *determinate* classification, i.e, a single class, only if the probability of such a class is larger than that of any other *for all* the precise posteriors in the posterior credal set. Otherwise, if different classes are found to be the most probable, depending on the specific posterior considered from the posterior credal set, CMA returns an *indeterminate* classification, i.e., multiple classes. We call 'hard to classify' the instances in the test set that give rise to indeterminate classifications, meaning that the learning set is not informative enough about them (in order to smooth the effect of *all* the priors in the credal set in favor of a single class). We expect Dash and Cooper's BMA to behave unreliably on the instances recognized as hard by CMA, as their classification is prior-dependent indeed.

In Section 3 we investigate this point empirically using 31 data sets from the UCI repository. We split the test instances according to whether they are deemed to be hard or not by CMA. Then we evaluate the predictive performance of BMA separately on the two types of instances. What we observe is indeed a striking drop in the predictive accuracy of BMA moving from the instances that are not hard to the others. The drop is observed on every data set we consider, with no exception. Moreover, we show indeterminate classification to be valuable, as they are informative (they return on average only a minority of the classes, not all of them) and reliable (they do contain the actual class with very high frequency). Summing up, extensive experiments show that CMA is a more robust approach than BMA.

Moreover, CMA implements an idea of model averaging that overcomes the arbitrariness in the choice of the prior in a novel way, which could be used more generally than what we do here. In fact, CMA leads very naturally to classification robustness. This is achieved, in particular, by relying on the paradigm of credal classification, which has already proven to be suitable for data mining purposes: in a recent work [7], we have extended naive Bayes to imprecise probabilities, in order to deal robustly with the specification of the prior density over the parameters of the model and with the treatment of missing data, achieving a remarkable reliability improvement compared to naive Bayes. Hence, in our view, allowing classifiers to give weaker answers than the determinate ones we are used to in classification may enhance the overall classification reliability.

2 Credal Model Averaging

In the following section we show how we extend the BMA framework of Dash and Cooper [4] to manage a set of priors over the models. Our setting is in fact characterized by the same assumptions of Dash and Cooper and by a similar notation.

2.1 Setup

We consider a supervised classification problem; there is a vector of N feature variables $\mathbf{F} := (F_1, F_2, \ldots, F_N)$ and a set of N_c classes $\mathcal{C} := \{c_1, c_2, \ldots, c_{N_c}\}$. The i-th instance of the data set $\mathbf{d}$ is the pair $(\mathbf{f}_i, c_i)$, where $\mathbf{f}_i := (f_{1i}, f_{2i}, \ldots, f_{Ni})$

is the instance of the feature variables in the instance under consideration. The data set contains n instances, generated by an *independently and identically distributed* mechanism.

We consider a Bayesian network with $N+1$ nodes, i.e., a single class node and N feature nodes; we assume the network to be *naive*, i.e., a feature node is either linked to the class or it is isolated. We denote by $\mathbf{X}$ the collection of nodes of the network; they are indexed by i so that $X_0 := C$, while, for $i \neq 0$, $X_i := F_i$. Moreover, the class node has no parent. A certain layout of the network, in which certain feature nodes are linked to the network and the remaining ones are isolated, is referred to as a *graph*. Given the N feature variables, we can hence design 2^N different graphs.

All feature variables are assumed to be *categorical*; i.e., each node X_i represents a categorical random variable with r_i possible states. In practice, this requires to discretize the numerical features before inducing the classifier.

We denote by θ_{ijk} the physical probability (or *chance*), about which we are uncertain, of X_i to be in state k when the parent node is in state j. The vector $\boldsymbol{\theta}_{ij}$ (made of r_i elements) contains hence the chances of the states of node i conditional on the j-th state of the parent; finally, $\boldsymbol{\theta}$ collects all the vectors $\boldsymbol{\theta}_{ij}$, i.e., it contains all the parameters of the network.

We take a Dirichlet density $Dir(\alpha_{ij1}, \alpha_{ij2}, \ldots, \alpha_{ijr_i})$ as prior over each vector $\boldsymbol{\theta}_{ij}$, with $\alpha(\cdot) > 0$. We adopt the following setting: for the i-th feature node, we set $\alpha_{ijk} = 1/(N_c \cdot r_i)$. For the class node, we set[2] $\alpha = 1/N_c$.

As usual with Bayesian networks, we assume moreover *parameter independence* and moreover we assume the data set to be *complete*, i.e., without missing data.

2.2 Overview of Dash and Cooper's BMA

In this section we briefly recall Dash and Cooper's approach to BMA. Let us denote by $\mathcal{G}$ the set of the 2^N graphs which can be designed given the N feature variables, and by g a generic graph in $\mathcal{G}$. BMA computes a weighted average of the probabilities produced by all the graphs as follows:

$$P(\mathbf{X} = \mathbf{x}|\mathbf{d}) = \sum_{g \in \mathcal{G}} P(\mathbf{X} = \mathbf{x}|g, \mathbf{d}) P(g|\mathbf{d}) \propto \sum_{g \in \mathcal{G}} P(\mathbf{X} = \mathbf{x}|g, \mathbf{d}) P(\mathbf{d}|g) P(g), \quad (2)$$

where $P(\mathbf{X} = \mathbf{x}|g, \mathbf{d})$ is the posterior probability of the instance to classify assuming that the underlying graph is g (in which some feature variables are linked to the class and some others are isolated), $P(\mathbf{d}|g)$ represents the (so-called *marginal*) *likelihood* of graph g and $P(g)$ represents the *prior probability* of graph g. The last relation in Eq. (2) is due to Bayes' rule.

Let us give the explicit form for the first term in the sum:

$$P(\mathbf{X} = \mathbf{x}|g, \mathbf{d}) = \prod_{i=0}^{N} \hat{\theta}_{iJK} := \prod_{i=0}^{N} \frac{\alpha_{iJK} + n_{iJK}}{\alpha_{iJ} + n_{iJ}}. \quad (3)$$

[2] To be more precise, the parameter referring to the class node should be denoted as α_{00k}.

Here the coefficients n_{ijk} are counts collected from the data set that report how many times feature variable i is in state k when its parent is in state j; coefficients α_{ijk} refer instead to the Dirichlet densities introduced in Section 2.1. Moreover, $n_{ij} := \sum_k n_{ijk}$ and $\alpha_{ij} := \sum_k \alpha_{ijk}$. The uppercase letters J and K denote the specific states of nodes and parents which have been read off from vector $\mathbf{x}$.

In practice, the coefficients n_{ijk} are computed differently depending on whether they refer to the class, to a feature node linked to the class or to an isolated feature node. In particular:

- for the class node, n_{0jk} has the meaning of class frequency, i.e., it indicates how many times class k occurs in the training set; at the denominator, n_{0j} corresponds to the data set size n. Note that the value of $\hat{\theta}_{0jk}$ is the same for all graphs;
- for feature nodes linked to the class, n_{ijk} represents a conditional frequency, i.e., it indicates how many times in the training set feature variable i assumes value k while the class has value j; at the denominator, n_{ij} represents the total occurrences of class j in the training set. Given a feature X_i ($i \neq 0$), $\hat{\theta}_{ijk}$ has the same value for all graphs in which X_i is linked to the class node; let us denote this quantity by $\hat{\theta}^C_{ijk}$;
- for isolated feature nodes, n_{ijk} represents an unconditional frequency, i.e., it indicates how many times feature variable i assumes value k in the training set; at the denominator, n_{ij} corresponds to the data set size n. Given a feature X_i ($i \neq 0$), $\hat{\theta}_{ijk}$ has the same value for all graphs in which node X_i is isolated; let us denote this quantity by $\hat{\theta}^{\emptyset}_{ijk}$.

The coefficients $\alpha(\cdot)$ can be interpreted in the same way of coefficients $n(\cdot)$, provided that they are regarded as referring to the so-called *hypothetical sample* rather than to the actual data set.

Let us now consider the marginal likelihood. We have:

$$P(\mathbf{d}|g) = \prod_{i=0}^{N} M_i := \prod_{i=0}^{N} \left(\prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \right), \tag{4}$$

where the coefficients $\alpha(\cdot)$ and $n(\cdot)$ have the meaning already discussed. Hence, M_0 is a fixed value for all the graphs, while M_i ($i \neq 0$) is a fixed value M^C_i for all the graphs in which X_i is linked to the class, and another fixed value $M^{\emptyset}_i$ for all the graphs in which X_i is isolated. Let i^C and as $i^{\emptyset}$ denote the set of indexes to feature variables which in graph g are respectively linked to the class node and isolated. We can eventually express Eqs. (3) and (4) in a more compact way as

$$P(\mathbf{X} = \mathbf{x}|g, \mathbf{d}) = \hat{\theta}_0 \prod_{i \in i^C} \hat{\theta}^C_{iJK} \prod_{i \in i^{\emptyset}} \hat{\theta}^{\emptyset}_{iJK}, \tag{5}$$

$$P(\mathbf{d}|g) = M_0 \prod_{i \in i^C} M^C_i \prod_{i \in i^{\emptyset}} M^{\emptyset}_i. \tag{6}$$

$$\tag{7}$$

Concerning the prior over the graphs, corresponding to the term $p(g)$ in Eq. (2), Dash and Cooper require it to be a *modular* prior, which means it should also factorize into a product of $N+1$ terms, each one corresponding to a node. Then they do not detail the prior any further, much probably because they use a flat prior that cancels out of the calculations. Since this will not be our case, we give here a few more details about the prior. Call p_i the probability that node i is connected to a parent.We design a modular prior by simply requiring that

$$P(g) = \prod_{i \in i^C} p_i \prod_{i \in i^\emptyset} (1 - p_i), \tag{8}$$

and in addition that $p_0 = 0$, because we know that the class variable has always no parents. Note that to recover the flat prior over the graphs, it would be sufficient to set $p_i := 0.5$ for all $i = 1, \ldots, N$.

We are finally in the condition to write an explicit formula for $P(\mathbf{X} = \mathbf{x}|\mathbf{d})$. Let us introduce the following quantities (which are all positive):

$$\begin{aligned} \rho_{0K} &:= \theta_{0JK} M_0, \\ \rho^C_{iJK} &:= \theta^C_{iJK} M^C_i, \\ \rho^\emptyset_{iK} &:= \theta^\emptyset_{iJK} M^\emptyset_i, \end{aligned} \tag{9}$$

where we have dropped index J in the definition of ρ_{0K} and $\rho^\emptyset_{iK}$; in fact, these quantities refer to the class node and to the isolated feature nodes, which have no parents. It turns out then that

$$P(\mathbf{X} = \mathbf{x}|\mathbf{d}) \propto \rho_{0K} \cdot \prod_{i=1}^{N} \left[(1 - p_i)\rho^\emptyset_{iK} + p_i \rho^C_{iJK} \right]. \tag{10}$$

This way of expressing $P(\mathbf{X} = \mathbf{x}|\mathbf{d})$ is an achievement from Dash and Cooper that is particularly important for computations: in fact, it means that once the $\rho(\cdot)$ coefficients have been computed, Eq. (10) is computed in $\mathcal{O}(N)$ time, without the need for implementing the 2^N models and without introducing any approximation. Dash and Cooper also show that computing BMA according to Eq. (2) is equivalent to implementing a single summary network characterized by a new vector $\hat{\boldsymbol{\theta}}^*$; assuming to adopt a uniform prior over the graphs, it holds that for $i = 0$, $\hat{\theta}^*_{0JK} \propto \rho_{0K}$ and, for $i \neq 0$, $\hat{\theta}^*_{iJK} \propto (\rho^C_{iJK} + \rho^\emptyset_{iK})$.

2.3 Extension of BMA to Imprecise Probabilities

We extend BMA to imprecise probabilities by considering a set $\mathcal{P}$ of priors over the graphs, instead of a single prior; $\mathcal{P}$ is referred to as *prior credal set*. Before detailing the construction of the prior credal set, let us consider the motivations behind such a choice and some of its consequences.

A major motivation behind using a credal set rather than a single prior is related to modeling prior ignorance. The point is that by a single prior it is possible to model indifference; in order to model ignorance, one should use a

IDENTIFICATION OF NON-DOMINATED CLASSES

1. set NonDominatedClasses := $\mathcal{C}$;
2. for class $c' \in \mathcal{C}$
   - for class $c'' \in \mathcal{C}$, $c'' \neq c'$
     - if c'' is dominated by c' (to be assessed via the below procedure), drop c'' from NonDominatedClasses;
     - exit;
   - exit
3. return NonDominatedClasses.

Fig. 1. Identification of non-dominated classes via pairwise comparisons

credal set.[3] Therefore credal sets allow us to express more satisfactorily the fact that initially we do not know about the relative credibility of the models; this naturally makes the resulting classifier more robust than BMA. Indeed, especially when the learning set is small, the class returned by BMA may well vary depending on the specification of the prior over the graphs; in this case, the classification is defined as *prior-dependent* and its reliability is questionable. On the other hand, since CMA considers a set of priors as possible, it is aware by construction that some classifications may change with the choice of the prior in the credal set, and this enables it to keep reliability. The way this is done in practice is related to the definition of the optimality criterion for the classes in the imprecise setting.

Let us recall that a Bayesian classifier returns as *optimal prediction* the class with the highest probability (in the case of 0-1 loss function), identified on the basis of a uniquely computed posterior, derived from a unique prior. In the imprecise probability setting, one specifies a set of priors that is turned into a set of posteriors by element-wise application of Bayes' rule. According to Section 3.9.2 of [5], the optimality criterion in this case is to return all the *non-dominated* classes.

The definition of dominance is as follows: class c_1 dominates c_2 if for all the computed posteriors, the probability of c_1 is greater than that of c_2; hence, c_2 is non-dominated if no class dominates c_2.

A key point is that there can be several non-dominated classes; in this case, the classifier returns an indeterminate (or set-valued) classification. Classifiers that issue set-valued classifications are called *credal classifiers* in [6]. Summing up, a credal classifier will become indeterminate on the instances whose classification would be prior-dependent when a single prior is used; on these instances, it will return all the non-dominated classes as a way to maintain reliability. It is important to realize that non-dominated classes are *incomparable*; this means

[3] Yet, complete prior ignorance is not compatible with learning, see Section 7.3.7 of [5]. This issue is re-considered later in this section when we define the credal set.

that there is no information in the model that allows us to rank them. In other words, credal classifiers are models that allow us to drop the dominated classes, as sub-optimal, and to express our indecision about the optimal class by yielding the remaining set of non-dominated classes.

Let us focus now in particular on the test of dominance; let $\mathbf{x}_1 := (\mathbf{f}, c_1)$ and $\mathbf{x}_2 := (\mathbf{f}, c_2)$. We say that class c_2 is dominated by c_1 if and only if

$$P(\mathbf{x}_1|\mathbf{d}) > P(\mathbf{x}_2|\mathbf{d}) \ \forall P \in \mathcal{P},$$

or, equivalently,[4] if and only if

$$\frac{P(\mathbf{x}_1|\mathbf{d})}{P(\mathbf{x}_2|\mathbf{d})} = \frac{P(\mathbf{x}_1, \mathbf{d})}{P(\mathbf{x}_2, \mathbf{d})} > 1 \ \forall P \in \mathcal{P},$$

which, taking Eq. (2) into consideration, can be finally re-written as

$$\inf_{P \in \mathcal{P}} \frac{\sum_{g \in \mathcal{G}} P(\mathbf{x}_1|g, \mathbf{d}) P(\mathbf{d}|g) P(g)}{\sum_{g \in \mathcal{G}} P(\mathbf{x}_2|g, \mathbf{d}) P(\mathbf{d}|g) P(g)} > 1. \tag{11}$$

A procedure to determine all the non-dominated classes via pairwise comparisons is shown in Figure 1.

Prior Credal Set. We are finally ready to define the prior credal set. Let us focus on p_i, that is, the probability that feature variable i is connected to the class. Remember that we want to model a condition of prior ignorance about the actual graph, among the 2^N possible ones, giving rise to the data. Since we are ignorant a priori, this means that for each feature variable, we ignore whether it is linked or not to the class. In turn, this means that our probability p_i for the related arc should lie in $[0, 1]$. We can therefore construct the credal set $\mathcal{P}$ by considering the set of all the mass functions defined as in (8) that are obtained when each p_i, $i = 1, \ldots, N$, is subject to the constraint $0 < p_i < 1$ (and, as before, $p_0 = 0$). However, it can be checked that this choice does not allow us to learn from data about the relative credibility of the models. Broadly speaking, this is a relatively well-known phenomenon (e.g., something similar was noticed in [8, Section 3] in the case of feature selection); the intuition here is that the modeled condition is of such deep ignorance a priori that no amount of data would be able to make us get out of such a state. For this reason, we need to consider a slightly smaller credal set as defined by the following constraints:

$$\begin{aligned} p_i &= 0 \text{ if } i = 0, \\ \epsilon \leq p_i &\leq 1 - \epsilon \text{ if } i \neq 0, \end{aligned} \tag{12}$$

where ϵ is a small number in $(0, 0.5)$, which we will set to 10^{-5} in our experiments.[5] By this simple consideration, we model a condition that is still close to ignorance but that at the same time enables us to learn.

[4] If the denominator is positive, which is always the case in this paper.

[5] We have not tried to optimize this parameter, we have chosen it very small once for all just to create a credal set close to that modeling ignorance.

Two final remarks are worth making. One is that when CMA is determinate, it returns the same class as BMA. This is the consequence of two facts: that (a) CMA returns a determinate output when a certain class dominates all the remaining ones, under all the priors of the credal set; and that (b), the credal set includes the flat prior adopted by BMA (remember that it is actually characterized by $p_i = 0.5$ for all the feature variables). Therefore, BMA and CMA achieve the same accuracy on the subset of instances determinately classified by CMA, and whose classification is prior-independent.

The second remark is that CMA will converge to BMA with increasing sizes of the learning set. This follows because all the precise priors in the prior credal set will converge towards a single posterior with more and more data. Therefore in the limit, CMA will yield a traditional classifier that always issues determinate classifications.

2.4 CMA Computation

Recalling Eqs. (10) and (11), and letting $\mathbf{p} := (p_1, \ldots, p_N)$, the CMA test of dominance for classes c_1 and c_2 can be written as follows:

$$\min_{P \in \mathcal{P}} \frac{\sum_{g \in \mathcal{G}} P(\mathbf{x_1}|g, \mathbf{d}) P(\mathbf{d}|g) P(g)}{\sum_{g \in \mathcal{G}} P(\mathbf{x_2}|g, \mathbf{d}) P(\mathbf{d}|g) P(g)} = \min_{\mathbf{p}} \frac{\rho_{01}}{\rho_{02}} \prod_{i=1}^{N} \frac{\left[(1-p_i)\,\rho_{iK}^{\emptyset} + p_i \rho_{i1K}^{C}\right]}{\left[(1-p_i)\,\rho_{iK}^{\emptyset} + p_i \rho_{i2K}^{C}\right]}. \quad (13)$$

Note that the function in (13) can be globally minimized by minimizing independently each term of the product, i.e., by minimizing independently over each p_i. The minimization problem to be solved for a single p_i is

$$\min_{p_i} \frac{\left[(1-p_i)\,\rho_{iK}^{\emptyset} + p_i \rho_{i1K}^{C}\right]}{\left[(1-p_i)\,\rho_{iK}^{\emptyset} + p_i \rho_{i2K}^{C}\right]} = \min_{p_i} \frac{p_i m_1 + a}{p_i m_2 + a} = \min_{p_i} f(p_i), \quad (14)$$

where

$$a := \rho_{iK}^{\emptyset},\ m_1 := \rho_{i1K}^{C} - \rho_{iK}^{\emptyset},\ m_2 := \rho_{i2K}^{C} - \rho_{iK}^{\emptyset} \quad (15)$$

and subject to the constraint $\epsilon \leq p_i \leq 1 - \epsilon$. This is an easy problem because the derivative of the function in (14), which is

$$\frac{\partial f}{\partial p_i} = \frac{a(m_1 - m_2)}{(p_i m_2 + a)^2},$$

is a ratio with positive denominator, as it follows from Eqs. (9) and (12). It follows that:

- if $m_1 > m_2$, the derivative is positive over the interval $(\epsilon, 1-\epsilon)$; the function is minimized by setting $p_i := \epsilon$;
- if $m_1 < m_2$, the derivative is negative over the interval $(\epsilon, 1-\epsilon)$; the function is minimized by letting $p_i := 1 - \epsilon$;
- if $m_1 = m_2$, the function is constant.

These three rules define the graph over which CMA will concentrate the prior probability when testing whether c_1 dominates c_2; hence, if $P(c_1|\mathbf{d})/P(c_2|\mathbf{d}) > 1$ under this prior, the same will happen under all the remaining priors of the credal set, and hence class c_2 can be safely dropped.

As a side remark, let us note that $(m_1 - m_2) = (\theta^*_{i1K} - \theta^*_{i2K})$. Hence, the architecture over which CMA concentrates the mass when testing whether c_1 dominates c_2 can be defined also in an alternate way, i.e., feature X_i is linked to the class node if and only if its addition decreases the ratio $P(c_1|\mathbf{d})/P(c_2|\mathbf{d})$ computed by the BMA summary network. It also interesting to note that CMA has the freedom to change architecture depending on the specific pair of classes that are compared.

Software Availability. The software implementing CMA has been realized in Java; we plan to release the package soon under the GNU GPL license. Sources, binaries and documentation (both user manual and sources documentation in javadoc format) will be available from the website `http://www.idsia.ch/~giorgio/jncc2.html`. Meanwhile, it is possible to obtain the software by contacting the authors by e-mail.

3 Experiments

We present the results obtained on 31 data sets from the UCI repository. The data sets cover a wide spectrum of conditions in terms of number of instances (min: 57, labor; max: 4601, spambase), number of feature variables (min: 3, haberman; max: 69, audiology) and number of classes (up to 24, audiology). On each data set, the classifiers have been evaluated via 10 runs of 10 folds cross-validation. Numerical features have been discretized via MDL-based discretization [9]; in each training/test experiment, the discretization intervals have been computed on the training set and then applied unchanged on the test set.

Some questions of interest are then: is CMA truly able to isolate instances which are hard to classify for BMA? How does BMA behave on the instances which are classified determinately and indeterminately by CMA? Are indeterminate classifications informative and reliable?

We start our analysis by measuring the accuracy of BMA on the instances classified determinately and indeterminately by CMA; these two indicators are denoted respectively by BMA(CMA D) and BMA(CMA I). If CMA is able to recognize instances that are hard to classify, we should observe a significant drop of BMA accuracy between the former and the latter set of instances.

These results are shown in Table 1, which also reports, to complement the information, the average accuracy of BMA. On average, there is a drop of 32 points between BMA(CMA D) and BMA(CMA I); moreover, on *every* data set we clearly observe that BMA(CMA D) is strictly larger that BMA(CMA I); hence, we can safely state that CMA isolates instances that are hard to classify and where, as a consequence, BMA becomes less reliable.

On the other hand, CMA reacts to the hard instances by returning indeterminate classifications. Another point of interest is hence to evaluate the informative

Table 1. Comparison of BMA and CMA on 31 UCI data sets. Note that the indicator BMA(CMA I) is not available for those data on which CMA achieves 100% determinacy.

Dataset	Accuracies			
	BMA (Avg.)	BMA (Cma D)	BMA (Cma I)	CMA Determ.
anneal	97.9%	98.7%	71.2%	97.3%
audiology	73.5%	99.5%	63.7%	27.0%
autos	66.7%	81.3%	31.7%	70.8%
balance-scale	72.8%	72.8%	n.a	100.0%
breast-cancer	74.9%	83.4%	68.0%	44.9%
c-14-heart-disease	83.0%	85.8%	60.9%	88.3%
cmc	50.3%	58.5%	39.4%	57.7%
credit-rating	85.5%	90.1%	51.8%	88.0%
german_credit	73.7%	87.1%	61.7%	46.9%
glass	71.1%	71.4%	60.3%	98.8%
haberman	71.8%	77.2%	50.8%	81.4%
heart-statlog	83.1%	85.1%	53.0%	93.6%
hepatitis	84.3%	95.6%	72.3%	51.4%
horse-colic	81.2%	86.2%	58.2%	82.1%
h-14-heart-disease	84.3%	85.6%	64.9%	94.2%
ionosphere	89.9%	89.9%	n.a	100.0%
iris	93.7%	93.7%	n.a	100.0%
kr-vs-kp	88.0%	93.7%	60.5%	82.9%
labor	86.9%	98.6%	82.3%	32.2%
liver-disorders	57.4%	60.0%	48.9%	80.1%
lymphography	81.1%	96.1%	73.5%	33.3%
pima_diabetes	75.7%	77.3%	37.2%	95.8%
segment	92.5%	92.5%	60.0%	99.9%
soybean	91.9%	95.8%	26.3%	94.2%
spambase	89.8%	89.8%	n.a.	100.0%
vote	90.2%	90.2%	75.0%	99.8%
wisc-breast-cancer	97.1%	97.1%	n.a	100.0%
yeast	57.2%	57.2%	29.5%	99.9%
zoo	96.4%	98.1%	65.1%	94.2%
primary-tumor	36.8%	83.4%	25.9%	19.0%
contact-lenses	87.3%	100.0%	85.7%	22.2%
average	79.6%	86.2%	54.7%	75.9%

content of the indeterminate classification; this can be properly assessed only on data sets with at least three classes, since, on data sets with two classes, indeterminate classifications contain all the classes.[6] Excluding hence data sets with two classes from the analysis, we have measured on average that set-valued classifications return 35% of the classes of the data set, dropping hence 65% of

[6] Nevertheless, we deem set-valued classifications to be valuable also in the case of data sets with two classes only, as they highlight that a certain classification is doubtful, thus preventing an over-confident use of the output of the model.

them; therefore, they convey significant information. Moreover, set-valued classifications are very reliable; in fact, they contain the actual class in 90% of cases. Summing up, CMA is able to detect hard instances where the accuracy of BMA drops indeed; on these instances, indeterminate classifications preserve the reliability of CMA, by conveying reliable information, without however drawing too strong conclusions.

A further important indicator of performance is the determinacy of CMA, i.e., the percentage of instances over which CMA returns a single class; on average, CMA achieves 77% determinacy, i.e., it yields set-valued classification on 23% of instances. The data sets which lead to the largest indeterminacy are characterized by a small number of instances and a relatively high number of feature variables/classes; see for instance: primary-tumor (339 instances, 17 feature variables, 22 classes, determinacy: 19%), contact-lenses (24 instances, 4 feature variables, 3 classes, determinacy: 22%), audiology (226 instances, 69 feature variables, 24 classes, determinacy: 27%). However, the caution of CMA on these data sets is justified, as the drop between BMA(CMA D) and BMA(CMA I) is respectively of 57.5, 14.3 and 35.9 points. On the other hand, the determinacy of CMA quickly increases on data sets which contain more instances or less features variables.

3.1 BMA Probabilities vs. CMA Set-Valued Classifications

We have shown that, thanks to imprecise probabilities, CMA delivers set-valued classifications on hard-to-classify instances, over which the accuracy of BMA clearly drops. In the following, we analyze the association between the posterior probabilities computed by BMA and the set-valued classifications returned by CMA. To this purpose, we focus on the example of the German credit data set, which is made of 2 classes, 20 feature variables and 1000 instances. As the data set has two classes, it is easy to spot instances that are deemed doubtful according to BMA (they are classified with probability lower than, say, 55%) and to CMA (indeterminate classifications).

To perform our analysis, we consider four pieces of information for each instance: (i) the actual class, (ii) the class returned by BMA, (iii) its probability, and (iv) whether the instance has been classified determinately or indeterminately by CMA. The instances are then partitioned into subsets, according to the probability estimated by BMA for the returned class, i.e., instances for which BMA estimates a probability in the range 50–55%, 55–60%, and so on (i.e., we use a step of 5% in probability to define the subsets). On each subset of instances, we measure: (a) the determinacy of CMA; (b) BMA(CMA D) and (c) BMA(CMA I).

The results are reported in Figure 2. There is a positive association between higher posterior probabilities computed by BMA and higher determinacy; indeed the choice of the prior over the graphs is less likely to change the classification outcome when the probability computed by BMA for the most probable class increases. The output of CMA is indeterminate for all the instances

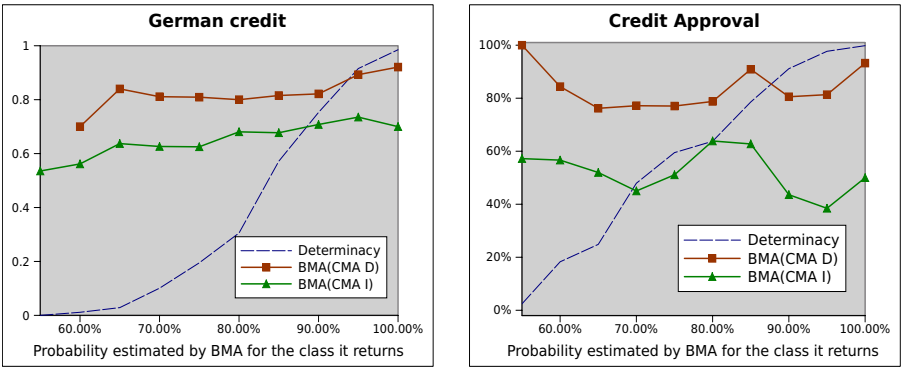


Fig. 2. Relationship between the posterior probabilities computed by BMA and the output of CMA

as long as the probability estimated by BMA for the returned class is lower than 55%. Hence, on the instances classified with probability less that 55% by BMA, BMA and CMA convey a similar message, i.e., that of a doubtful classification: BMA by returning a low probability for the class, CMA by becoming indeterminate.

Moving on to greater probabilities, the determinacy of CMA rises progressively; however, CMA keeps returning a mix of determinate and indeterminate classifications, even on instances classified very confidently by BMA (for instance, with probability higher than 80%). The point is that the behavior of CMA is justified, since at any level of posterior probability estimated by BMA there is a clear drop of accuracy between BMA(CMA D) and BMA(CMA I).

Similar patterns have been observed on most of the data sets with two classes included in our list; we report for instance in Figure 2 also the results obtained on the credit approval data set.[7]

From Figure 2, one can also appreciate that the behavior of CMA cannot be mimicked by a BMA with threshold, i.e., a BMA which returns two classes unless the probability for the most probable class exceeds a fixed threshold t. In fact, a BMA with threshold would assume all instances classified with probability less than t to be hard; instead CMA identifies in a sensible way a mix of easy and hard instances between both the instances classified with probability greater or smaller than the threshold. Moreover, CMA is able to detect hard instances also among those classified with very high probability from BMA, something which would not be possible to accomplish with a BMA with threshold.

[7] To prevent ambiguities, we point out that German-credit and credit approval are two distinct data sets; the former has been donated by Prof. Hofmann, and contains 20 feature variables; the latter has been donated by Prof. Quinlan, and contains 15 feature variables.

4 Conclusions

In this paper we have proposed an extension of Bayesian model averaging to imprecise probabilities that we have called credal model averaging. By CMA, we have tried to tackle one of the more serious challenges of BMA, which is related to the choice of the prior over the models: both the difficulty in defining such a prior, and the unavoidable arbitrariness that any choice entails. In our approach, prior beliefs model a condition close to ignorance about the models, thus trying to implement an objective-minded approach to model averaging. This naturally leads to a new form of averaging whose conclusions are robust to the definition of prior beliefs.

We have applied CMA in particular to problems of classification based on naive Bayes nets. Our empirical experiments over many data sets have confirmed that CMA leads to reliable inference. It leads, in particular, to create classifiers that can suspend the judgment when the conditions do not justify strong conclusions, and that we have called credal classifiers. What we have seen clearly from the experiments is that suspending judgment has been well motivated: the attempt of BMA-based classifiers to produce a determinate classification when CMA leads to suspend judgment, yields fragile classifications that heavily deteriorate the predictive performance of the former.

In summary, CMA approaches in an original way the controversial problem of setting the prior over the models for BMA; moreover, it performs well and reliably in classification problems.

CMA has been derived assuming to have a complete data set; it would be however interesting to extend CMA to deal also with missing data. If one assumes the missing data to be generated by a missing-at-random (MAR) process, realizing such an extension would be straightforward. However, the MAR assumption is not always met; therefore, a more sophisticated treatment of missing data should be developed, able to deal with missing data differently, depending on whether they are generated by a MAR or non-MAR missingness process. We have followed a similar avenue in [7]; however, incorporating such a treatment of missing data into CMA could be technically quite involved and therefore it needs careful investigation.

Acknowledgments. Work for this paper has been partially supported by the Swiss NSF grants 200021-113820/1 and 200020-116674/1, and by the Hasler Foundation (Hasler Stiftung) grant 2233.

References

1. Hoeting, J., Madigan, D., Raftery, A., Volinsky, C.: Bayesian Model Averaging: a Tutorial. Statistical Science 14(4), 382–417 (1999)
2. Clyde, M., George, E.I.: Model Uncertainty. Statistical Science 19, 81–94 (2004)
3. Boullé, M.: Compression-Based Averaging of Selective Naive Bayes Classifiers. Journal of Machine Learning Research 8, 1659–1685 (2007)

4. Dash, D., Cooper, G.: Exact Model Averaging with Naive Bayesian Classifiers. In: Proceedings of the Nineteenth International Conference on Machine Learning, pp. 91–98 (2002)
5. Walley, P.: Statistical Reasoning with Imprecise Probabilities. Chapman and Hall, New York (1991)
6. Zaffalon, M.: The Naive Credal Classifier. Journal of Statistical Planning and Inference 105(1), 5–21 (2002)
7. Corani, G., Zaffalon, M.: Learning Reliable Classifiers from Small or Incomplete Data Sets: the Naive Credal Classifier 2. Journal of Machine Learning Research 9, 581–621 (2008)
8. Abellán, J., Moral, S.: A New Score for Independence Based on the Imprecise Dirichlet model. In: Cozman, F.G., Nau, R., Seidenfeld, T. (eds.) ISIPTA 2005: Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications, Manno, Switzerland, pp. 1–10. SIPTA (2005)
9. Fayyad, U.M., Irani, K.B.: Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)

A Fast Method for Training Linear SVM in the Primal

Trinh-Minh-Tri Do and Thierry Artières

LIP6 - Université Pierre et Marie Curie
104 avenue du président Kennedy, 75016 Paris France
Trinh-Minh-Tri.Do@lip6.fr, Thierry.Artieres@lip6.fr

Abstract. We propose a new algorithm for training a linear Support Vector Machine in the primal. The algorithm mixes ideas from non smooth optimization, subgradient methods, and cutting planes methods. This yields a fast algorithm that compares well to state of the art algorithms. It is proved to require $O(1/\lambda\epsilon)$ iterations to converge to a solution with accuracy ϵ. Additionally we provide an exact shrinking method in the primal that allows reducing the complexity of an iteration to much less than $O(N)$ where N is the number of training samples.

1 Introduction

Support Vector Machines (SVMs) are a very popular method for supervised learning tasks such as classification and regression. The standard way for solving the SVM learning problem is to introduce Lagrange multipliers, one for each constraint, and to optimize the equivalent dual problem which is an instance of quadratic programming. Since direct optimization of this quadratic problem becomes uneasy when the training set size (N) increases, some solutions have focused on efficient optimization of the dual through decomposition of the learning problem [1]. Decomposition methods like Sequential Minimal Optimization (SMO) [2], SVM-light [3], LIBSVM [4] can handle larger problems. However, their super-linear scaling behavior with N makes them intractable for very large datasets. To overcome this limitation some techniques have been investigated that mainly rely on an approximation of the dual [5,6]. However, as suggested in [7], optimizing an approximation of the dual might not be a good idea when one actually wants to optimize the primal.

Consequently, a number of recent works tackled the learning of SVM through direct optimization of the primal. By introducing the hinge loss function ($hinge(z) = max(0, z)$), the constrained optimization problem is transformed into an unconstrained convex one. The main difficulty for optimizing this objective function lies in the non-differentiability of the hinge loss function at 0. Two main solutions have been proposed. The first one is to use a differentiable approximation of the loss (e.g. by smoothing) in which case standard optimization methods can be applied (e.g. [8,7]). [8] proposes a efficient method to solve the primal linear SVM which can be very fast if the number of features d is

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 272–287, 2008.

small, scaling roughly as $O(Nd^2)$. The second solution is to rely on sub-gradient methods for direct optimization of the objective function [9]. The advantage of this latter approach is its simplicity, but its rate of convergence is usually much dependent on the setting of the stepsize. An exception is the Pegasos algorithm [10] which is the only one of this family that does not require the setting of this hyperparameter. Yet, note that most of those algorithms in primal [9,10] have been proposed for linear SVM.

Lastly, recent works proposed to use the cutting planes technique to build an approximated problem (i.e. a lower bound to be maximized) which is refined every iteration [11,12]. Generally speaking, the approximated problem is represented as an optimization problem with linear constraints. At each iteration, a new constraint is added to the problem and the approximated problem is solved in its dual form (it is somehow a semi dual method). Besides, one advantage of this method is that one can gain information of the approximation quality every iteration, which provides a meaningful stopping criterion.

In this paper, we propose a new algorithm for solving the SVM learning unconstrained optimization problem in the primal form (with hinge loss). Our work is a mix of subgradient methods, non smooth optimization, and cutting planes method. Our algorithm relies on the following ideas:

Simple approximation. We use the cutting planes method to iteratively build a lower bound of the primal objective function. We use a simple lower bound which can be solved very quickly. Despite its simplicity, we prove that the number of iterations required to reach the solution with accuracy ϵ is $O(1/\lambda\epsilon)$.

Efficient linesearch in the primal. Unlike many previous works, we deal with the particular shape of the objective function to derive an efficient optimal line search in a given direction from the current solution.

Shrinking in Primal. Our iterative algorithm approaches the solution iteratively. If the current solution doesn't move too far from a given iteration to the next, many constraints can be ignored in the optimization step. We propose a shrinking method that reduces the number of terms actually used in the simplified objective function to much less than N.

Section 2 presents the context and discusses related works, Section 3 recall non smooth optimization results applied to the primal objective function. Section 4 details our algorithm, the building of the lower bound, the optimal linesearch, convergence analysis, and the shrinking method. Section 5 compares experimentally our algorithm with state of the art techniques.

2 Preliminaries and Related Work

Given a training set $S = \{(x_1, y_1), ..., (x_N, y_N)\}$, where $x_i \in R^d$ and $y \in \{-1, +1\}$, we are interested in learning a linear classifier: $h_w(x) = sign(\langle x, w\rangle)$

where $w \in R^d$ is the model parameter set to be learned. The optimization problem can be written as follows:

$$\begin{array}{lll} \min_w & \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{i=1..N}\xi_i & \\ subject\,to & y_i\langle x_i, w\rangle \geq 1 - \xi_i & \forall i \\ & \xi_i \geq 0 & \forall i \end{array} \quad (1)$$

where λ is the regularization parameter of the SVM, ξ_i are non-negative variables which represent the loss for the case where the margin constraint is violated for example x_i. Introducing the hinge loss function $hinge(z) = \max(0, z)$, we obtain the equivalent unconstrained problem:

$$\min_w f(w) with f(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{i=1..N}\max(0, 1 - y_i\langle x_i, w\rangle) \quad (2)$$

where the second term, denoted $R_{emp}(w) = \frac{1}{N}\sum_{i=1..N}\max(0, 1 - y_i\langle x_i, w\rangle)$ is a upper bound of the empirical risk.

A simple approach to solve this problem is to use the sub-gradient method, which is proved to converge to the global minimum [13]. Pegasos [10] is an example of this type of algorithm which requires a number of iterations, independent of N, that scales with $O(1/\lambda\epsilon)$ to reach a given ϵ accuracy. However, this algorithm lacks a good stopping criterion as we observed in our experiments that the primal value may oscillate during training.

The idea of using cutting planes method for SVM-like problem was first introduced in [14,11], where the authors proposed to approximate the original problem with many linear constraints sharing a slack variable by another one with much less constraints. Their algorithm builds an approximated problem by iteratively adding the most violated constraint every iteration, until the solution of the approximate problem does not violate any constraint in the original problem more than ϵ, which means that the two problems are close enough.

Recently, [12] proposed to use a variant of bundle methods for minimizing a convex regularized risk. It is equivalent to the cutting planes method of [11] in our particular case of linear SVM learning, however its simplicity is much appealing, and our work is inspired by this one. Actually, the objective function in (2) can be lower bounded by a quadratic function with a cutting planes technique. Every iteration, one adds a new lower bound built with the (sub)gradient of the $R_{emp}(w)$ at the current solution. The approximate problem at iteration t is then:

$$\min_w \frac{\lambda}{2}\|w\|^2 + max\left\{\langle a_{j+1}, w\rangle + b_{j+1}\right\} \forall j = 1..t-1 \quad (3)$$

where every term $\langle a_{j+1}, w\rangle + b_{j+1}$ is a cutting plane lower bound of R_{emp} computed at the solution at iteration j, w_j. This problem can be solved in its dual form, and its solution w_t is used for computing a_{t+1} and b_{t+1}. Note that the minimum of the lower bound increases every iteration so that the gap between the minimum observed value of the primal and the minimum of the lower bound decreases. In [12] the number of iterations required for reaching a gap less than ϵ is proved to be $O(1/\lambda\epsilon)$.

3 Primal Subgradient and Subdifferential

We present now useful properties of the primal objective that we want to minimize and discuss what the subgradient and the subdifferential of $f(w)$ (which will be used in Sect. 4) look like. Properties of interest of $f(w)$ are its convexity, its piecewise quadratic form, and its non differentiability which prevents the use of a number of smooth optimization methods.

Actually, $f(w)$ is differentiable everywhere but on hyperplanes $H^i : 1 - y_i\langle x_i, w\rangle = 0$. Every hyperplane H^i divides the parameter space into two half-spaces: $H_0^i : \{w|1 - y_i \langle x_i, w\rangle \leq 0\}$ and $H_1^i : \{w|1 - y_i \langle x_i, w\rangle \geq 0\}$. Hence, the N hyperplanes divide the parameter space into many polytopes: $C^k = \bigcap_{i=1..N} H^i_{\sigma_i^k}$ where $\sigma_i^k \in \{0, 1\}$. We note $I^k = \{i|\sigma_i^k = 1\}$ the set of active hyperplanes in C^k. Within any polytope C^k $f(w)$ is quadratic.

In order to optimize $f(w)$ one has to rely on the notions of subgradient and of subdifferential. Let $h : R^d \to R$ be a convex function. A vector $u \in R^d$ is called a subgradient of h at x_0 if:

$$h(x) \geq h(x_0) + \langle x - x_0, u\rangle \quad \forall x \in R^d \tag{4}$$

The set of all subgradients of h at x_0 is called the subdifferential at x_0, and is noted $\partial h(x_0)$. The subdifferential is a nonempty, convex and compact set. The subdifferential of the objective $f(w)$ has a particular form, as we show now.

Theorem 1. *The subdifferential of the primal objective at w is defined as:*

$$\partial f(w) = \left\{ \begin{array}{l} \lambda w + \sum_{i:y_i\langle x_i,w\rangle<1} \frac{-y_i x_i}{N} \\ + \sum_{i:y_i\langle x_i,w\rangle=1} (-\beta_i \frac{y_i x_i}{N}) \end{array} \Big| \ \forall i \beta_i \in [0, 1] \right\} \tag{5}$$

where every vector $\beta = \beta_1, ..., \beta_N$ corresponds to a particular subgradient of $f(w)$.

Proof. We base our proof on the two following results, both from [15].

Theorem 2. *Let $\{f_j : R^d \to R, j = 1, ..., m\}$ be a set of convex and differentiable functions, then the subdifferential of $f = \max_{j=1..m} f_j$ is*

$$\partial f(x) = conv\{\nabla f_j(x)|j \in I(x)\} \tag{6}$$

where $I(x) = \{i|f_i(x) = \max_j f_j(x)\}$, $\nabla f_j(x)$ stands for the gradient of $f_j(x)$ at x, and conv(.) stands for the convex hull of a vector set.

Proposition 1. *Let $\{f_j : R^d \to R, j = 1, ..., m\}$ be a set of convex functions and let $f = \sum_{j=1..m} f_j$, then*

$$\partial f(x) = \left\{ u = \sum_{j=1}^{m} u_j \Big| (u_1, .., u_m) \in \partial f_1(x) \times ... \times \partial f_m(x) \right\} \tag{7}$$

Algorithm 1. Global

Input: $S = \{(x_1, y_1), ..., (x_N, y_N)\}$, λ, ϵ
$t \leftarrow 0$, $w_0 \leftarrow 0$, $\tilde{w} \leftarrow 0$, $v_0 \leftarrow 0$
repeat
  $t \leftarrow t + 1$
  $g_t(w) \leftarrow LowerBound(w_{t-1}, \tilde{w}_{t-1}, v_{t-1})$
  $\tilde{w}_t \leftarrow argmin_w g_t(w)$ and $v_t \leftarrow min_w g_t(w) = g_t(\tilde{w}_t)$
  $w_t \leftarrow linesearch(f, \Delta_t)$ where $\Delta_t \leftarrow \tilde{w}_t - w_{t-1}$
  $\gamma_t \leftarrow f(w_t) - v_t$
until $\gamma_t < \epsilon$

We begin with the definition of the subdifferential of an elementary function $\max(0, \frac{1-y_i\langle x_i, w\rangle}{N})$, which is differentiable except for w such that $y_i\langle x_i, w\rangle = 1$. For w such that $y_i \langle x_i, w\rangle \neq 1$, the function is differentiable and its subdifferential resumes to its gradient. In other cases using Theorem 2, we get:

$$\partial\left(max(0, \frac{1 - y_i \langle x_i, w\rangle}{N})\right) = conv\left\{0, -\frac{y_i x_i}{N}\right\} = \left\{-\beta_i \frac{y_i x_i}{N} \middle| \beta_i \in [0, 1]\right\} \quad (8)$$

Then the subdifferential of an elementary function is given by:

$$\partial\left(\max(0, \frac{1 - y_i \langle x_i, w\rangle}{N})\right) = \begin{cases} 0 & if\ y_i \langle x_i, w\rangle > 1 \\ \{-\beta_i \frac{y_i x_i}{N} / \beta_i \in [0, 1]\} & if\ y_i \langle x_i, w\rangle = 1 \\ -\frac{y_i x_i}{N} & if\ y_i \langle x_i, w\rangle < 1 \end{cases} \quad (9)$$

Next, applying Proposition 1 to $f(w)$ which is the sum of convex functions (Cf. (2)), together with the result of (9), we get the expected result. □

4 Algorithm

The pseudo-code is given in Algorithm 1. Iteration t begins with the building of a simple lower bound $g_t(w)$ which is built based on the cutting plane method. Then the solution $\tilde{w}_t$ minimizing this lower bound, and the minimum value v_t, are found by quadratic programming. Finally a linesearch is performed along the line from the solution in previous iteration, w_{t-1}, to $\tilde{w}_t$, yielding the new current solution at the t^{th} iteration, w_t. The algorithm stops once the gap $\gamma_t = f(w_t) - v_t$ is less than ϵ, which means that an ϵ-solution has been reached. The key point is that the minimum of the lower bound increases every iteration and approaches the minimum of the objective function $f(w)$. Consequently, the gap between the current value of the primal and the minimum of the lower bound decreases every iteration. The way the lower bound is built guarantees the convergence, while the linesearch procedure guarantees the decrease of $f(w_t)$.

4.1 Building a Lower Bound

In this section, we describe how we build, at the t^{th} iteration, a lower bound $g_t(w)$ of $f(w)$. Then, we describe its minimization.

Our lower bound is inspired by the recent work [12] which is based on the cutting plane method which we present first. As proposed in [12], we approximate $R_{emp}(w)$ only and let the quadratic term $\frac{\lambda}{2}\|w\|^2$ aside (Cf. (2)). This is a key issue of the technique which provides a fast rate of convergence.

Cutting Plane Technique: Whatever w_0, the convex function $R_{emp}(w)$ can be lower bounded by using the inequality $R_{emp}(w) \geq \langle a_{w_0}, w\rangle + b_{w_0}$ where a_{w_0} is the (sub)gradient of $R_{emp}(w)$ at w_0, b_{w_0} is the offset which can be obtained from the equality at w_0: $\langle a_{w_0}, w\rangle + b_{w_0} = R_{emp}(w_0)$. Then, the function $g_{cp}^{w_0}(w) = \frac{\lambda}{2}\|w\|^2 + \langle a_{w_0}, w_0\rangle + b_{w_0}$ may be seen as a cutting plane approximation of $f(w)$ (with equality at w_0) which is accurate for w lying in the vicinity of w_0. Figure 1-a shows the cutting plane approximation of $R_{emp}(w)$ at a particular w_0 and Fig. 1-d shows the corresponding quadratic lower bound $g_{cp}^{w_0}(w)$ of $f(w)$. If $R_{emp}(w)$ is not differentiable at w_0, one can use any (e.g. random) subgradient (Fig. 1-b). However the quality of the approximation may be poor since equality $g_{cp}^{w_0}(w) = f(w)$ holds for w_0 only (Fig. 1-e).

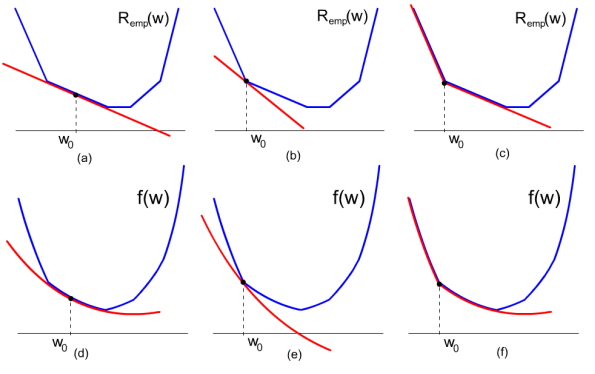


Fig. 1. Building a Lower Bound (LB, in red) of $R_{emp}(w)$ (top) and of $f(w)$ (bottom) based on cutting planes. Left ((a) and (d)): LB of the risk, $\langle a_{w_0}, w_0\rangle + b_{w_0}$, and of the objective, $g_{cp}^{w_0}(w)$, using the gradient in the differentiable case. Middle ((b) and (e)): Same as left where $g_{cp}^{w_0}(w)$ is defined using one subgradient, in the non differentiable case. Right ((c) and (f)): LB of $R_{emp}(w)$ using the maximum over all cutting planes built from all subgradients (c) and corresponding lower bound of $f(w)$ using (12) (f).

The lower bound built from one cutting plane only $g_{cp}^{w_0}(w)$ is not a good approximation of $f(w)$. A solution is to build iteratively an increasingly accurate lower bound by adding, every iteration, a new cutting plane at the current solution [12] (Cf. (3)). At iteration t one uses the lower bound:

$$g_t(w) = \frac{\lambda}{2}\|w\|^2 + max_{j=1..t-1}\left\{\langle a_{w_j}, w\rangle + b_{w_j}\right\} \tag{10}$$

where w_j stands for the solution at iteration j.

Lower Bound: The main difference between our lower bound and the one in (10) is that we build the lower bound from only three elementary lower bounds $g_t(w) = \max(g_t^1(w), g_t^2(w), g_t^3(w))$. Hence we get an optimization problem that is very fast to solve. We detail now these three lower bounds.

The first lower bound is a cutting plane approximation at $\tilde{w}_{t-1}$, the minimum of the lower bound in previous iteration:

$$g_t^1(w) = g_{cp}^{\tilde{w}_{t-1}}(w) \stackrel{not}{=} \frac{\lambda}{2}\|w\|^2 + \langle a_t^1, w\rangle + b_t^1 \tag{11}$$

where, as before, a_t^1 is a (sub)gradient of $R_{emp}(w)$ at $\tilde{w}_{t-1}$, and $b_t^1 = R_{emp}(\tilde{w}_{t-1}) - \langle a_t^1, \tilde{w}_{t-1}\rangle$. The idea behind this is that we want to improve the approximation quality around $\tilde{w}_{t-1}$ (In particular: $g_t^1(\tilde{w}_{t-1}) = f(\tilde{w}_{t-1})$).

The second lower bound is a simplification of the lower bound $g_{t-1}(w)$ of the previous iteration. The role of this lower bound, together with $g_t^1(w)$, is to guarantee a minimum improvement of the lower bound, thus providing a fast convergence rate of the algorithm (as we will see in Section 4.3). For this, We use a quadratic function $g_t^2(w) = \frac{\lambda}{2}\|w\|^2 + \langle a_t^2, w\rangle + b_t^2$, which is minimized at $\tilde{w}_{t-1}$, and takes the same minimum value than $v_{t-1} = g_{t-1}(\tilde{w}_{t-1})$. These conditions determine uniquely a_t^2 and b_t^2. Note that by doing so $g_t^2(w) \leq g_{t-1}(w) \forall w$, so that $g_t^2(w)$ is also a lower bound of $f(w)$.

The third lower bound is an approximation of $f(w)$ at the solution in previous iteration, w_{t-1}. The way this solution is found (it is the result of the line search step described in the next section) makes it very often that $R_{emp}(w)$ (hence $f(w)$) is not differentiable at w_{t-1}. In theory, we could use a cutting plane approximation with any subgradient (e.g. steepest descent subgradient). However, we observed experimentally that this strategy may fail when dealing with high dimensional data (it may happen that the linesearch direction is not a descent direction of $f(w)$). To get a better approximation that equals $f(w)$ on a neighbourhood of w_{t-1}, we rather exploit the whole subdifferential. The idea is to use the maximum of all cutting plane approximations $\langle a, w\rangle + b_a$ (note that the offset b_a depends on a) built from all subgradients in the subdifferential. One then gets the following lower bound:

$$g_t^3(w) = \frac{\lambda}{2}\|w\|^2 + \max_{a \in \partial R_{emp}(w_{t-1})} (\langle a, w\rangle + b_a) \tag{12}$$

This is a better lower bound of $f(w)$ as shown in Figure 1 where a pair of (extreme) cutting plane approximations of $R_{emp}(w)$ are shown in Figure 1-c while the lower bound found by exploiting all cutting plane approximations built with all subgradients is shown in Fig. 1-f. We exploit this idea but we use a faster way to build the above approximation. The idea is to split the non-differentiable and differentiable parts of $R_{emp}(w)$ at w_{t-1}. Let $LA = \{i | \langle y_i x_i, w_{t-1}\rangle = 1\}$ denote the set of index of non-differentiable terms in $R_{emp}(w)$ at w_{t-1} (note that we omit the dependency of LA on w_{t-1} for clarity). Then:

$$R_{emp}(w) = \sum_{i \notin LA} \max(0, \frac{1 - \langle y_i x_i, w_{t-1}\rangle}{N}) + \sum_{i \in LA} \max(0, \frac{1 - \langle y_i x_i, w_{t-1}\rangle}{N}) \tag{13}$$

Let consider the cutting plane lower bound of the differentiable part of $R_{emp}(w)$ at w_{t-1}, $\langle a_t^3, w\rangle + b_t^3$, with $a_t^3 = \frac{1}{N}\sum_{i:\langle y_i x_i, w_{t-1}\rangle<1}(-y_i x_i)$ being the gradient of the differentiable part, and $b_t^3 = \frac{1}{N}\sum_{i:\langle y_i x_i, w_{t-1}\rangle<1} 1$. Then our third lower bound is defined as:

$$g_t^3(w) = \frac{\lambda}{2}\|w\|^2 + \langle a_t^3, w\rangle + b_t^3 + \sum_{i\in LA} \max(0, \frac{1 - \langle y_i x_i, w_{t-1}\rangle}{N}) \tag{14}$$

Putting all together, our lower bound in iteration t, $g_t(w)$ is defined as:

$$\begin{array}{l} g_t(w) = \frac{\lambda}{2}\|w\|^2 + \max(\langle a_t^1, w\rangle + b_t^1, \langle a_t^2, w\rangle + b_t^2, \\ \qquad \langle a_t^3, w\rangle + b_t^3 + \sum_{i\in LA} \max(0, \frac{1-\langle y_i x_i, w_{t-1}\rangle}{N})) \end{array} \tag{15}$$

Minimizing the Lower Bound: To minimize $g_t(w)$, we rewrite the problem in a constrained form with slack variables then we solve this constrained minimization problem in its dual form by quadratic programming. The size of this dual problem is very small with respect to the original problem, there are only $K + 3$ variables where K is the number of hyperplanes that cross w_{t-1}.

The constrained minimization problem of $g_t(w)$ can be written as:

$$\begin{array}{lll} \min_{w,\xi,\xi_i} & \frac{\lambda}{2}\|w\|^2 + \xi & \\ s.t. & \langle a^1, w\rangle + b^1 \le \xi & \\ & \langle a^2, w\rangle + b^2 \le \xi & \\ & \langle a^3, w\rangle + b^2 + \frac{1}{N}\sum_{i\in' LA'} \xi_i \le \xi & \\ & 1 - \langle y_i x_i, w_{t-1}\rangle \le \xi_i & \forall i \in LA \\ & \xi_i \ge 0 & \forall i \in LA \\ & \xi \ge 0 & \end{array} \tag{16}$$

Following standard derivation, this optimization problem can be solved by writing the Lagrangian then noticing that the solution is given by a saddle point of the Lagrangian, that must be minimized wrt. parameters w, ξ, ξ_i and maximized wrt. Lagrange multipliers. Omitting details, one can get the dual form:

$$\begin{array}{lll} \min_{\alpha} & \frac{1}{2\lambda}\beta^T A_t^T A_t \beta + \beta^T B & \\ s.t. & \alpha_i \ge 0 & \forall i \in LA \\ & \gamma_i \ge 0 & \\ & \gamma_1 + \gamma_2 + \gamma_3 \le 1 & \\ & \alpha_i \le \frac{\gamma_3}{N} & \forall i \in LA \end{array} \tag{17}$$

where $A = [...(x_i y_i)..., -a_1, -a_2, -a_3]$, $B = [...1..., -b_1, -b_2, -b_3]$ and $\beta = [...\alpha_i..., \gamma_1, \gamma_2, \gamma_3]$ is the vector of Lagrange multipliers. Solving this optimization problem resumes to a quadratic programming problem of limited size $(K + 3)$ (usually K is less than 20). Once the problem is solved in dual, the primal solution may be obtained by using the equality at saddle point: $\tilde{w}_t = \frac{\sum_{i\in LA} \alpha_i x_i y_i - \gamma_1 a_1 - \gamma_2 a_2 - \gamma_3 a_3}{\lambda}$.

It is straightforward to see that if w_{t-1} is not the optimum of $f(w)$ then the search direction given by the solution $\tilde{w}_t$ minimizing the lower bound is a

descent direction of $f(w)$ at w_{t-1}. Actually, because $g_t(w)$ is convex, the direction from any non optimum point (for instance w_{t-1}) to $\tilde{w}_t$ is a descent direction. Furthermore, since $g_t(w)$ matches exactly $f(w)$ on a neighbourhood of w_{t-1} the direction is also a descent direction of $f(w)$.

4.2 Optimal Line Search

The line search procedure (line 7 in Algorithm 1) is used to find an optimum solution along a line from the solution in previous iteration, w_{t-1}, to the minimum $\tilde{w}_{t-1}$ of the current lower bound $g_t(w)$. To improve readability we consider the line search when starting from a point w_s and along a particular direction Δ. This yields the one-dimensional minimization problem:

$$\eta^* = arg\min_{\eta} g(\eta) \tag{18}$$

where $g(\eta) = f(w_s + \eta\Delta)$. Let imagine that we examine $w = w_s + \eta\Delta$ for increasing η, then w will successively cross hyperplanes H^i that are at frontiers between polytopes. Without loss of generality (by renumbering hyperplanes) assume that w crosses successively hyperplanes H^1, H^2, H^3... (see Fig. 2). The intersection with hyperplane $H^i : 1 - y_i \langle x_i, w\rangle = 0$ occurs (if it exists) for a particular value of η, denoted η^i, which may be computed according to:

$$\begin{array}{rl} & 1 - y_i \left\langle x_i, (w_s + \eta^i \Delta)\right\rangle = 0 \\ \iff & \eta^i = \frac{1 - y_i \langle x_i, w_s\rangle}{y_i \langle x_i, \Delta\rangle} \end{array} \tag{19}$$

$g(\eta)$ is a piecewise quadratic function (see figure 2), which is not differentiable at η^n. Within any segment $\left[\eta^n, \eta^{n+1}\right]$, $g(\eta)$ is quadratic and equals:

$$g^n(\eta) = \frac{\lambda}{2}\|w_s + \eta\Delta\|^2 + \frac{1}{N}\sum_{i \in I^{k(n)}} (1 - y_i \langle x_i, (w_s + \eta\Delta)\rangle) \tag{20}$$

where $k(n)$ stands for the index of the polytope corresponding to the n^{th} segment (and $I^{k(n)}$ is defined as in Section 3). To determine $\eta*$, one sets $\frac{\partial g^n(\eta)}{\partial \eta} = 0$. In the n^{th} segment this yields an optimal stepsize η^n_{opt}:

$$\eta^n_{opt} = \frac{\sum_{i \in I^{k(n)}} \frac{y_i \langle x_i, \Delta\rangle}{N} - \lambda \langle w_s, \Delta\rangle}{\lambda \langle \Delta, \Delta\rangle} \tag{21}$$

Two cases may arise. Either $g(\eta)$ is differentiable at $\eta*$ or it is not. In the first case, there exists one particular n such that $\eta^n \leq \eta^n_{opt} \leq \eta^{n+1}$ and η^n_{opt} is the solution of (18). Otherwise,$\eta^* = \eta^{\hat{n}}$ and whatever n, η^n_{opt} does not belong to the n^{th} segment. In this case it is easy to show that the solution of (18) satisfies $\eta^n_{opt} \leq \eta^{\hat{n}} \leq \eta^{n-1}_{opt}$. With this discussion in mind, our algorithm examines successively the existence of $\eta*$ in segments $[0, \eta^1), [\eta^1, \eta^2), ..., [\eta^{L-1}, \eta^L), [\eta^L, +\infty)$ until the solution (i.e. one of the two cases arises) is found.

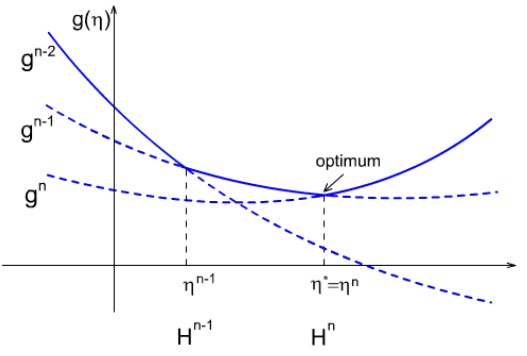


Fig. 2. Line search

4.3 Convergence Analysis

In this section, we analyse the convergence rate of the proposed algorithm based on the improvement of the lower bound.

Theorem 3. *Let G be an upper bound of the norm of the subgradient then the Algorithm 1 reaches a gap of at most ϵ after* $\left[\log_2\left(\frac{\lambda}{4G^2}\right) + \frac{8G^2}{\lambda\epsilon} - \frac{8G^2}{\lambda}\right]$ *iterations.*

Let $\gamma_t = f(w_t) - v_t$ be the gap between the current objective value and the minimum of the lower bound at iteration t. We want to find the maximum number of iterations to reach a gap γ_t less than ϵ. For simplicity, we consider $g_t^*(w) = \max(g_t^1(w), g_t^2(w))$ (hence $v_t = \min_w g_t(w) \geq \min_w g_t^*(w)$) and study how the minimum of the lower bound v_t behaves w.r.t. $\min_w g_t^*(w)$. Note that by construction $g_t^1(w)$ and $g_t^2(w)$ have the same quadratic component, and that $g_t^1(\tilde{w}_{t-1}) > g_t^2(\tilde{w}_{t-1})$.

Let note $w^* = argmin(g_t^*(w))$. By definition of our lower bounds only two cases may happen: either w^* minimizes $g_t^1(w)$ (cf. Fig. 3a), or $g_t^1(w^*) = g_t^2(w^*)$ (cf Fig. 3c), the limit case is illustrated in Fig. 3b where $\min g_t^1(w) = \frac{g_t^1(\tilde{w}_{t-1}) + g_t^2(\tilde{w}_{t-1})}{2}$. We examine now the two cases in more details.

In the first case, w^* minimizes $g_t^1(w)$ then $\min_w g_t^1(w) \geq \frac{g_t^1(\tilde{w}_{t-1}) + g_t^e(\tilde{w}_{t-1})}{2}$, and:

$$\begin{aligned} v_t &\geq \min g_t^1(w) \geq \tfrac{g_t^1(\tilde{w}_{t-1}) + g_t^2(\tilde{w}_{t-1})}{2} \\ &\geq v_{t-1} + \tfrac{g_t^1(\tilde{w}_{t-1}) - v_{t-1}}{2} \geq v_{t-1} + \tfrac{f(\tilde{w}_{t-1}) - v_{t-1}}{2} \end{aligned} \tag{22}$$

where we have used the equality $g_t^1(\tilde{w}_{t-1}) = f(\tilde{w}_{t-1})$ by construction of g_t^1, and that $v_{t-1} = g_t^2(\tilde{w}_{t-1})$ by construction of g_t^2.

Furthermore, the linesearch at iteration t makes that $f(w_t) \leq f(w_{t-1})$, and the one at iteration $t-1$ makes that $f(w_{t-1}) \leq f(\tilde{w}_{t-1})$, and hence:

$$\begin{aligned} \gamma_t = f(w_t) - v_t &\leq f(w_t) - v_{t-1} - \tfrac{f(\tilde{w}_{t-1}) - v_{t-1}}{2} \\ &\leq f(w_{t-1}) - v_{t-1} - \tfrac{f(w_{t-1}) - v_{t-1}}{2} \\ &\leq \gamma_{t-1} - \tfrac{\gamma_{t-1}}{2} \end{aligned} \tag{23}$$

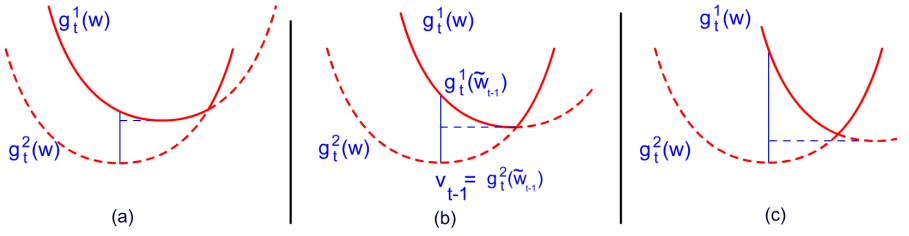


Fig. 3. Minimizing g_t^*

Let examine briefly the second case now. We do not provide the full proof since it is simple but long. We rather give hints. We know that w^* satisfies $g_t^1(w^*) = g_t^2(w^*)$, which defines a hyperplane $\langle a_t^1, w\rangle + b_t^1 = \langle a_t^2, w\rangle + b_t^2$ (since $g_t^1(w)$ and $g_t^2(w)$ have the same quadratic component). Furthermore, one can easily show that w^* must lie on the segment line delimited by the two minimum points, $\frac{-a_t^1}{\lambda}$ and $\frac{-a_t^2}{\lambda}$, of $g_t^1(w)$ and $g_t^2(w)$. After simple manipulations, one gets that the minimum $g_t^*(w)$ is given by $g_t^*(w^*) = g_t^2(\tilde{w}_{t-1}) + \frac{\lambda}{2}\frac{(g_t^1(\tilde{w}_{t-1}) - g_t^2(\tilde{w}_{t-1}))^2}{(a1-a2)^2} = v_{t-1} + \frac{(f(\tilde{w}_{t-1}) - v_{t-1})^2}{(a1-a2)^2}$. This may be used to build a upper bound on γ_t since:

$$\begin{aligned}\gamma_t = f(w_t) - v_t &\leq f(w_t) - v_{t-1} - \tfrac{\lambda}{2}\tfrac{(f(\tilde{w}_{t-1}) - v_{t-1})^2}{(a2-a3)^2} \\ &\leq f(w_{t-1}) - v_{t-1} - \tfrac{\lambda}{2}\tfrac{(f(w_{t-1}) - v_{t-1})^2}{(a2-a3)^2} \\ &\leq \gamma_{t-1} - \tfrac{\lambda}{8G^2}\gamma_{t-1}^2\end{aligned} \tag{24}$$

where G is an upper bound on the norm of gradient.

Putting all together, we get $\gamma_t \leq \frac{\gamma_{t-1}}{2}\min(1, \frac{\lambda}{4G^2}\gamma_{t-1})$. We see that if $\gamma_{t-1} \geq \frac{4G^2}{\lambda}$ then the inequality shows that $\gamma_t \leq \frac{\gamma_{t-1}}{2}$ and the gap is at least divided by two. Then the condition $\gamma_{t-1} \geq \frac{4G^2}{\lambda}$ happens for at most $T_0 = \log_2\left(\frac{\lambda}{4G^2}\right)$ steps because $\gamma_0 = 1$. Then, we have $\gamma_t \leq \frac{\lambda}{8G^2}\gamma_{t-1}^2$. To estimate the number of iterations required to reach $\gamma_t \leq \epsilon$, we introduce a function $\gamma(t)$ which is its upper bound. Solving differential equation $\gamma'(t) = -\frac{\lambda}{8G^2}\gamma^2(t)$ with boundary condition $\gamma(T_0) = 1$ gives us $\gamma(t) = \frac{8G^2}{\lambda(t + \frac{8G^2}{\lambda} - T_0)}$ is an upper bound of γ_t. Since $\gamma(t) \leq \epsilon \iff t \geq \frac{8G^2}{\lambda\epsilon} + T_0 - \frac{8G^2}{\lambda}$, the solution is reached with accuracy ϵ within $\left\lceil \log_2\left(\frac{\lambda}{4G^2}\right) + \frac{8G^2}{\lambda\epsilon} - \frac{8G^2}{\lambda} \right\rceil$ iterations.

4.4 Complexity and Shrinking

The algorithmic complexity of our algorithm has three main components, the computation of the subdifferential (5), the quadratic programming problem, and the linesearch procedure (Sect. 4.2). However, the complexity of the quadratic programming problem is rather small, and the subdifferential is iteratively updated along the linesearch (whenever a hyperplane is crossed). Finally, the algorithmic complexity is dominated by the complexity of the linesearch. Estimation

of η^n requires 2 dot products, and sorting positive η^n is upper bounded by $N \log_2 N$. Then the overall algorithm cost is about $2Nd + N \log_2 N$, where d is the dimension of the data. In case of sparse data the complexity is related to the average number of non null components rather than to d.

In practice, w_{t-1} , w_t and $\tilde{w}_t$ are usually close so that the set of active hyperplanes may be expected not to change much between two iterations. The idea of shrinking is to take this into account to design a more efficient algorithm by considering a limited number of hyperplanes only. Let consider a convex region Q around w_{t-1} (e.g. a ball) and let H_I denote the set of inactive hyperplanes with respect to Q (i.e. hyperplanes that don't cross Q), and let $H_A = \{1, .., N\} \backslash H_I$ the set of active hyperplanes. Then $f(w)$ coincides with $f^Q(w)$ on Q, where:

$$f^Q(w) = \frac{\lambda}{2} w^2 + \sum_{i \in H_A} \max(0, 1 - \langle y_i x_i, w \rangle) - \langle A_{H_I}, w \rangle + C_{H_I} \tag{25}$$

where $A_{H_I} = \sum_{i \in H_I, y_i x_i w_t < 1} y_i x_i$ and $C_{H_I} = \sum_{i \in H_I, y_i x_i w_t < 1} 1$. We may consider the simplified line search problem $\min_\eta f^Q(w_t + \eta \Delta_t)$ which can be solved with the algorithm in Sect. 4.2. If the solution $w_{opt} \in Q$ then w_{opt} is a local minimum of f because the approximation is correct within Q. Hence w_{opt} is a global minimum of f since it is convex. A simple method for implementing these ideas consists in starting with a ball centered at w_{t-1} and with a small radius so that number of active hyperplanes is small enough (e.g 10). If the obtained solution w_{opt} belongs to the ball then we have our solution $w_t = w_{opt}$. Otherwise, we increase the radius of the ball (i.e. the number of active hyperplanes) and start a new linesearch until we find a solution which is correct with respect to the ball considered. This approach is described in Algorithm 2. To simplify the presentation, we assume that all the distances from w_{t-1} to all hyperplanes are precomputed and sorted in ascending order so that: $dist(w_{t-1}, H_{h_1}) \leq dist(w_{t-1}, H_{h_2}) \leq ... \leq dist(w_{t-1}, H_{h_N})$. Also we consider here that N is a power of 2 and that we use $log(N)$ balls, the k^{th} one corresponding to 2^k active hyperplanes.

Algorithm 2 outputs w_{t+1} without considering all hyperplanes in the linesearch, but it requires to compute the distance from the current solution w_t to the N hyperplanes. The number of dot products to be calculated is then still linear with N, while the complexity of the linesearch $(M \log_2 M)$ now depends of the average number of active hyperplanes M (possibly N). It is possible to go further by maintaining balls with different centers. Imagine that at iteration t, we define a ball $B_k = Ball(w_t, R_k)$ that is large enough to include all forthcoming solutions $w_{t+1}, w_{t+2}, ..., w_{final}$. Then all hyperplanes that are inactive with respects to B_k could be completely ignored in the following iterations. This means we may avoid computing all distances each iteration and instead reuse balls from previous iterations up to a certain extent. We do not detail this algorithm here because we lack room for that, we only give the idea. The main difference with the previous simpler shrinking algorithm is that the balls that are maintained during linesearch are not all centered on w_{t-1}, but may be centered on w_{t-2}, w_{t-3}, ... Once a solution of the reduced line search is found (with the set of active

Algorithm 2. line search one center

Input: $w, \Delta, H_1, ..., H_N$
$NB = log_2 N$
Compute distances from w to all hyperplanes and renumber hyperplanes with increasing distance.
for k=1:NB **do**
  Set R_k = distance to hyperplane number 2^k.
  Set L = number of active hyperplanes in $Ball(w, R_k)$.
  Compute $\eta^1, ..., \eta^L, g^1, ..., g^{L+1}$ with active hyperplanes $H_1, ..., H_{2^k}$.
  $\eta_{opt} = LineSearch(\eta^1, ..., \eta^L, g^1, ..., g^{L+1})$
  $w_{opt} = w + \eta_{opt}\Delta$
  if $w_{opt} \in Ball(w, R_k)$ return w_{opt}
end for

hyperplanes of a ball B_j), one has to check if this solution is valid, that is $w_{opt} \in \cap_{k=j}^{NB} B_k$. If this is not the case, one has to recompute distances with hyperplanes in the smallest ball including the solution and restart the procedure.

5 Experimental Results

We performed binary classification experiments on 6 datasets (Table 1) to compare our algorithm (named Hyperpass) with two reference methods, the subgradient algorithm Pegasos (on-line version) of [10] and the Cutting Planes algorithm svmperf of [11]. We used our own implementation of Pegasos where we set the parameter K (number of examples uses to compute sub-gradients) to 1000 whereas we used the available code for svmperf[1]. Cross validation over the full datasets is used to determine the parameter λ for each dataset ($\lambda = 10^{-3}, 10^{-4}, .., 10^{-8}$). To fairly compare the behaviour of the three algorithms w.r.t. the number of learning iterations, one iteration of Pegasos stands for $\frac{N}{K}$ subiterations (using K examples). The complexity of an iteration is $O(N)$ whatever the algorithm.

Figure 4 shows the evolution of the Primal objective as a function of the number of iterations. Actually, it plots the gap obtained for the current solution $gap(w) = f(w) - v^*$ where v^* stands for the best lower bound value observed during training (in Hyperpass and Cutting Plane runs). On the one hand, on low-dimensional datasets (first row of plots), one can see that Hyperpass both requires much less iterations than Pegasos and svmperf to reach a good ϵ-solution for all these datasets and that it converges to a more accurate solution. On the other hand, for high-dimensional datasets, Hyperpass may be slower than Pegasos in the first 20 to 100 iterations, then its convergence is faster and more accurate. Whatever the dataset Hyperpass is much faster and accurate than svmperf. Similar to these above situations, we observed that Hyperpass attends the minimum error rate faster than Pegasos in low-dimensional settings (roughly

[1] http://svmlight.joachims.org/svm_perf.html

Table 1. Datasets

Dataset	Adult9	Covertype	Web8	CCAT	C11	RealSim
#examples	48842	581012	64700	804414	697641	72309
dimension	123	54	300	47236	47236	20958

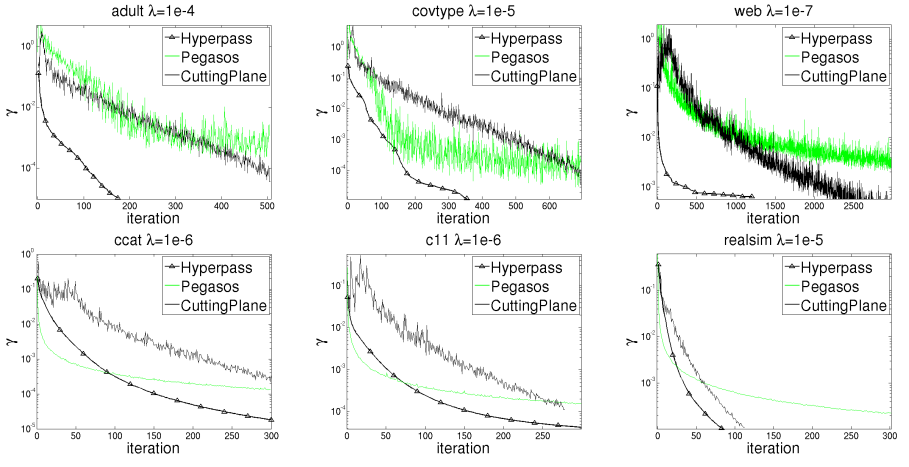


Fig. 4. Evolution of the gap (Primal-v*) as a function of the number of iterations

when the gap is about 1% of the primal value); for high-dimentional data, Pegasos achieves the minimal error rate with less iterations. The three methods reach same performance after convergence (svmperf being the slowest one in any case).

Furthermore the complexity of an iteration in Hyperpass may be drastically reduced using shrinking as shown experimentally. Figures 5-a,b plot the same quantity as in Figure 4 but as a function of the number of dot products normalized by N (it is a number equivalent to a number of iterations without shrinking) for two low dimensional datasets [2]. The curves do not change for Pegasos and svmperf while we see that the total number of operations actually required by Hyperpass is actually much reduced. For instance with the *web* dataset, there are less than $250 * N$ dot product computations in about 1500 iterations. Figure 5-c gives more insight about the efficiency of the shrinking method by plotting the number of dot products actually computed every iteration. For visibility reason, we only present the result for three datasets *adult*, *web*8 and *covtype*. As can be seen the number of operations in one iteration may be up to 50 times lower than N. It is interesting to note that although *covtype* is 10 times larger than *web*8 the complexity of an iteration of *covtype* may be less than the one of *web*8.

[2] These plots compare the computational complexity of the three algorithms. Please note that a comparison based on CPU time would be unfair because the implementations might not be equally optimized.

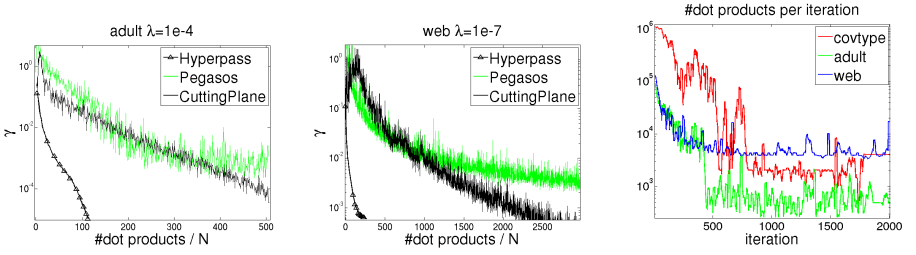


Fig. 5. Evolution of the gap (Primal-v*) as a function of the number of $\frac{\#\text{dot products}}{N}$

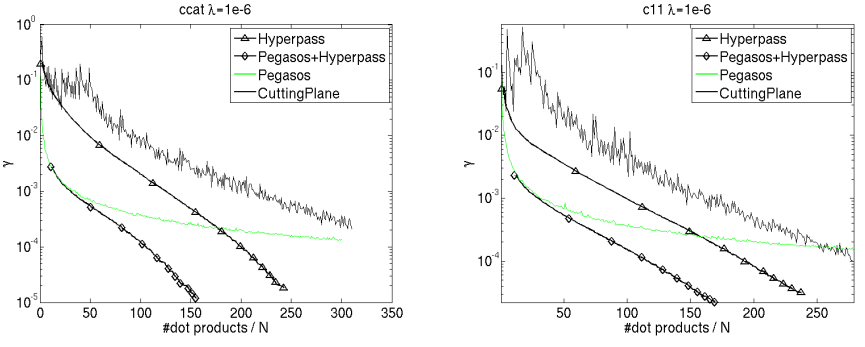


Fig. 6. Evolution of the gap (Primal-v*) as a function of $\frac{\#\text{dot products}}{N}$

For some high dimensional datasets as we said previously, Pegasos may converge faster at the beginning of the learning. But then, it requires a larger number of iterations than Hyperpass, to reach a solution that is less accurate. While Hyperpass may be seen to outperform Pegasos in the long run (i.e. to reach a very accurate solution) it may benefit from initialization, i.e. a bootstrap, using Pegasos. We investigate the use of Pegasos as a smart starter for Hyperpass for high dimensional datasets in Fig. 6. This Figures plot the evolution of the gap as a function of the number of dot products for four methods, the three already compared and a fourth one which is based on Hyperpass with an initialization given by Pegasos after 10 iterations. As may be seen on these plots, this combined Pegasos-Hyperpass algorithm performs much better than any of the three others, and converges at the same rate (the fastest one) than Hyperpass.

6 Conclusion

We presented an original algorithm for training SVM in the primal, for which we provided convergence rate analysis. We proposed an efficient shrinking technique and showed experimentally that our algorithm converges much faster than state

of the art algorithms on number of benchmark datasets, wrt. to the number of iterations and to the number of operations.

References

1. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: An application to face detection. In: CVPR 1997 (1997)
2. Platt, J.C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research (1998)
3. Joachims, T.: Making large-scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge (1999)
4. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001)
5. Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. Journal of Machine Learning Research 6, 1579–1619 (2005)
6. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. J. Mach. Learn. Res. 6, 363–392 (2005)
7. Chapelle, O.: Training a support vector machine in the primal. Neural Computation 19(5), 1155–1178 (2007)
8. Mangasarian, O.L., Musicant, D.R.: Lagrangian support vector machines. Journal of Machine Learning Research 1, 161 (2001)
9. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: ICML 2004, vol. 116. ACM Press, New York (2004)
10. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for SVM. In: ICML 2007, pp. 807–814. ACM Press, New York (2007)
11. Joachims, T.: Training linear SVMs in linear time. In: ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD), pp. 217–226 (2006)
12. Teo, C.H., Le, Q.V., Smola, A., Vishwanathan, S.V.: A scalable modular convex solver for regularized risk minimization. In: KDD 2007, pp. 727–736 (2007)
13. Nedic, A., Bertsekas, D.: Convergence rate of incremental subgradient algorithms. In: Uryasev, S., Pardalos, P.M. (eds.) Stochastic Optimization: Algorithms and Applications. Kluwer Academic, Dordrecht (2000)
14. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML 2004, pp. 104–112 (2004)
15. Bertsekas, D.P., Nedic, A., Ozdaglar, A.E.: Convex analysis and optimization. Athena Scientific, Belmont (2003)

On the Equivalence of the SMO and MDM Algorithms for SVM Training

Jorge López, Álvaro Barbero, and José R. Dorronsoro*

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

Abstract. SVM training is usually discussed under two different algorithmic points of view. The first one is provided by decomposition methods such as SMO and SVMLight while the second one encompasses geometric methods that try to solve a Nearest Point Problem (NPP), the Gilbert–Schlesinger–Kozinec (GSK) and Mitchell–Demyanov–Malozemov (MDM) algorithms being the most representative ones. In this work we will show that, indeed, both approaches are essentially coincident. More precisely, we will show that a slight modification of SMO in which at each iteration both updating multipliers correspond to patterns in the same class solves NPP and, moreover, that this modification coincides with an extended MDM algorithm. Besides this, we also propose a new way to apply the MDM algorithm for NPP problems over reduced convex hulls.

1 Introduction

Given a sample $\mathcal{S} = \{(X_i, y_i) : i = 1, \ldots, N\}$ with $y_i = \pm 1$, the standard formulation of SVM for linearly separable problems seeks [1,2] to maximize the margin of a separating hyperplane by solving the problem

$$\min \frac{1}{2}\|W\|^2 \;\text{ with }\; y_i(W \cdot X_i + b) \geq 1, i = 1, \ldots, N. \tag{1}$$

Any pair (W, b) verifying the restrictions in (1) is said to be in canonical form. In practice, however, the problem actually solved is the simpler dual problem of minimizing

$$W(\alpha) = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_i \alpha_i \;\text{ with }\; \alpha_i \geq 0, \;\sum_i \alpha_i y_i = 0. \tag{2}$$

The optimal weight W^o can be then written as $W^o = \sum \alpha_i^o y_i X_i$ and patterns for which $\alpha_i^o > 0$ are called support vectors (SV). There are quite a few proposals of algorithms to solve (2); many of them can be broadly classified into two categories that usually are discussed as independent procedures, decomposition

* All authors have been partially supported by Spain's TIN 2007–66862. The second author is kindly supported by the FPU–MEC grant reference AP2006–02285.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 288–300, 2008.

algorithms and geometrically inspired methods. Many decomposition algorithms can be traced to Platt's SMO [3] or Joachims's SVM–Light [4] algorithms. SMO, one of the most popular methods, proceeds iteratively, working at each step with a reduced set of only two multipliers, α_{i_1}, α_{i_2} and solving problem (2) exactly for them while keeping fixed all others. To stop training, SMO looks at the KKT conditions for the dual of (2). At the optimal $W^o = \sum \alpha_i^o y_i X_i$, they imply $\alpha_i^o y_i (W^o \cdot X_i + b^o - y_i) = 0$ and, thus, we have

$$\begin{aligned} \alpha_i^o > 0 &\Rightarrow y_i(W^o \cdot X_i + b^o - y_i) = 0, \\ \alpha_i^o = 0 &\Rightarrow y_i(W^o \cdot X_i + b^o - y_i) \geq 0. \end{aligned} \tag{3}$$

Hence, during training there might be two kinds of violations of these KKT conditions. The first one happens when $\alpha_i > 0$ but $y_i(W \cdot X_i + b - y_i) \neq 0$. The second one takes place if $\alpha_i = 0$ but $y_i(W \cdot X_i + b - y_i) < 0$. Platt's SMO algorithm essentially tries to choose i_2 as the index of the pattern X_i that somehow most violates these conditions for the current W and i_1 as the index that gives then a maximum decrease in $W(\alpha)$. However, and as pointed out in [5], this may lead to some difficulties as the KKT conditions only hold approximately during training. To avoid this Keerthi et al. propose in [5] two modifications to SMO and recommend the second one, Modification 2, as the most effective (see also [6], where it is shown to be equivalent to 2–vector SVM–Light); we will briefly describe it in section 2.

Turning our attention to geometric algorithms, they are usually motivated through another way of setting up SVM training, the Nearest Point Problem (NPP; see [7]) in which we want to find the nearest points W_+^* and W_-^* of the convex hulls $C(\mathcal{S}_\pm)$ of the positive $\mathcal{S}_+ = \{X_i : y_i = 1\}$ and negative $\mathcal{S}_- = \{X_i : y_i = -1\}$ sample subsets. The maximum margin hyperplane is then $W^* = W_+^* - W_-^*$ and the optimal margin is given by $\|W^*\|/2$. If we write a $W_+ \in C(\mathcal{S}_+)$ as $W_+ = \sum \alpha_p X_p$, with $\sum \alpha_p = 1$ and a $W_- \in C(\mathcal{S}_-)$ as $W_- = \sum \alpha_q X_q$, with $\sum \alpha_q = 1$ we have $W = W_+ - W_- = \sum \alpha_i y_i X_i$ with $X_i \in \mathcal{S} = \mathcal{S}_+ \bigcup \mathcal{S}_-$. We can thus state the NPP problem as follows:

$$\min \frac{1}{2}\|W\|^2 = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j X_i \cdot X_j, \quad \text{with} \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0, \sum_i \alpha_i = 2, \tag{4}$$

where we assume again a linearly separable training sample. In [8,9] specific algorithms have been proposed for NPP that originate in the more classical Gilbert–Schlesinger–Kozinec (GSK; [10,11]) and Mitchell–Demyanov–Malozemov (MDM; [12]) algorithms to find the minimum norm vector of a convex set. While the GSK algorithm can be very slow, the MDM algorithm and some improvements (see [8]) are quite efficient.

While, as mentioned before, decomposition and geometric algorithms are usually discussed as independent procedures, we shall give in section 2 a new derivation of the MDM algorithm and show that for linearly separable problems, it

essentially coincides with a slight variant of SMO in which we require that both updating vectors belong to the same class. Although SVM algorithms for linearly separable problems extend immediately to non separable ones if square penalties $C\sum \xi_i^2$ are applied to margin slacks ξ_i [13], a different set up has to be pursued if linear penalties $C\sum \xi_i$ are considered. For SVM training this implies a restriction $\alpha_i \leq C$ for the multipliers α_i while NPP has to be solved over the so–called μ–Reduced Convex Hulls, where an extra restriction $\alpha_i \leq \mu$ has to be added to those in (4). It is well known that both problems are equivalent [7], but in the Appendix we will give a new, short proof of this fact. In section 3 we will extend to these settings the equivalence between SMO and MDM already proved for linearly separable problems in section 2. We will briefly compare numerically the performance of basic versions of the SMO and MDM algorithms in section 4 and show that, for square penalties, the final models they arrive at are essentially the same, as they have similar test accuracies and numbers of support vectors. SMO, however, needs less iterations than MDM, something to be expected, as it has to meet less restrictions when iteratively looking for maximum gains. The comparison for linear penalties is somewhat more involved, but the faster convergence of SMO still holds. A brief discussion ends the paper.

2 The SMO–MDM Equivalence for Linearly Separable Problems

2.1 Keerthi et al.'s Modification 2

Writing $F_i^o = W^o \cdot X_i - y_i$, the KKT conditions (3) at the optimal W^o, b^o can be expressed as

$$y_i(F_i^o + b^o) = 0 \ \text{ if } \ \alpha_i^o > 0, \ \ y_i(F_i^o + b^o) \geq 0 \ \text{ if } \ \alpha_i^o = 0. \tag{5}$$

Thus, if we define first the index sets $I^+ = \{i : y_i = 1\}$, $I^- = \{i : y_i = -1\}$ and then $I_{nSV} = \{i : \alpha_i = 0\}$, $I_{SV} = \{i : \alpha_i > 0\}$ (I_0 in the notation of [5]), $I_{nSV}^+ = I^+ \bigcap I_{nSV}$ (I_1 in Keerthi's notation), $I_{nSV}^- = I^- \bigcap I_{nSV}$ (I_4 in Keerthi's notation), the preceding conditions can be written as

$$F_i^o + b^o \geq 0 \ \text{ for } \ i \in I_{SV} \bigcup I_{nSV}^+, \ \ F_i^o + b^o \leq 0 \ \text{ for } \ i \in I_{SV} \bigcup I_{nSV}^-.$$

In particular, we will have $F_i^o \geq -b^o$ for $i \in I_{SV} \bigcup I_{nSV}^+$ and $-b^o \geq F_j^o$ for $j \in I_{SV} \bigcup I_{nSV}^-$. Thus, if we write $F_i = W \cdot X_i - y_i$ and define

$$b_{low} = \max\{F_j : j \in I_{SV} \bigcup I_{nSV}^-\}, \quad b_{up} = \min\{F_i : i \in I_{SV} \bigcup I_{nSV}^+\},$$

we must have $b_{low} \leq -b^o \leq b_{up}$ at the optimum. In practice one has to relax these conditions to $b_{low} - \epsilon/2 \leq -b^o \leq b_{up} + \epsilon/2$ for some $\epsilon > 0$. These observations

motivate Keerthi et al.'s Modification 2 in [5]. More precisely, they define at each step two indices

$$\begin{aligned} i_{low} &= \arg\max \ \{F_j : j \in I_{SV} \bigcup I^-_{nSV}\}, \\ i_{up} &= \arg\min \ \{F_i : i \in I_{SV} \bigcup I^+_{nSV}\}, \end{aligned} \tag{6}$$

and propose to take $i_2 = i_{low}$ and $i_1 = i_{up}$ in SMO. We then have $b_{low} = F_{i_{low}}$, $b_{up} = F_{i_{up}}$ and training will continue while $b_{low} > b_{up} + \epsilon$ or, in other words, while the i_2, i_1 indices violate the KKT conditions. As the experiments reported in [5] illustrate, these choices can significantly speed up Platt's original algorithm.

2.2 An Alternative Motivation for Choosing i_2 and i_1

Keerthi's heuristics are motivated by an attempt to simplify Platt's original ones but we will show next how they also arise if we try to choose directly the updating indices i_2, i_1 so that they maximize the gain in the dual cost function $W(\alpha)$ (see also the Appendix A in [6] for another way to arrive at these selections). Notice first that for any such pair (i_2, i_1) the new multipliers α' to be considered are $\alpha'_{i_1} = \alpha_{i_1} + \delta_{i_1}$, $\alpha'_{i_2} = \alpha_{i_2} + \delta_{i_2}$ while $\alpha'_j = \alpha_j$ for all others. The new W' has thus the form $W' = W + \delta_{i_1} y_{i_1} X_{i_1} + \delta_{i_2} y_{i_2} X_{i_2}$. Taking into account the restriction $\sum_i \alpha_i y_i = 0$, we must have $y_{i_1}\delta_{i_1} + y_{i_2}\delta_{i_2} = 0$ and, therefore, $\delta_{i_1} = -y_{i_1} y_{i_2} \delta_{i_2}$ and

$$W' = W + \delta_{i_2} y_{i_2} (X_{i_2} - X_{i_1}) = W + \delta_{i_2} y_{i_2} Z_{i_2,i_1},$$

where $Z_{j,k} = X_j - X_k$. Thus, $W(\alpha') = \frac{1}{2}\|W'\|^2 - \sum \alpha'_i$ is just a function $\Phi(\delta_{i_2})$ of δ_{i_2}, and we have

$$\begin{aligned} \Phi(\delta_{i_2}) &= \frac{1}{2}\|W\|^2 + \delta_{i_2} y_{i_2} W \cdot Z_{i_2,i_1} + \frac{\delta^2_{i_2}}{2}\|Z_{i_2,i_1}\|^2 - \sum \alpha_i - \delta_{i_1} - \delta_{i_2} \\ &= W(\alpha) + \delta_{i_2} y_{i_2} W \cdot Z_{i_2,i_1} + \frac{\delta^2_{i_2}}{2}\|Z_{i_2,i_1}\|^2 - \delta_{i_2} y^2_{i_2} + y_{i_1} y_{i_2} \delta_{i_2} \\ &= W(\alpha) + \delta_{i_2} y_{i_2} \left(W \cdot Z_{i_2,i_1} - (y_{i_2} - y_{i_1})\right) + \frac{\delta^2_{i_2}}{2}\|Z_{i_2,i_1}\|^2. \end{aligned} \tag{7}$$

Solving $\Phi'(\delta^*_{i_2}) = 0$ to obtain the optimal $\delta^*_{i_2}$ yields

$$\delta^*_{i_2} = -\frac{y_{i_2}\left(W \cdot Z_{i_2,i_1} - (y_{i_2} - y_{i_1})\right)}{\|Z_{i_2,i_1}\|^2} = -y_{i_2}\frac{\Delta}{\|Z_{i_2,i_1}\|^2}, \tag{8}$$

where $\Delta = W \cdot Z_{i_2,i_1} - (y_{i_2} - y_{i_1})$, and, in turn, $\delta^*_{i_1} = -y_{i_1} y_{i_2} \delta^*_{i_2} = y_{i_1}\frac{\Delta}{\|Z_{i_2,i_1}\|^2}$. Moreover, we have

$$\Phi(\delta^*_{i_2}) = W(\alpha) - \frac{1}{2}\frac{\left[y_{i_2}\left(W \cdot Z_{i_2,i_1} - (y_{i_2} - y_{i_1})\right)\right]^2}{\|Z_{i_2,i_1}\|^2} = W(\alpha) - \frac{1}{2}\frac{\Delta^2}{\|Z_{i_2,i_1}\|^2}.$$

Now, to maximize the decrease in $W(\alpha')$ we should choose (i_2, i_1) so that

$$(i_2, i_1) = \arg\max_{i,j} \left\{ \frac{(W \cdot Z_{i,j} - (y_i - y_j))^2}{\|Z_{i,j}\|^2} \right\}.$$

Such a choice of i_2, i_1 is sometimes called a second order working set selection [14]. If we simply ignore the $\|Z_{i,j}\|^2$ denominator, we can choose instead

$$(i_2, i_1) = \arg\max_{i,j} \{|W \cdot Z_{i,j} - (y_i - y_j)|\}. \tag{9}$$

It is clear that the maximum in (9) is attained at

$$\max_i \{W \cdot X_i - y_i\} - \min_j \{W \cdot X_j - y_j\},$$

which tells us to choose in principle (i_2, i_1) as

$$i_2 = \arg\max_j \{W \cdot X_j - y_j\}, \;\; i_1 = \arg\min_i \{W \cdot X_i - y_i\}. \tag{10}$$

These choices imply $\Delta \geq 0$ and we note in passing that there is a gain in $W(\alpha)$ whenever $\Delta > 0$ or, stated equivalently, whenever there is a violating pair; this gives a new and simple derivation of a well known result of Hush and Scovel (see Theorem 3 in [15]). Now, notice that if $y_{i_2} = 1$, $\delta_{i_2} < 0$ and, hence, we must have $\alpha_{i_2} > 0$. On the other hand, if $y_{i_1} = -1$, $\delta_{i_1} < 0$ and, hence, we must have $\alpha_{i_1} > 0$. As a consequence, we must refine our previous choices of i_2 and i_1 in (10) to

$$i_1 = \arg\min_i \{F_i : i \in I^+ \bigcup I^-_{SV}\}, \;\; i_2 = \arg\max_j \{F_i : i \in I^- \bigcup I^+_{SV}\}. \tag{11}$$

with $I^{\pm}_{SV} = I^{\pm} \bigcap I_{SV}$ and $F_i = W \cdot X_j - y_j$ again. Now it can be easily seen that $I^+ \bigcup I^-_{SV} = I_{SV} \bigcup I^+_{nSV}$ and, similarly, $I^- \bigcup I^+_{SV} = I_{SV} \bigcup I^-_{nSV}$. It is thus clear that these are the same selections done in Modification 2 of [5] as given in (6).

2.3 Solving NPP a la SMO

As discussed in section 1 there are several procedures for the NPP problem that have their origin in the MDM algorithm. In its original formulation as a minimum norm problem, the MDM algorithm selects at each step updating indices $i_2 = \arg\min_j \{W \cdot X_j\}$, $i_1 = \arg\max_i \{W \cdot X_i : \alpha_i > 0\}$. While the algorithm's objective is to update the current weight W with the one in the line segment between W and $W + \alpha_{i_2}(X_{i_2} - X_{i_1})$ with minimum norm, it is clear that the i_2 and i_1 choices also maximize $\Delta^2 = (W \cdot (X_i - X_j))^2$ (the condition $\alpha_i > 0$ for i_1 candidates is needed, as the W update will decrease α_{i_1}). While the approach in [8] to NPP is closer to the original MDM one as given in [12], the one in [9] does in fact try to maximize Δ^2.

In any case the above index choices are clearly related to the previous discussion for SMO and their minimization of Δ suggests to solve NPP as just done in the preceding section, that is, to work at each step with just two multipliers α_{i_1} and α_{i_2} and update a given $W = \sum \alpha_i y_i X_i$ to another one of the form $W' = W + \delta_{i_1} y_{i_1} X_{i_1} + \delta_{i_2} y_{i_2} X_{i_2}$ so that the minimization in the norm $\|W'\|^2$ is largest. The restrictions in (4) imply $2 = \sum \alpha'_i = \sum \alpha_i + \delta_{i_1} + \delta_{i_2} = 2 + \delta_{i_1} + \delta_{i_2}$ and $0 = \sum y_i \alpha'_i = \sum y_i \alpha_i + y_{i_1}\delta_{i_1} + y_{i_2}\delta_{i_2} = y_{i_1}\delta_{i_1} + y_{i_2}\delta_{i_2}$. The second one implies that $y_{i_1}\delta_{i_1} = -y_{i_2}\delta_{i_2}$ and, since the first one gives $\delta_{i_1} = -\delta_{i_2}$, we must also have $y_{i_1} = y_{i_2}$. As a consequence, $W' = W + \delta_{i_2} y_{i_2} (X_{i_2} - X_{i_1}) = W + \delta_{i_2} y_{i_2} Z_{i_2,i_1}$, where again $Z_{i,j} = X_i - X_j$; thus, $\|W'\|^2$ is a function of δ_{i_2} and we have

$$\Phi(\delta_{i_2}) = \|W'\|^2 = \|W\|^2 + 2\delta_{i_2} y_{i_2} W \cdot Z_{i_2,i_1} + \delta_{i_2}^2 \|Z_{i_2,i_1}\|^2.$$

As done before, solving $\Phi'(\delta^*_{i_2}) = 0$ gives

$$\delta^*_{i_2} = -y_{i_2} \frac{\Delta}{\|Z_{i_2,i_1}\|^2}, \quad \delta^*_{i_1} = y_{i_2} \frac{\Delta}{\|Z_{i_2,i_1}\|^2},$$

where now $\Delta = W \cdot Z_{i_2,i_1}$ and, in turn,

$$\Phi(\delta^*_{i_2}) = \|W\|^2 - \frac{\Delta^2}{\|Z_{i_2,i_1}\|^2}. \tag{12}$$

Thus, just as before, if we ignore the $\|Z_{i_2,i_1}\|^2$ denominator, we can maximize the gain in Φ by selecting i_1 and i_2 so that Δ is maximized. We do so setting first

$$\begin{aligned} i_2^+ &= \arg\max_i \{W \cdot X_i : y_i = 1\}, \quad i_1^+ = \arg\min_j \{W \cdot X_j : y_j = 1\}, \\ i_2^- &= \arg\max_i \{W \cdot X_i : y_i = -1\}, \quad i_1^- = \arg\min_j \{W \cdot X_j : y_j = -1\}, \end{aligned} \tag{13}$$

and deciding next which one of the pairs $(i_2^\pm, i_1^\pm)$ to choose, for which we compute

$$\Delta^+ = W \cdot \left(X_{i_2^+} - X_{i_1^+}\right), \quad \Delta^- = W \cdot \left(X_{i_2^-} - X_{i_1^-}\right),$$

(notice that both are positive) and take $i_2 = i_2^+$, $i_1 = i_1^+$ if $\Delta^+ > \Delta^-$ and $i_2 = i_2^-$, $i_1 = i_1^-$ otherwise. We observe that the corresponding index choices in the extension of MDM to NPP are

$$\begin{aligned} i_2^+ &= \arg\max_i \{W \cdot (X_i - W_-) : y_i = 1\}, \\ i_1^+ &= \arg\min_j \{W \cdot (X_j - W_-) : y_j = 1\}, \\ i_2^- &= \arg\max_i \{W \cdot (X_i - W_+) : y_i = -1\}, \\ i_1^- &= \arg\min_j \{W \cdot (X_j - W_+) : y_j = -1\}, \end{aligned}$$

which are obviously equivalent to the previous ones.

In any case, and just as it was done for SMO, we must make sure that the updated coefficients remain positive. Just as before we have $\Delta^\pm > 0$. Thus, if $y_{i_2} = 1$, $\delta_{i_2^+} < 0$ and, hence, we must have $\alpha_{i_2^+} > 0$. On the other hand, if

$y_{i_1} = -1$, $\delta_{i_1^-} < 0$ and, hence, we must have $\alpha_{i_1^-} > 0$. As a consequence, we refine our previous choices of i_2^+ and i_1^- in (13) to

$$i_1^- = \arg\min_i\{W \cdot X_i : i \in I_{SV}^-\}, \;\; i_2^+ = \arg\max_j\{W \cdot X_i : i \in I_{SV}^+\}. \quad (14)$$

As we show next, these choices coincide with those made in a slight variant of SMO.

2.4 Enforcing $y_{i_1} = y_{i_2}$ in SMO

Although in standard SMO the y_{i_1} and y_{i_2} values do not have to be equal, let us discuss SMO's formulation when at each iteration we force $y_{i_1} = y_{i_2}$ (the use of updates where all patterns used belong to the same class has also been proposed for ν–SV training [16]). We then have $\delta_{i_1} = -y_{i_1}y_{i_2}\delta_{i_2} = -\delta_{i_2}$ and $W' = W + \delta_{i_2}y_{i_2}X_{i_2} - \delta_{i_2}y_{i_2}X_{i_1} = W + \delta_{i_2}y_{i_2}Z_{i_2,i_1}$. Furthermore, (7) becomes now

$$\Phi(\delta_{i_2}) = W(\alpha') = W(\alpha) + y_{i_2}W \cdot Z_{i_2,i_1}\delta_{i_2} + \frac{\delta_{i_2}^2}{2}\|Z_{i_2,i_1}\|^2, \quad (15)$$

equation (8) for the optimum $\delta_{i_2}^*$ becomes

$$\delta_{i_2}^* = -y_{i_2}\frac{W \cdot Z_{i_2,i_1}}{\|Z_{i_2,i_1}\|^2} = -y_{i_2}\frac{\Delta}{\|Z_{i_2,i_1}\|^2},$$

where here $\Delta = W \cdot Z_{i_2,i_1}$, and, again, we have

$$\Phi(\delta_{i_2}^*) = W(\alpha) - \frac{1}{2}\frac{\Delta^2}{\|Z_{i_2,i_1}\|^2},$$

which has the same form that (12). Ignoring once more the denominator $\|Z_{i_2,i_1}\|^2$, this also suggests to take i_2, i_1 so as to maximize $|\Delta|$, which leads to the same index choices as done for MDM in the previous section.

Moreover, enforcing $y_{i_2} = y_{i_1}$ implies that $\delta_{i_1} = -\delta_{i_2}$ and also that, after initialization, the multipliers' sum $\sum \alpha_i$ remains constant at each iteration. Thus, in this setting, minimizing the dual criterion $W(\alpha) = \|W\|^2/2 - \sum \alpha_i$ reduces to minimize just $\|W\|^2$ and if the α_i are initialized so that $\sum \alpha_i = 2$, the problem that this SMO variant solves coincides with NPP. Moreover, since the updating indices' choices are the same in both cases, we can conclude that after a proper initialization, enforcing $y_{i_2} = y_{i_1}$ in SMO is equivalent to using MDM to solve NPP.

3 SMO and MDM for Non–linearly Separable Problems

In the preceding discussion we have assumed that the original sample classes were linearly separable. This assumption must be relaxed in practice allowing

for margin slacks that are penalized using either a linear or a quadratic cost function. The theory for the linearly separable case extends easily to the quadratic cost setting [13], but for a linear penalty we want to solve now the quadratic minimization problem

$$\min_{W,b,\xi} \frac{1}{2}\|W\|^2 + C\sum_i \xi_i, \tag{16}$$

subject to the linear restrictions $y_i(W \cdot X_i + b) + \xi_i \geq 1, i = 1, \ldots, N$. Its Wolfe dual is now

$$W(\alpha) = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_i \alpha_i, \tag{17}$$

where $0 \leq \alpha_i \leq C$, $\sum_i \alpha_i y_i = 0$. If NPP is to be considered, the alternative to (16) would be to consider it for the so–called μ–Reduced Convex Hulls, defined as $C_\mu(\mathcal{S}_\pm) = \{\sum \alpha_i X_i : X_i \in \mathcal{S}_\pm, \sum \alpha_i = 1, 0 \leq \alpha_i \leq \mu\}$. We shall refer to this new problem as RCH_μ–NPP (see [7] for more details).

Considering first SMO, the only difference with respect the discussion in section 2 is the restriction $\alpha_i \leq C$, which forces the δ_i increments to be positive only when $\alpha_i < C$. Thus, if $y_{i_2} = -1$ we must have $\alpha_{i_2} < C$ and if $y_{i_1} = 1$ we must have $\alpha_{i_1} < C$. As a consequence, in the non–linearly separable setting we must refine the i_2 and i_1 choices in (11) to

$$i_1 = \arg\min_i\{F_i : i \in I^+_{nB_C} \bigcup I^-_{SV}\}, \;\; i_2 = \arg\max_j\{F_i : i \in I^-_{nB_C} \bigcup I^+_{SV}\} \tag{18}$$

where now $I^\pm_{nB_C} = \{i : y_i = \pm 1, \alpha_i < C\}$. It can be easily checked that these are the same selections done in Modification 2 of [5]. Turning our attention to the MDM algorithm for RCH_μ–NPP, the situation is quite similar to the one just discussed for SMO, as we have to make sure that when $\alpha = \mu$, decrementing α is then the only option. As a consequence, we must now refine our previous choices of i_2^- and i_1^+ in (13) to

$$i_1^+ = \arg\min_i\{W \cdot X_i : i \in I^+_{nB_\mu}\}, \;\; i_2^- = \arg\max_j\{W \cdot X_i : i \in I^-_{nB_\mu}\}, \tag{19}$$

where now $I^\pm_{nB_\mu} = \{i : y_i = \pm 1, \alpha_i < \mu\}$. Arguing as before, initializing the α_i and scaling C adequately, enforcing $y_{i_1} = y_{i_2}$ results in SMO solving RCH_μ–NPP. We finally note that MDM–type algorithms for RCH_μ–NPP have been recently proposed [17] but they are conceptually more involved and computationally costlier than our just explained proposal.

4 Numerical Experiments

We shall compare the performance of the most basic versions of the SMO and NPP algorithms over 10 of the datasets provided in G. Rätsch's Benchmark Repository [18]. We employed the same experimental set–up described in the data site; in particular we used the provided 100 partitions (with about 40% training and 60% test patterns) to compute the test accuracies and the number of final SVs and training iterations, as well as the corresponding standard

Table 1. Average test accuracies, number of support vectors and number of iterations given by the CH-MDM and 2-SMO algorithms, with $\epsilon = 10^{-8}$

	Test err.		# SVs		# iters.	
Dataset	SMO	MDM	SMO	MDM	SMO	MDM
Titanic	22.8±1.2	22.8±1.2	150.0±0.0	150.0±0.0	363.1±20.2	402.1±16.7
Heart	15.7±3.2	15.7±3.2	163.3±2.4	163.3±2.4	338.9±13.0	399.1±14.9
Diabetes	23.1±1.6	23.1±1.6	412.7±7.7	412.7±7.7	1565.7±45.1	1666.1±38.2
Cancer	26.5±4.8	26.5±4.8	179.3±5.9	179.3±5.9	1140.6±52.0	1226.2±54.3
Thyroid	4.3±1.9	4.3±1.9	87.4±3.0	87.4±3.0	226.7±10.3	243.6±9.9
Flare	33.5±1.7	33.5±1.7	664.5±0.7	664.5±0.7	1398.4±53.1	1652.6±51.2
Splice	10.6±0.7	10.6±0.7	728.6±12.7	728.7±12.8	4402.3±635.8	4835.6±667.7
Image	2.9±0.5	2.9±0.5	215.5±11.5	215.3±11.5	34447.9±2117.9	39560.6±3203.9
German	23.56±2.0	23.5±2.0	590.22±12.4	590.0±12.4	19099.1±971.7	20441.6±711.3
Banana	10.4±0.4	10.4±0.4	230.9±14.0	230.9±14.0	1313.0±83.1	1364.6±91.3

deviations. Before giving the concrete results, we briefly comment on some implementation details. First, and as usual, all algorithms only involve dot products, that can be replaced through an appropriate positive definite kernel K. Next, we notice that many improvements have been made to the basic SMO and MDM algorithms, such as Platt's type I and type II updates or support vector shrinking. We will not consider them in our experiments as they are more or less applicable to both procedures and likely to have similar effects. We also point out that we must make sure that, for instance, $0 \leq \alpha_i + \delta_i \leq C$ for linear penalties' SMO and that $0 \leq \alpha_i + \delta_i \leq \mu$ for RCH_μ–NPP. This means that the δ_i will have to be adequately bounded from above and below as necessary. Finally, the b^o and b^* bias values are also different in SMO and MDM. For SMO we will take, as usual,

$$b^o = \frac{1}{N_{SV}} \sum_{i \in I_{SV}} (y_i - W^o \cdot X_i) = \frac{1}{N_{SV}} \left(\sum_{i \in I_{SV}} y_i - \sum_{i,j \in I_{SV}} \alpha_j y_j X_j \cdot X_i \right)$$
$$= \frac{1}{N_{SV}} \left(\sum_{i \in I_{SV}} y_i - \sum_{i,j \in I_{SV}} \alpha_j y_j K'(x_j, x_i) \right),$$

with N_{SV} the number of support vectors. For quadratic penalties we will use $K'(x_j, x_i) = K(x_j, x_i) + \delta_{ij}/C$ as the square penalty–adjusted version of a standard positive definite kernel K while we just take $K' = K$ for linear penalties. A simple geometric reasoning implies that the MDM bias will be

$$b^* = -W^* \cdot \frac{\left(W_+^* + W_-^*\right)}{2} = -\frac{1}{2} \sum_{i,j \in I_{SV}} \alpha_i \alpha_j y_i K'(x_i, x_j).$$

Table 2. Average test accuracies, number of support vectors and number of iterations given by the RCH-MDM and 1-SMO algorithms, with $\epsilon = 10^{-8}$. For test errors a * stands for a statistically significant difference in a Wilcoxon rank test.

Dataset	Test err.		# SVs		# iters.	
	SMO	MDM	SMO	MDM	SMO	MDM
Titanic	24.1±8.0	24.0±7.4	67.2±11.3	113.9±8.8	156.6±32.7	164.2±28.7
Heart	15.8±3.2	16.0±3.1	82.4±5.4	82.4±5.4	217.1±63.6	306.2±59.6
Diabetes	23.4±1.6*	23.7±1.8	264.9±7.2	264.8±7.2	464.7±91.5	741.4±81.9
Cancer	27.3±5.9*	28.9±4.8	113.6±6.5	113.8±6.3	1705.3±897.4	3352.7±3841.2
Thyroid	4.4±2.1	4.2±2.0	25.3±5.7	25.3±5.7	328.2±124.4	398.9±117.1
Flare	32.7±1.6*	32.8±1.6	477.1±12.2	508.9±9.9	862.2±391.0	1401.4±881.5
Splice	10.7±0.6	10.8±0.6	620.2±14.2	629.2±13.6	2569.6±177.4	2797.4±290.5
Image	3.0±0.4	3.0±0.5	167.6±9.2	172.0±8.8	47972.5±11219.0	56169.9±10309.4
German	23.62±2.1*	24.0±2.1	407.6±10.7	407.7±10.8	1660.6±149.2	1884.5±144.8
Banana	11.5±0.6*	11.6±0.6	89.6±10.1	89.5±10.0	38236.0±14307.7	43449.9±25339.9

4.1 Quadratic Penalties

It is well known that SVM algorithms for linearly separable problems extends immediately to non separable ones if square penalties $C\sum \xi_i^2$ are applied to margin slacks ξ_i [13]. We shall use a common initialization for both SMO and MDM choosing a single vector from each class and setting $\alpha_{i_1} = \alpha_{i_2} = 1$. As mentioned in section 2, the usual SMO stopping condition is $b_{low} \leq b_{up} + \epsilon$; for the MDM algorithm one might use either $\Delta \leq \epsilon$ or also $\Delta \leq \epsilon\|W\|^2/2$. While these conditions look similar, the norms of the SMO and MDM W vectors involved are very different. Thus, in order to make more homogeneous performance comparisons, we will use in both cases a similar relative precision criterion, stopping SMO when $W(\alpha) - W(\alpha') \leq \epsilon W(\alpha)$ and MDM when $\|W\|^2 - \|W'\|^2 \leq \epsilon\|W\|^2$.

We will compare the performance of the basic SMO and MDM implementation over three values: the number of training iterations they need, the number of support vectors the final SVMs have and the test accuracies of the final models. We will do so for a relative $\epsilon = 10^{-8}$ precision and the results of each method are shown in table 1. In all cases we have used Gaussian kernels $\exp\left(-\|x\|^2/2\sigma^2\right)$ and optimal σ and C have been estimated by cross–validation. It can be seen in the table that SMO is faster, as it needs less iterations to achieve the desired precisions. This is quite natural, as it has greater freedom when choosing at each iteration the maximum gain multipliers. On the other hand, the final models obtained seem to be very similar, as they essentially have the same accuracies and support vector numbers; moreover, after the appropriate scaling, the corresponding optimal dual function values were essentially the same. A * superscript for the test errors indicates a significant difference in a Wilcoxon rank test at the 10% level; the final test error values are similar to those in [18].

4.2 Linear Penalties

While for square penalties SMO and MDM use the same C parameter, the situation for linear penalty SMO and RCH–MDM is more complex. In fact, and as shown in the Appendix, the relationship between the C and μ parameters is now $\mu = 2C/\rho^o$, with $\rho^o = \|W^o\|^2 + C\sum \xi_i^o$. Hence C and μ are not independent, and they should be chosen differently depending on which algorithm is to be used. In our experiments we have chosen for C the values proposed in [18] and once SMO finishes for each training–test pair, we have subsequently trained RCH–MDM using a μ value computed as just explained. Moreover, while the previous two vector initialization for SMO is still possible, this is not so for RCH–MDM and in this case we have chosen each sample barycenters as the initial $W_\pm$ vectors. All this makes final model comparisons somewhat less homogeneous than the square penalty ones, as shown in table 2, where now final accuracies are similar for both methods (but less so than in the square penalty case) and SMO models clearly have less support vectors. This last fact is due, however, to the different initializations used: if SMO is trained starting from the barycenters (not a good idea anyway), its final models have more SVs, implying that RCH–MDM is better at removing wrong initial SV choices (the algorithm is in some sense designed for that to be true). In any case, and for the initializations used, SMO is again faster than RCH–MDM.

5 Discussion

The SMO algorithm for SVM construction and, on the other hand, the geometrically inspired NPP solving algorithms such as extended MDM are usually discussed as different, independent methods. We have shown in this note that, however, these two methods are in fact very closely related, as they can be seen as maximum gain algorithms for working sets of 2 multipliers. More precisely, the extended MDM algorithm typically used to solve NPP essentially coincides with a restricted form of SMO in which the working set multipliers correspond to sample patterns in the same class. As we have numerically illustrated for quadratic penalties, the basic SMO and MDM algorithms seem to arrive at the same models when a moderately high precision is imposed in their final minima. However, SMO seems to be faster, something quite natural, as it has greater freedom when choosing at each iteration the maximum gain multipliers. While the linear penalty comparison is more involved, it seems clear that SMO is again faster. Another contribution of the present work is a proposal of an MDM algorithm for RCH_μ–NPP considerably simpler than previous ones.

While this would seem to imply that there will not be great advantages from the consideration of geometric algorithms for SVM construction, we point out that the usual speed enhancements for SMO, such as shrinking, can also be applied to the MDM algorithm. On the other hand, there has been a considerable amount of work in efficient solutions of the Minimum Norm Problem (MNP) for convex sets, the question that lies at the heart of the MDM algorithm. Given the close relationship shown here between the SMO and MDM methods, it is

thus conceivable that insights gained for MNP algorithms can provide new ways of accelerating SMO and other algorithms derived from it.

References

1. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
2. Schölkopf, B., Smola, A.: Learning with kernels support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge (2002)
3. Platt, J.: Fast training of support vector machines using sequential minimal optimization. Advances in Kernel Methods - Support Vector Machines, 185–208 (1999)
4. Joachims, T.: Making large-scale support vector machine learning practical. Advances in Kernel Methods - Support Vector Machines, 169–184 (1999)
5. Keerthi, S., Shevade, S., Bhattacharyya, C., Murthy, K.: Improvements to platt's smo algorithm for SVM classifier design. Neural Computation 13(3), 637–649 (2001)
6. Fan, R., Chen, P.H., Lin, C.: Working set selection using second order information for training support vector machines. Journal of Machine Learning Research 6, 1889–1918 (2005)
7. Bennett, K., Bredensteiner, E.: Duality and geometry in svm classifiers. In: Proc. 17th Int. Conf. Machine Learning, pp. 57–64 (2000)
8. Keerthi, S., Shevade, S., Bhattacharyya, C., Murthy, K.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Transactions on Neural Networks 11(1), 124–136 (2000)
9. Franc, V.: Simple solvers for large quadratic programming tasks. In: Kropatsch, W., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 75–84. Springer, Heidelberg (2005)
10. Gilbert, E.: Minimizing the quadratic form on a convex set. SIAM J. Contr. 4, 61–79 (1966)
11. Franc, V., Hlaváč, V.: An iterative algorithm learning the maximal margin classiffier. Pattern Recognition 36, 1985–1996 (2003)
12. Mitchell, B., Dem'yanov, V., Malozemov, V.: Finding the point of a polyhedron closest to the origin. SIAM J. Contr. 12, 19–26 (1974)
13. Shawe-Taylor, J., Cristianini, N.: On the generalisation of soft margin algorithms. IEEE Transactions on Information Theory 48(10), 2711–2735 (2002)
14. Glasmachers, T., Igel, C.: Second order smo improves svm online and active learning. Neural Computation 20(2), 374–382 (2008)
15. Hush, D., Scovel, C.: Polynomial-time decomposition algorithms for support vector machines. Machine Learning 51(1), 51–71 (2003)
16. Chang, C.C., Lin, C.J.: Training ν-support vector classifiers: Theory and algorithms. Neural Computation 13(9), 2119–2147 (2001)
17. Tao, Q., Wu, G.W., Wang, J.: A general soft method for learning svm classifiers with l1-norm penalty. Pattern Recogn. 41(3), 939–948 (2008)
18. Rätsch, G.: Benchmark repository, `ida.first.fraunhofer.de/projects/bench/benchmarks.htm`

Appendix: The Equivalence between SVM and NPP

We will consider in what follows the linear penalty case, the arguments for the penalty–free situation being similar and simpler. It is well known that the KKT

conditions for SVM imply that at the optimum $W^o = \sum \alpha_i^o y_i X_i$ we have $\alpha_i^o = C$ if $\xi_i^o > 0$, and also

$$\alpha_i^o \left(y_i \left(W^o \cdot X_i + b^o \right) - 1 + \xi_i^o \right) = 0,$$

that is, $\alpha_i^o = \alpha_i^o \left(y_i \left(W^o \cdot X_i + b^o \right) + \xi_i^o \right)$. Summing over i gives

$$\begin{aligned} \sum \alpha_i^o &= \sum \alpha_i^o y_i W^o \cdot X_i + b^o \sum \alpha_i^o y_i + \sum \alpha_i^o \xi_i^o \\ &= \|W^o\|^2 + C \sum \xi_i^o, \end{aligned}$$

since $\sum \alpha_i^o y_i = 0$. If we write $\rho^o = \|W^o\|^2 + C \sum \xi_i^o$ and define now

$$W' = \frac{2}{\rho^o} W^o = \sum_i \frac{2\alpha_i^o}{\rho^o} y_i X_i = \sum_i \alpha_i' y_i X_i,$$

with $\alpha_i' = 2\alpha_i^o / \rho^o$, we shall show that W' coincides with the optimal solution W^* to the RCH_μ problem, with $\mu = 2C/\rho^o$ To prove it, notice first that $\sum_i \alpha_i' y_i = 0$, $\sum_i \alpha_i' = 2$ and $\alpha_i' \leq \mu$. Thus, W' is a feasible solution of the RCH_μ problem. For any other RCH_μ feasible $W = \sum \alpha_i y_i X_i$, we have

$$\begin{aligned} W \cdot W' &= \sum_i \alpha_i y_i W' \cdot X_i = \frac{2}{\rho^o} \sum_i \alpha_i y_i W^o \cdot X_i = \frac{2}{\rho^o} \sum_i \alpha_i y_i \left(W^o \cdot X_i + b^o \right) \\ &\geq \frac{2}{\rho^o} \sum_i \alpha_i \left(1 - \xi_i^o \right) \geq \frac{2}{\rho^o} \left(\sum_i \alpha_i - \frac{2C}{\rho^o} \sum_i \xi_i^o \right) \\ &= \frac{2}{\rho^o} \left(2 - \frac{2C}{\rho^o} \sum_i \xi_i^o \right) = \frac{4}{(\rho^o)^2} \left(\rho^o - C \sum_i \xi_i^o \right) \\ &= \frac{4}{(\rho^o)^2} \|W^o\|^2 = \|W'\|^2. \end{aligned}$$

By Schwarz's inequality this implies $\|W\| \geq \|W'\|$ and, in particular $\|W^*\| \geq \|W'\|$, which by the uniqueness of the NPP solution implies $W' = W^*$.

Nearest Neighbour Classification with Monotonicity Constraints

Wouter Duivesteijn and Ad Feelders

Utrecht University, Department of Information and Computing Sciences,
P.O. Box 80089, 3508TB Utrecht, The Netherlands
wduivest@cs.uu.nl, ad@cs.uu.nl

Abstract. In many application areas of machine learning, prior knowledge concerning the monotonicity of relations between the response variable and predictor variables is readily available. Monotonicity may also be an important model requirement with a view toward explaining and justifying decisions, such as acceptance/rejection decisions. We propose a modified nearest neighbour algorithm for the construction of monotone classifiers from data. We start by making the training data monotone with as few label changes as possible. The relabeled data set can be viewed as a monotone classifier that has the lowest possible error-rate on the training data. The relabeled data is subsequently used as the training sample by a modified nearest neighbour algorithm. This modified nearest neighbour rule produces predictions that are guaranteed to satisfy the monotonicity constraints. Hence, it is much more likely to be accepted by the intended users. Our experiments show that monotone kNN often outperforms standard kNN in problems where the monotonicity constraints are applicable.

1 Introduction

Monotonicity of relations between a response variable and predictor variables is a form of prior knowledge that is available in many application areas of machine learning. For example, in house pricing, the price of a house typically increases with the lot size, and decreases with the distance to the city center. Other examples of monotonicity constraints can be found in medicine [8,25], finance [16], and law [18].

Monotonicity may also be an important model requirement with a view toward explaining and justifying decisions, such as acceptance/rejection decisions. Pazzani et al.[21], report on an application of rule induction algorithms to early detection of dementia, and prediction of mild mental retardation. They show that the rules learned with monotonicity constraints were significantly more acceptable to medical experts than rules learned without the monotonicity restrictions.

While human experts tend to feel uncomfortable expressing their knowledge and experience in terms of numeric assessments, they typically are able to state their knowledge in a semi-numerical or qualitative form with relative conviction and clarity, and with less cognitive effort [10]. Experts, for example, can often easily indicate which of two probabilities is smallest. In addition to requiring less cognitive effort, such relative judgements tend to be more reliable than direct numerical assessments [19].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 301–316, 2008.

Hence, monotonicity constraints occur frequently in machine learning problems and such constraints can be elicited from subject area experts with relative ease and reliability. This has motivated the development of learning algorithms that are able to enforce such constraints in a justified manner. Several machine learning techniques have been adapted to be able to handle monotonicity constraints in one form or another. Examples are: classification trees [7,12,22], neural networks [3,26], and Bayesian networks [1,13].

In this paper we present an algorithm for nonparametric monotone classification. Our approach consists of two steps. In the first step, the training data is made monotone by relabeling as few cases as possible. This relabeled data set may be viewed as the monotone classifier with the smallest error rate on the training data. In the second step, we use a modified nearest neighbour rule to predict the class labels of new data in such a way that the monotonicity constraints are satisfied.

The paper is organized as follows. In the next section, we establish some notation and definitions that will be used throughout the paper. In section 3, we discuss the problem of relabeling a non-monotone data set, and give an algorithm to make it monotone with as few label changes as possible. Subsequently, we present in section 4 a monotone variant of the k-nearest neighbour rule to predict the class labels of new data points. Related work on nonparametric monotone classification is discussed in section 5. In section 6 we present the results of experiments in which we compare the monotone nearest neighbour rule with standard nearest neigbour prediction. Finally, we draw conclusions in section 7.

2 Notation and Preliminaries

Let $\mathbf{X}$ denote the vector of predictors (attributes), which takes values $\mathbf{x}$ in a p-dimensional input space $\mathcal{X} = \times \mathcal{X}_i$, and let Y denote the class variable which takes values y in a one-dimensional space $\mathcal{Y}$. Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denote the set of observed data points in $\mathcal{X} \times \mathcal{Y}$. We also use the alternative representation $U = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, of n *distinct* points in $\mathcal{X} \times \mathcal{Y}$ together with a vector of weights $w_i = n(\mathbf{x}_i, y_i), \quad i = 1, \ldots, n$, where $n(\mathbf{x}_i, y_i)$ denotes the number of observations in D with $\mathbf{X} = \mathbf{x}_i$ and $Y = y_i$. Clearly, we have $N = \sum_{i=1}^{n} w_i$. Furthermore, we assume a partial order on $\mathcal{X}$ and a total order on $\mathcal{Y} = \{1, 2, \ldots, c\}$, where c is the number of class labels. Typically, the partial order on $\mathcal{X}$ is the product order induced by total orders on $\mathcal{X}_i$, that is

$$\mathbf{x} \leq \mathbf{x}' \Leftrightarrow x_i \leq x'_i \qquad \forall i = 1, \ldots, p,$$

but at no point do we require this to be the case. The objective is to learn from data an allocation rule $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow f(\mathbf{x}) \leq f(\mathbf{x}'), \tag{1}$$

that is, a lower ordered input is not allowed to have a higher class label. A pair of points $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$ from U (or D) is called non-monotone if

$$\mathbf{x}_i \leq \mathbf{x}_j \text{ and } y_i > y_j \tag{2}$$

We define the *monotonicity violation graph* (MVG) to be the directed graph $G = (V, E)$, with $V = \{1, 2, \ldots, n\}$ and $(i, j) \in E$ if $\mathbf{x}_i \leq \mathbf{x}_j$ and $y_i > y_j$. We note that the monotonicity violation graph is the graph of a strict partial order, since it is

1. Anti-symmetric: $(i, j) \in E \Rightarrow (j, i) \notin E$.
2. Transitive: $(i, j) \in E$ and $(j, k) \in E \Rightarrow (i, k) \in E$.

These properties follow immediately from the order on the class labels. We associate with each node $i \in V$ the weight w_i. Finally, we define the downset $\downarrow_{(i,S)}$ and the upset $\uparrow_{(i,S)}$ for any $S \subseteq V$ and $i \in V$:

$$\downarrow_{(i,S)} = \{j \in S | \mathbf{x}_j \leq \mathbf{x}_i\} \text{ and } \uparrow_{(i,S)} = \{j \in S | \mathbf{x}_i \leq \mathbf{x}_j\}.$$

3 Relabeling Non-monotone Data

The first step in our approach is to relabel the training data in order to remove all monotonicity violations, using as few label changes as possible. The relabeled data set can be viewed as a monotone classifier that minimizes the error rate on the training data.

A subset of the vertices of a graph is an *independent set* if no two vertices in the subset are adjacent. As Rademaker et al. [23] observe, a maximum weight independent set in the monotonicity violation graph, corresponds to a maximum size monotone subset of the data. Relabeling the complement of the maximum independent set results in a monotone data set with as few label changes as possible; it is important to note that it is always possible to find a consistent relabeling. Although finding a maximum independent set in an arbitrary graph is known to be NP-hard [17], we make use of the fact that this is not the case for *comparability graphs* (the graph of a partial order). For such graphs, a maximum independent set corresponds to a maximum antichain in the corresponding partial order, and can be computed in $O(n^3)$ time by solving a minimum flow problem on a transportation network that is easily constructed from the comparability graph, see [20,14]. As we noted, the monotonicity violation graph is a comparability graph, so we have an $O(n^3)$ algorithm that minimizes

$$\sum_{i=1}^{N} I(y_i \neq f(\mathbf{x}_i)),$$

subject to

$$\mathbf{x}_i \leq \mathbf{x}_j \Rightarrow f(\mathbf{x}_i) \leq f(\mathbf{x}_j), \tag{3}$$

for an arbitrary partial order on $\mathcal{X}$, and for an arbitrary number of linearly ordered class labels.

We next describe the transformation of the monotonicity violation graph $G = (V, E)$ to the corresponding transportation network $G' = (V', E')$. Let V^- denote the set of nodes in V with non-zero degree. Because the monotonicity violation graph has weights associated with the vertices rather than the edges (as is assumed by standard network flow algorithms), we transform vertices to edges, by so-called vertex splitting:

$$V' = \bigcup_{i \in V^-} \{i_a, i_b\} \cup \{s, t\},$$

where s is the source, and t is the sink of the transportation network. The edge set E' contains edges (i_a, i_b) for all $i \in V^-$, and edges (i_b, j_a) for all $(i, j) \in E$. Furthermore, E' contains edges (s, i_a) for all minimal points $\mathbf{x}_i$, and edges (j_b, t) for all maximal points $\mathbf{x}_j$. The edges $(i_a, i_b) \in E'$ are assigned lower capacities w_i and upper capacities $+\infty$. All remaining edges of E' are assigned lower capacities of zero and upper capacities of $+\infty$. The problem of finding the maximum weight independent set in G can now be solved by finding the minimum flow value $\mathtt{f}_{val}^{min}$ in G'. Furthermore, by the min-flow max-cut theorem [15], $\mathtt{f}_{val}^{min}$ equals the maximum capacity of an s, t-cut (or maximum cut) in G', that is,

$$\mathtt{f}_{val}^{min} = \max_{S,T} \left[\sum_{\substack{(v,w) \in E' \\ v \in S, w \in T}} lc(v, w) - \sum_{\substack{(v,w) \in E' \\ v \in T, w \in S}} uc(v, w) \right],$$

where S, T is an s, t-cut of $G' = (V', E')$, that is, $V' = S \cup T$, $S \cap T = \varnothing$, $s \in S$, $t \in T$, and where $lc(v, w)$ and $uc(v, w)$ denote the lower and upper capacity of edge $(v, w) \in E'$.

Obviously, $\mathtt{f}_{val}^{min}$ must be positive, and therefore an optimal cut S, T contains no edges $(v, w) \in E'$ with $v \in T$ and $w \in S$, since $uc(v, w) = +\infty$ for each such edge. This implies that the set of vertices

$$A = \{i \in V \mid (i_a, i_b) \in E', i_a \in S, i_b \in T\} \tag{4}$$

corresponding to an optimal cut, is an antichain of G. To see this, suppose that A is not an antichain, that is, it contains comparable points i and j. Then, by the definition of A, we have i_a and $j_a \in S$, and i_b and $j_b \in T$. Since i and j are comparable, we have either $(i_b, j_a) \in E'$ or $(j_b, i_a) \in E'$ which means we would have an edge from T to S in E'. But this contradicts our observation that $\mathtt{f}_{val}^{min}$ must be positive.

Furthermore, since each antichain A in G induces an S, T cut in G' by putting

$$S = \{v \in V' \mid \text{there is a directed path in } G' \text{ from } v \text{ to } i_b \text{ for some } i \in A\}$$

we have that the minimum flow value in G' equals the maximum weight of an antichain in G [20].

Although in the worst case, the run time of the algorithm is cubic in the number of distinct observations, it may be quite fast in practice because of two reasons. First, all points that are not involved in any monotonicity violations can be disregarded, because they will never be relabeled. Since they are not connected to any other point in the MVG, they will belong to every maximum independent set. Hence the restriction to nodes in V^- in the transportation network. If the data generating process is indeed monotonic, and any monotonicity violation is caused by noise, then it is reasonable to assume that most points are not involved in a monotonicity violation. Secondly, we can apply a divide-and-conquer strategy by finding a maximum independent set for each

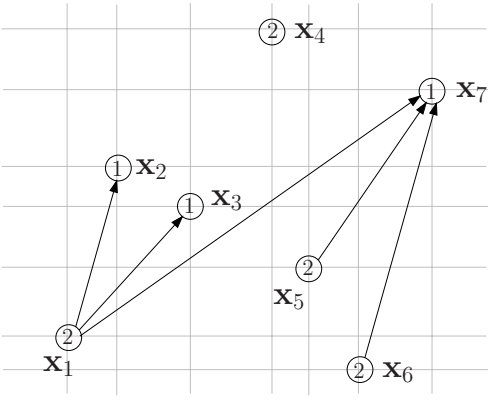


Fig. 1. Example Monotonicity Violation Graph

connected component of the MVG separately. The union of these sets will then be a maximum independent set for the complete graph.

The relabeling algorithm is summarized in Algorithm 1. It takes the training set and its monotonicity violation graph as inputs, and returns the relabeled training set. In line 6 the actual relabeling takes place. The function *select* picks a value from the interval of allowed class labels $[y_{\min}, y_{\max}]$; this interval always contains at least one element, since the set of points with index in M is always consistent. Which element it picks is arbitrary from the viewpoint of error-rate minimization.

Algorithm 1. relabel($U, G = (V, E)$)

1: $M \leftarrow$ maximum independent set(G)
2: $R \leftarrow V \setminus M$
3: **for all** $j \in R$ **do**
4: $\quad y_{\min} \leftarrow \max\{y_i | i \in \downarrow_{(j,M)}\}$
5: $\quad y_{\max} \leftarrow \min\{y_i | i \in \uparrow_{(j,M)}\}$
6: $\quad y_j \leftarrow$ select($y_{\min}$,$y_{\max}$)
7: $\quad M \leftarrow M \cup \{j\}$
8: **end for**
9: **return** U

As an example, consider the data set with monotonicity violation graph depicted in Figure 1. Here $\mathbf{x}_1, \ldots, \mathbf{x}_7$ are plotted as points in the plane, and their observed class labels are given inside the points. If both x_1 and x_2 are known to have a positive influence on y, then the appropriate ordering on the input points is given by

$$\mathbf{x} \leq \mathbf{x}' \Leftrightarrow x_1 \leq x_1' \wedge x_2 \leq x_2'$$

So, for example, we have $\mathbf{x}_1 \leq \mathbf{x}_2$, but $\mathbf{x}_2$ and $\mathbf{x}_3$ are incomparable points. The corresponding monotonicity violation graph is $G = (V, E)$ with $V = \{1, 2, 3, 4, 5, 6, 7\}$,

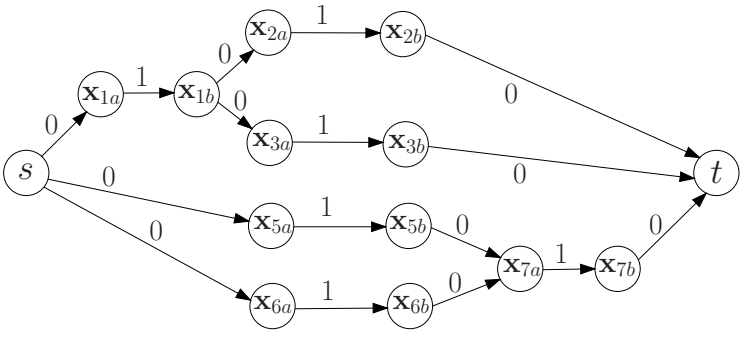


Fig. 2. Transportation network based on the Monotonicity Violation Graph in Figure 1

$E = \{(1,2),(1,3),(1,7),(5,7),(6,7)\}$, and $V^- = V \backslash \{4\}$. Figure 2 depicts the transportation network associated with G. Each of the points in V^- is represented by an edge with a lower capacity of one (all data points happen to be unique in this example), and an upper capacity of $+\infty$. The connections to the source and the sink, and the edges representing the monotonicity violations are assigned lower capacities of zero and upper capacities of $+\infty$. For the network flow in Figure 2, we find $\mathfrak{f}_{val}^{min} = 4$, that is, the weight of the maximum weight antichain A is 4. This set is obtained by the S,T-cut, where $S = \{s, \mathbf{x}_{1a}, \mathbf{x}_{1b}, \mathbf{x}_{2a}, \mathbf{x}_{3a}, \mathbf{x}_{5a}, \mathbf{x}_{6a}\}$, $T = V' \backslash S$. Using equation (4), we find $A = \{2,3,5,6\}$. Adding $\mathbf{x}_4$ gives $M = \{2,3,4,5,6\}$. Finally, the set complement to the maximum weight independent set is the set of points that need to be relabeled to get monotone data. Hence, we find that the set of points that need to be relabeled to make D monotone is $R = V \backslash M = \{1,7\}$.

Because the class label is binary, there is only one alternative label for each point, so relabeling is automatic. For binary classification problems, we can also accomodate different misclassification (relabeling) costs. Let $C(j,k)$ denote the cost of relabeling an example from class j to class k. Define weights

$$w_i = \begin{cases} n(\mathbf{x}_i, y_i)C(1,2) \text{ if } & y_i = 1 \\ n(\mathbf{x}_i, y_i)C(2,1) \text{ if } & y_i = 2 \end{cases}$$

We now obtain a minimum cost relabeling by finding a maximum weight independent set in the MVG, and relabeling its complement. To illustrate, consider the case where $C(1,2) = 1$ and $C(2,1) = 3$. Hence, all points in Figure 1 with class label 1 receive a weight of 1, and all points with class label 2 receive a weight of 3. The reader can verify that the maximum weight independent set is $M = \{1,4,5,6\}$ and hence $R = \{2,3,7\}$: it has become cheaper to relabel both 2 and 3, instead of relabeling 1. Unfortunately, this straightforward approach can not be extended to the non-binary case, because we then have more than one relabeling option, and we can therefore not associate a unique weight with each node.

The relabeled data set can be viewed as a monotone classifier that minimizes the error-rate on the training data. This classifier is however only defined on the observed data points. In case we have just a few discrete input variables, these might cover the

entire input space, but in general this will not be the case. Hence, the classifier has to be extended to the entire input space in such a way that the monotonicity constraints are satisfied, and the information contained in the observed data points is used to its full extent to classify new cases. This problem is discussed in the next section.

4 Monotone kNN

In order to satisfy the monotonicity restrictions, it is clear that the class label assigned to a new data point $\mathbf{x}_0$ is constrained to lie in the the interval $[y_{\min}, y_{\max}]$, where

$$y_{\min} = \max\{y|(\mathbf{x}, y) \in D \wedge \mathbf{x} \leq \mathbf{x}_0\},$$

and

$$y_{\max} = \min\{y|(\mathbf{x}, y) \in D \wedge \mathbf{x}_0 \leq \mathbf{x}\},$$

where D is the relabeled data set. The choice of a value from this interval is however free, and hence it makes sense to make further use of the observed data to guide this choice. We consider two variants on the standard nearest neighbour rule:

1. Take the k nearest neighbours of $\mathbf{x}_0$ from D and predict the label from $[y_{\min}, y_{\max}]$ that occurs most often among these k points. If none of the k labels are allowed, choose at random from $[y_{\min}, y_{\max}]$.
2. Take the k nearest neighbours of $\mathbf{x}_0$ from D with label in $[y_{\min}, y_{\max}]$ and predict the label by majority voting.

Variant 1 uses at most k neighbours in the majority voting, variant 2 always uses exactly k neighbours. The two variants are equivalent if the class label is binary, since in that case there is a choice of label only when both labels are allowed, but then both variants are the same as the standard nearest neighbour rule.

If we want predictions to be consistent among themselves as well (and not just with the training sample), then we should store the points with their predicted class labels to be used in subsequent predictions. It is clear that in this case, the order of arrival of points to be predicted makes a difference.

To visually illustrate the difference between standard nearest neighbour and monotone nearest neighbour, we consider a small example. Suppose the training data consists of the three points plotted in Figure 3. Next to each data point, its (x_1, x_2) coordinates and class label are given. The figure also gives the partitioning of the input space according to the 1-nearest neighbour rule, the so-called Voronoi diagram. It is clear that the resulting allocation rule is not monotone. In Figure 4 we have given the allocation rule for the *next prediction* of the monotone 1-nearest neighbour rule. Since all points smaller than (4,8) can not get a class label bigger than 1, the allocation rule has been adjusted accordingly. Note that this allocation rule is not monotone in general, but it is monotone with respect to the three points in the training sample. If predictions do not have to be monotone among themselves, then Figure 4 gives the monotone 1-nearest neighbour allocation rule. Otherwise, it may have to be updated after each prediction.

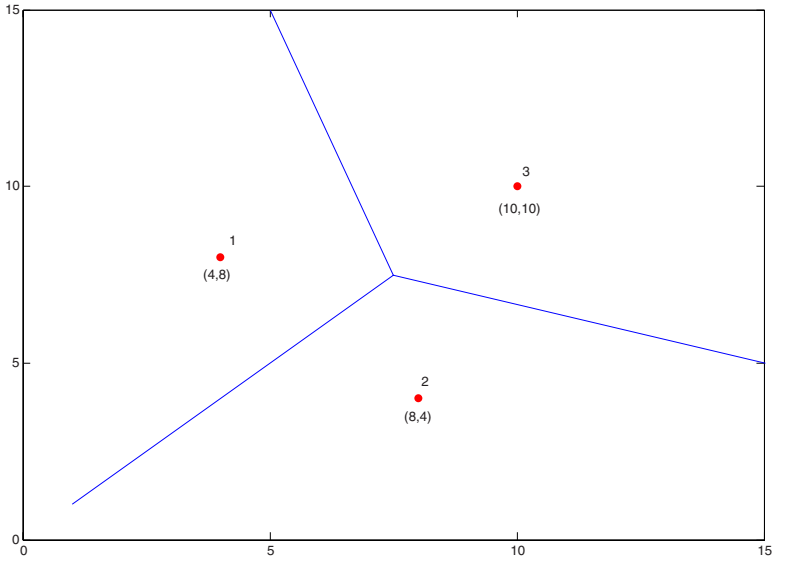


Fig. 3. Allocation rule of 1 nearest neighbour. For each data point its (x_1, x_2) coordinates and class label are given.

5 Related Work

As mentioned in the introduction, several machine learning methods have been adapted to incorporate monotonicity constraints. In this section, we restrict our attention to work that is relevant specifically to the *nonparametric* monotone classification problem that we are considering in this paper.

The earliest work in this area known to us is the Ordinal Learning Model (OLM) of Ben-David [5,6]. They construct a so-called *rule base* from a set of training examples. The rule-base R is a subset of the training examples, and is composed of *consistent* and *irredundant* examples. Consistency here refers to the monotonicity requirement. The algorithm sequentially adds examples from the training set to the rule-base, but if an example violates the monotonicity restriction with one or more examples already present in R, then it is discarded. Due to the prediction rule of OLM, examples may also be redundant with respect to the current rule base. OLM allocates a new case $\mathbf{x}_0$ to the largest class among the points in R that precede it:

$$f_{\text{OLM}}(\mathbf{x}_0) = \max\{y \mid (\mathbf{x}, y) \in R \wedge \mathbf{x} \leq \mathbf{x}_0\} \tag{5}$$

As a consequence, if $(\mathbf{x}, y) \in R$ then any $(\mathbf{x}', y)$ with $\mathbf{x} \leq \mathbf{x}'$ does not affect the labeling of new instances. Hence $(\mathbf{x}', y)$ is redundant with respect to $(\mathbf{x}, y)$. If there is no $(\mathbf{x}, y) \in R$ with $\mathbf{x} \leq \mathbf{x}_0$, then $\mathbf{x}_0$ is allocated to the class of the point in R nearest to it, that is, according to the 1-nearest neighbor rule. We note that the composition of the final rule-base critically depends on the order in which the examples are processed. In particular, when an unfortunate choice is made for the first example, then many of

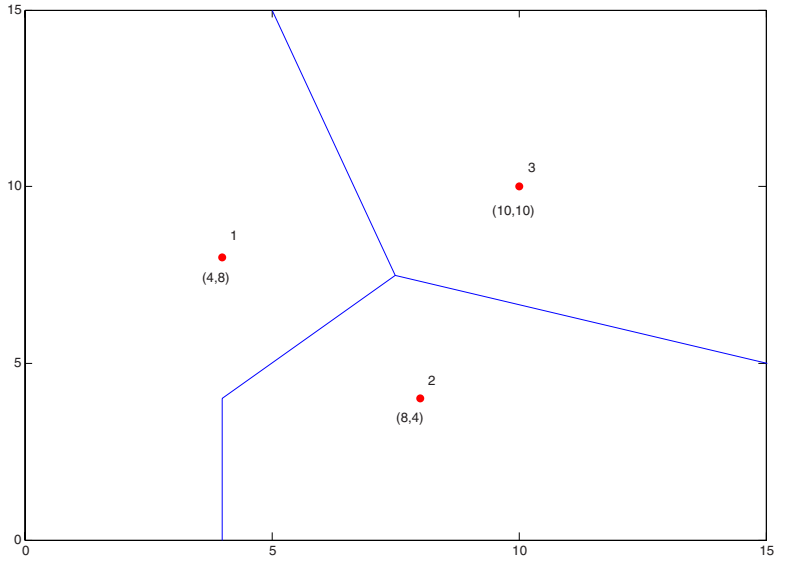


Fig. 4. Allocation rule of monotone 1-nearest neighbour. For each data point its (x_1, x_2) coordinates and class label are given.

the training samples will have to be discarded. Furthermore, as pointed out in [9], non-monotone prediction results are possible due to the 1-nearest neighbor rule used in case no points smaller than $\mathbf{x}_0$ are contained in the rule-base. The most important difference between OLM's prediction rule and ours, is that OLM does not make use of the class labels of nearby points in making its predictions: as shown in equation (5) it simply takes the maximum label of all points that are smaller than the point to be predicted.

Cao-Van [9] presents an algorithm called Ordinal Stochastic Dominance Learner (OSDL), which learns a collection of probability distributions over the class variable, under the restriction that

$$\mathbf{x} \leq \mathbf{x}' \Rightarrow \sum_{j=1}^{i} \Pr(y = j \mid \mathbf{x}) \geq \sum_{j=1}^{i} \Pr(y = j \mid \mathbf{x}'), \tag{6}$$

for $i = 1, 2, \ldots, k-1$. In words, if $\mathbf{x}$ precedes $\mathbf{x}'$ in the ordering, then the distribution of Y in $\mathbf{x}'$ must be stochastically larger than the distribution of Y in $\mathbf{x}$. Although the interpretation of the monotonicity constraint in terms of stochastic dominance is a useful one for probabilistic classifiers, an allocation rule that assigns an input point to the mode of this distribution will not in general be monotone, unless the class variable is binary. In case an outright assignment to a class is required, OSDL therefore takes the median class value according to $\widehat{\Pr}(Y|\mathbf{x})$. This could be interpreted as an attempt to minimize L_1 loss, although this is not stated explicitely. The conditional probabilities $\Pr(Y|\mathbf{x})$ are estimated in a nonparametric way; for details we refer to [9].

Dykstra et al.[11] propose a nonparametric monotonic classification procedure that minimizes L_1 loss

$$\sum_{i=1}^{N} |y_i - f(\mathbf{x}_i)|,$$

subject to

$$\mathbf{x}_i \leq \mathbf{x}_j \Rightarrow f(\mathbf{x}_i) \leq f(\mathbf{x}_j),$$

where the class labels are numbered $\{1, 2, \ldots, c\}$. Their algorithm requires the performance of $c - 1$ isotonic regressions [24] to find an optimal solution. They also provide an algorithm that minimizes L_2 loss that requires the performance of a single isotonic regression. Note that in the important special case of binary classification, minimizing either of these loss functions results in minimal 0/1 loss as well. This is however not the case if there are more than two class labels. The use of squared error loss or absolute error loss presupposes more than an ordering of the class values. Even though these values may be numbered $1, 2, \ldots, c$ for convenience, this does not imply that performance of numerical operations on them is meaningful. On the other hand, it does make sense to presume that classifying a class 1 observation as class 5, is worse than classifying it as class 2.

Dykstra et al.[11] indicate possibilities to extend the relabeled training data to a monotone prediction rule for the entire input space, but like OLM without using any information in the training data beyond the ordering of data points.

6 Experiments

In order to test the proposed classification algorithm, we conducted a number of experiments. In all these experiments, we compared the performance of monotone kNN with that of standard kNN, in order to make sure we are not obtaining monotone models at the expense of predictive accuracy. If a monotone model is really required, then a small increase of the error might be acceptable, but clearly this should be within reasonable limits. On the other hand, if the problem really is monotone, then we might even expect an improvement of the accuracy.

We selected a number of data sets for which the presence of an increasing (or decreasing) relation between the attributes and the response variable was a priori plausible. Table 1 gives an overview of the data sets we used. All data sets have been taken from the UCI machine learning repository [4], except for *Windsor Housing*[1] [2], and *Employee Selection*[2] [6].

As an example, in Table 2 we give the signs of the relations between the attributes and the response that we use for the *AutoMpg* data set.

For the Australian credit approval data, we only used columns 7, 8, 9 and 10 of the attributes from the original data set. For the Boston housing data, we excluded the

[1] Available from the Journal of Applied Econometrics Data Archive at `http://econ.queensu.ca/jae/`

[2] Available at `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`

Table 1. Data sets used in the experiments. The number of attributes given is *after* preprocessing. The column labeled *Comparability* gives the fraction of all pairs of data points $\mathbf{x}, \mathbf{x}'$ for which $\mathbf{x} \leq \mathbf{x}'$, or $\mathbf{x}' \leq \mathbf{x}$.

Data set	# points	# attributes	Target	Comparability
Australian credit approval	690	4	Binary	0.7162
AutoMpg	392	7	Numeric	0.4009
Boston housing	506	12	Numeric	0.1910
Employee Selection	488	4	9 Classes	0.7065
Haberman survival	306	3	Binary	0.3123
Machine (cpu performance)	209	6	Numeric	0.4950
Pima indians diabetes	768	8	Binary	0.0732
Windsor housing	546	11	Numeric	0.2737
Wisconsin breastcancer	683	9	Binary	0.2710

Table 2. Signs used between the target *Miles per gallon* and the different attributes in the Auto-Mpg data set

Attribute	Type	Sign
mpg	continuous	target
cylinders	multi-valued discrete	$-$
displacement	continuous	$-$
horsepower	continuous	$-$
weight	continuous	$-$
acceleration	continuous	$+$
model year	multi-valued discrete	$+$
origin	multi-valued discrete	$+$

Charles River dummy variable. In the experiments we used a number of data sets with a binary target, one with 9 class values (Employee Selection) and some with a numeric target (see Table 1). The numeric targets have been discretized into two and four intervals; the intervals were chosen so that each one contained approximately the same number of cases. In order to check whether our a priori ideas about monotonicity are confirmed by the data, we compared the number of non-monotone pairs present in a data set of size N, to the average number of non-monotone pairs of that same data set, but with the N class labels randomly permuted. The idea is that such a random permutation of class labels represents a non-monotone process, and hence the distribution obtained by computing the number of non-monotone pairs for a great number of such random permutations, can be loosely interpreted as its distribution under the null-hypothesis of a non-monotone process. Table 3 shows the result of these computations for the data sets with binary class label. The last column gives the ratio of the observed number of non-monotone pairs (as given in the first column) to its average for 1000 permuted data sets (as given in the second column). Note that the Haberman data set has by far the highest ratio, and so perhaps the monotonicity assumption is dubious in this case. We don't have a clearcut criterion to decide on that however. Table 4 provides the same information for the data sets with non-binary classes.

Table 3. Monotonicity test results: two classes

Data set	# pairs	Mean	Std.dev.	Ratio
Australian	8087	42649.0	2258.1	0.190
AutoMpg	21	7716.5	733.3	0.003
Boston	309	6113.0	612.5	0.051
Haberman	1784	2848.2	379.4	0.626
Machine	86	2714.0	311.4	0.032
Pima	482	4923.9	569.2	0.098
Windsor	429	10187.0	875.1	0.042
Wisconsin	9	14472.0	1357.4	0.001

Table 4. Monotonicity test results: non-binary classes

Data set	# pairs	Mean	Std.dev.	Ratio
AutoMpg	74	11557.0	816.4	0.006
Boston	691	9175.8	739.4	0.075
Employee	1125	34236.0	1600.1	0.033
Machine	167	4070.2	359.2	0.041
Windsor	1328	15302.0	970.2	0.087

The experiments were performed with 10-fold cross-validation. For monotone nearest neighbour this was done as follows. For each fold, we

1. relabeled the observations in 9 parts of the data to remove any monotonicity violations;
2. used the relabeled data to predict the class labels of the remaining part with the monotone nearest neighbour rule.

For prediction we considered the quasi-monotone prediction rule (predictions have to be consistent only with the training data) as well as the monotone prediction rule (predictions also have to be consistent among themselves). For the monotone prediction rule, the points predicted thus far were used *only* to determine the interval of allowed class labels for a new point; they were not used in voting for the class label of the new point.

The results for the quasi monotone prediction rule for problems with two classes are given in Table 6, and for problems with more than two classes in Table 7. They were computed with prediction rule variant 1 as discussed in Section 4. Preliminary experiments showed the results of the two variants were virtually the same, and variant 1 is easier to compute. Likewise, the results for the monotone prediction rule were virtually identical to those for the quasi monotone rule, and are therefore not reported separately. The results have been summarized in Table 6 and Table 7 as follows. We took the best result of kNN for $k = 1, 3, 5$ and compared its error with the error of monotone kNN for that same value of k (usually they had their lowest error rate for the same value of k). The last column indicates what value of k that was. The p-values were

Table 5. Cross-table for comparison of classifiers. For example, b is the number of cases classified incorrectly by kNN but correctly by monotone kNN.

	I_{mkNN}	C_{mkNN}
I_{kNN}	a	b
C_{kNN}	c	d

Table 6. Comparison of error rates of kNN and monotone kNN with quasi-monotone prediction rule for two-class problems

data set	kNN	mkNN	Winner	p-value	k
Australian	18.3%	16.4%	mkNN	0.0984	5
AutoMPG	8.7%	7.9%	mkNN	0.6476	1
Boston	20.6%	18.8%	mkNN	0.1996	1
Haberman	26.5%	28.4%	kNN	0.4050	3
Machine	15.3%	14.8%	mkNN	1	1
Pima	25.7%	25.9%	kNN	0.9050	3
Windsor	26.4%	20.9%	mkNN	0.0001	5
Wisconsin	3.7%	3.5%	mkNN	1	5

computed using an exact binomial test. We computed a cross-table as given in Table 5 and performed a binomial test of b successes on $b + c$ trials under

$$H_0 : \pi = \tfrac{1}{2} \qquad\qquad H_a : \pi \neq \tfrac{1}{2}$$

where π denotes the probability of success. The p-value was computed with the function `binom.test` in the `R` system[3].

Comparing the error rates for the two-class problems (see Table 6), monotone kNN performs better in all cases, except for the Haberman and Pima data sets. The result for Haberman is not surprising, given the relatively high number of nonmonotone pairs we found in our preliminary calculations. For the Pima data, this ratio is also relativey high, but not as high as for the Australian data, and there we *did* find a substantial improvement of monotone kNN over standard kNN. Hence, the computed ratio by itself is not a perfect indicator for the success of the monotone model.

Looking at the problems with more than two classes (see Table 7), the advantage of enforcing the monotonicity constraint appears even more prominent. Monotone kNN has the lower estimated error rate in all cases, and in two cases significantly so. The effect of the monotonicity constraint can be appreciated clearly by looking at the performance for $k = 1$ (see Table 8): it appears to reduce if not prevent the overfitting of standard kNN. The *Employee Selection* data set is a good example: standard kNN breaks down, whereas monotone kNN isn't performing much worse than for higher values of k.

[3] See `www.r-project.org`

Table 7. Comparison of error rates of kNN and monotone kNN with quasi-monotone prediction rule for problems with more than two classes

data set	kNN	mkNN	Winner	p-value	k
AutoMPG	22.2%	21.9%	mkNN	1	1
Boston	49.8%	42.1%	mkNN	7.2×10^{-6}	5
ESL	30.1%	29.7%	mkNN	0.890	5
Machine	37.8%	34.5%	mkNN	0.324	3
Windsor	51.8%	46.5%	mkNN	0.009	3

Table 8. Comparison of error rates of kNN and monotone kNN for problems with more than two classes and $k = 1$

data set	kNN	mkNN	Winner	p-value
AutoMPG	22.2%	21.9%	mkNN	1
Boston	50.0%	44.5%	mkNN	0.0015
ESL	45.1%	30.5%	mkNN	2.231×10^{-11}
Machine	39.7%	33.5%	mkNN	0.066
Windsor	52.4%	46.2%	mkNN	0.0017

We conclude on the basis of these experiments that enforcing the monotonicity constraint does not lead to a deterioration of predictive accuracy, on the contrary, we have found it usually leads to an improvement. In addition, the monotone models are much more likely to be accepted by their intended users, since its predictions are in accordance with their qualitative domain knowledge.

7 Conclusion

We have proposed an adaptation of the k-nearest neighbour rule, to allow for the inclusion of monotonicity constraints. Such constraints can often be elicited reliably from subject area experts. We have shown that the use of monotonicity constraints can give substantial improvements in predictive performance over the standard k-nearest neighbour classifier. More importantly, the resulting models are much more likely to be accepted by their intended users, because their predictions are in accordance with their qualitative domain knowledge.

The results we obtained encourage us to explore further possibilities in this direction. One could, for example, investigate how minimization of L_1 or L_2 loss in the relabeling phase (as proposed by Dykstra et al. [11]) would influence the predictive performance of the monotone nearest neighbour rule. Minimization of the error rate on the training sample (as performed by the current relabeling algorithm) does after all not necessarily lead to the lowest error on a test sample. Another possibilty is to look for improvements in the relabelling phase. Currently, points are relabelled to arbitrary values from their allowed intervals. More sophisticated alternatives could be considered here.

References

1. Altendorf, E.A., Restificar, A.C., Dietterich, T.G.: Learning from sparse data by exploiting monotonicity constraints. In: Bacchus, F., Jaakkola, T. (eds.) Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), pp. 18–25. AUAI Press (2005)
2. Anglin, P.M., Gençay, R.: Semiparametric estimation of a hedonic price function. Journal of Applied Econometrics 11(6), 633–648 (1996)
3. Archer, N.P., Wang, S.: Application of the backpropagation neural network algorithm with monotonicity constraints for two-group classification problems. Decision Sciences 24(1), 60–75 (1993)
4. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
5. Ben-David, A.: Automatic generation of symbolic multiattribute ordinal knowledgebased DSS: methodology and applications. Decision Sciences 23, 1357–1372 (1992)
6. Ben-David, A., Sterling, L., Pao, Y.: Learning and classification of monotonic ordinal concepts. Computational Intelligence 5, 45–49 (1989)
7. Ben-David, A.: Monotonicity maintenance in information-theoretic machine learning algorithms. Machine Learning 19, 29–43 (1995)
8. Bloch, D.A., Silverman, B.W.: Monotone discriminant functions and their applications in rheumatology. Journal of the American Statistical Association 92(437), 144–153 (1997)
9. Cao-Van, K.: Supervised ranking, from semantics to algorithms. PhD thesis, Universiteit Gent (2003)
10. Druzdzel, M.J., van der Gaag, L.C.: Elicitation of probabilities for belief networks: combining qualitative and quantitative information. In: Besnard, P., Hanks, S. (eds.) Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI 1995), pp. 141–148. Morgan Kaufmann, San Francisco (1995)
11. Dykstra, R., Hewett, J., Robertson, T.: Nonparametric, isotonic discriminant procedures. Biometrika 86(2), 429–438 (1999)
12. Feelders, A., Pardoel, M.: Pruning for monotone classification trees. In: Berthold, M.R., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 1–12. Springer, Heidelberg (2003)
13. Feelders, A., van der Gaag, L.: Learning Bayesian network parameters with prior knowledge about context-specific qualitative influences. In: Bacchus, F., Jaakkola, T. (eds.) Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), pp. 193–200. AUAI Press (2005)
14. Feelders, A., Velikova, M., Daniels, H.: Two polynomial algorithms for relabeling non-monotone data. Technical Report UU-CS-2006-046, Department of Information and Computing Sciences, Utrecht University (2006)
15. Ford, L.R., Fulkerson, D.R.: Flows in networks. Princeton University Press, Princeton (1962)
16. Gamarnik, D.: Efficient learning of monotone concepts via quadratic optimization. In: Proceedings of the eleventh annual conference on computational learning theory, pp. 134–143. ACM Press, New York (1998)
17. Garey, M., Johnson, D.: Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York (1979)
18. Karpf, J.: Inductive modelling in law: example based expert systems in administrative law. In: Proceedings of the third international conference on artificial intelligence in law, pp. 297–306. ACM Press, New York (1991)
19. Meyer, M.A., Booker, J.M.: Eliciting and Analyzing Expert Judgment: A Practical Guide. Statistics and Applied Probability. ASA-SIAM, Philadelphia (2001)
20. Möhring, R.H.: Algorithmic aspects of comparability graphs and interval graphs. In: Rival, I. (ed.) Graphs and Order, pp. 41–101. Reidel (1985)

21. Pazzani, M.J., Mani, S., Shankle, W.R.: Acceptance of rules generated by machine learning among medical experts. Methods of Information in Medicine 40, 380–385 (2001)
22. Potharst, R., Bioch, J.C.: Decision trees for ordinal classification. Intelligent Data Analysis 4(2), 97–112 (2000)
23. Rademaker, M., De Baets, B., De Meyer, H.: On the role of maximal independent sets in cleaning data for supervised ranking. In: 2006 IEEE International Conference on Fuzzy Systems, pp. 1619–1624 (2006)
24. Robertson, T., Wright, F., Dykstra, R.L.: Order Restricted Statistical Inference. Wiley, Chichester (1988)
25. Royston, P.: A useful monotonic non-linear model with applications in medicine and epidemiology. Statistics in Medicine 19(15), 2053–2066 (2000)
26. Sill, J.: Monotonic networks. In: Advances in neural information processing systems. NIPS, vol. 10, pp. 661–667 (1998)

Modeling Transfer Relationships Between Learning Tasks for Improved Inductive Transfer

Eric Eaton[1], Marie desJardins[1], and Terran Lane[2]

[1] University of Maryland Baltimore County,
Department of Computer Science and Electrical Engineering
{ericeaton,mariedj}@umbc.edu
[2] University of New Mexico, Department of Computer Science
terran@cs.unm.edu

Abstract. In this paper, we propose a novel graph-based method for knowledge transfer. We model the transfer relationships between source tasks by embedding the set of learned source models in a graph using transferability as the metric. Transfer to a new problem proceeds by mapping the problem into the graph, then learning a function on this graph that automatically determines the parameters to transfer to the new learning task. This method is analogous to inductive transfer along a manifold that captures the transfer relationships between the tasks. We demonstrate improved transfer performance using this method against existing approaches in several real-world domains.

1 Introduction

Knowledge transfer from previously learned tasks to a new task is a fundamental component of human learning. Transfer enables us to learn complex tasks quickly by automatically building on our previous knowledge. Recent research efforts have shown that transfer can also improve machine learning, enabling more rapid learning or higher levels of performance.

Most machine learning methods for transfer rely on an explicit set of *source* tasks to identify a set of model parameters that can be transferred to a new *target* task. In many cases, these source tasks are hand-selected by an expert in advance. Methods for transfer may combine information from all source tasks [1,2] or may use information from only a few tasks chosen by an automated process [3]. Accidentally transferring from irrelevant source tasks may inhibit learning and decrease performance—a phenomenon known as *negative transfer*. Our approach to transfer explicitly models the transfer relationships between the source tasks to automatically avoid this problem and transfer only relevant information.

Given a set of source tasks and a target task, our method attempts to automatically determine the knowledge to transfer in learning the target task. In our formulation, this knowledge is a vector of model parameters. We estimate the transfer relationships between the source tasks and embed them into a graph, using a notion of *transferability* to determine the edge weights. This model transfer graph corresponds to a discrete approximation of a high-dimensional manifold

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 317–332, 2008.

that captures the transfer relationships between the source tasks. Tasks that are close on this manifold have high transferability; tasks that are far apart have low transferability. Each task has an associated vector of model parameters that represents the knowledge at its location.

Intuitively, each location on the transfer manifold has an ideal parameter vector that should be transferred in learning a task at that location. Therefore, we can determine the knowledge to transfer to a target task by approximating the parameter vector at the target task's location on the manifold. Given a new target task, we first extend the the graph to include the new task. We define a function on the graph that determines the parameters to transfer to each location in the graph. By its construction, this transfer function respects the local geometry of the graph and, therefore, the transfer relationships among the source tasks. We learn the transfer function using the source tasks' parameters as samples of the function at various locations on the transfer manifold. Then, we evaluate the function at the new task's location to yield the parameter vector to transfer in learning the new task. We also define a reusable form of the transfer function that can be used for multiple transfer scenarios without relearning.

2 Related Work

Parameter-based transfer has been used by Marx et al. [1] to learn logistic regression models. They fit logistic regression models independently to each source task, and then estimate the prior distribution for the target model's weights *a posteriori* from the source tasks' models. Kienzle & Chellapilla [2] use a weight vector for transfer in SVMs, biasing the regularization term toward the weight vector, instead of the zero vector as in standard SVM training. The biased logistic regression method we propose in Sect. 3 is based on a combination of biased regularization and Marx et al.'s logistic regression transfer.

In contrast to the approaches of Marx et al. and Kienzle & Chellapilla, which combine knowledge from all given source tasks for transfer, Thrun & O'Sullivan's Task Clustering (TC) algorithm [3,4] groups tasks for more selective transfer. Their method also transfers parameter vectors, sharing weighted Euclidean distance metrics between k-nearest-neighbor classifiers. Transfer occurs by having one k-nearest-neighbor model use the distance metric from another model. Their approach optimizes a single distance metric for each cluster, effectively determining an average parameter vector for each cluster of tasks. Upon receiving a new task, the TC algorithm matches the new task to a cluster, then transfers that cluster's distance metric to the new task. Our approach is similar to Thrun & O'Sullivan's in determining the transfer relationships between tasks. However, the TC algorithm transfers only a single parameter vector to all tasks in a cluster, while our flexible transfer function allows each location on the transfer surface to have a different parameter vector based on the local geometry.

Bakker & Heskes [5] take a Bayesian approach to clustering tasks, using EM to optimize the clusters. They also use a gating network, similar to that used in the mixture-of-experts model [6], on top of the Bayesian EM framework to allow

the priors to vary depending on the task's features. Pratt's Discriminability-Based Transfer method [7] for neural networks selectively transfers weights from a learned network, modifying them as needed to enable learning on a target task. Explanation-Based Neural Networks [8] use a more indirect approach to parameter transfer, using extracted invariances about a domain to bias the learning of model parameters. These approaches allow some of the model parameters to be dependent on the source tasks, while others are fit to the target task's data. In our approach, we allow all of the transferred model parameters to be modified in the final learned model, if the target task's data warrants such an adjustment.

3 Transfer Using Biased Logistic Regression

We define a *task* as a mapping from an instance space $X \subset \mathbb{R}^d$ to a set of labels $Y \in \mathbb{N}$. All tasks map from the same X to the same Y. The goal for learning the model for a task is to recover the true mapping $X \rightarrow Y$ from the labeled training data. Each learned model can be characterized as a vector of parameters, which can be transferred in learning a model for another task.

Our approach requires a base transfer learning algorithm to learn the models for each task. We use a biased form of logistic regression as the base learning algorithm in the experiments. Biased logistic regression penalizes deviations from a given parameter vector in $\mathbb{R}^d$, effectively biasing the learned model toward the transferred parameters. While we focus on this transfer learning algorithm, our method can utilize other parameter-based transfer learning algorithms.

The well-known logistic regression model gives the probability of an instance x having a binary label y as:

$$P(y = 1|x) = \frac{\exp(x\beta)}{1 + \exp(x\beta)} \ , \tag{1}$$

$$P(y = 0|x) = 1 - P(y = 1|x) \ , \tag{2}$$

where $x \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$. The parameter vector β is obtained by maximizing the log-likelihood of the labeled training data $\{(x_i, y_i)\}_{i=1}^{q}$:

$$l(\beta) = \sum_{i=1}^{q} [y_i \log P(y_i = 1|x_i) + (1 - y_i) \log P(y_i = 0|x_i)] \ . \tag{3}$$

Combining ridge estimation with logistic regression[1] [10,11] adds a penalty on the norm of β, and involves choosing β to maximize the penalized log-likelihood $l^{\lambda}(\beta) = l(\beta) - \lambda\|\beta\|^2$, where λ is the ridge parameter that controls the shrinkage of the norm $\|\beta\| = \sqrt{\sum_j \beta_j^2}$.

To use logistic regression for transfer, we penalize deviations of β from a given transferred vector β_0:

$$l^{\lambda}(\beta) = l(\beta) - \lambda\|\beta - \beta_0\|^2 \ . \tag{4}$$

[1] We use the Weka machine learning toolkit's implementation of this method [9].

This approach is inspired by biased regularization of support vector machines [2,12] and the logistic regression transfer method of Marx et al. [1].

Standard (non-biased) logistic regression corresponds to β_0 as the zero vector. This bias vector β_0 can be transferred from the learned β of another logistic regression model, allowing one logistic regression model to be biased toward the parameters of another model. In practice, x and β are often augmented to include a constant term for the intercept. Note that we do not transfer the constant term, allowing it to be fit individually to each problem. When $\lambda = 0$, the bias term disappears and does not affect the learned weights; as $\lambda \to \infty$, the logistic regression learned weights approach the bias weights.

We use the Bayesian-optimal $\lambda = \frac{\sigma^2}{\tau^2}$ [13], where σ^2 is the variance of the model's log-likelihood errors $\{-\log P(y = y_i|x_i)\}_{i=1}^{q}$, and τ^2 is the variance of the elements of $(\beta - \beta_0)$. Viewed from the perspective of transfer, this assumption implies a normal probability distribution over the transfer from β_0 to β.

The logistic regression transfer method of Marx et al. [1] uses a similar construction, in which they penalize deviations of the model parameters from a given set of normal distributions, considering both means and variances derived from the transferred parameter vectors. The method we use here (based on ridge regression) corresponds to their method using a constant variance for all parameters, which is absorbed into λ. The major problem with using their method in this application is that it is dependent on having a *set* of source tasks from which to estimate the parameter variances and thereby the regularization; in this application, we have only one source parameter vector and, therefore, no variance.

4 Modeling the Transferability Between Source Tasks

Given a set of source tasks $\{t_i\}_{i=1}^{n}$, our approach is composed of three steps:

- **Learn the base models** $\{m_i\}_{i=1}^{n}$ for the source tasks $\{t_i\}_{i=1}^{n}$ (Sect. 4.1).
- **Construct the model transfer graph** to model the transfer relationships between the source tasks (Sect. 4.2).
- **Transfer to a new task** t_{n+1} by extending the model transfer graph to include t_{n+1}, and then learning the transfer function f to determine the parameter vector v_{n+1} to transfer to t_{n+1} (Sect. 5).

4.1 Learning the Base Models

Given the set of source tasks $\{t_i\}_{i=1}^{n}$, our first step is to learn the set of base models $\{m_i\}_{i=1}^{n}$ for the source tasks. For a task t_i, we learn the corresponding model m_i using biased logistic regression without transfer, biasing the model parameters toward zero. We assume that sufficient training examples are given for each source task to learn base models that have a high degree of performance.

Each trained model m_i has an associated parameter vector $v_i \in \mathbb{R}^{\theta}$, which can be transferred in learning models for other tasks. For biased logistic regression models, $\theta = d$, with v_i corresponding to the learned β vector.

4.2 Constructing the Model Transfer Graph

Given the set of source tasks $\{t_i\}_{i=1}^n$ and their corresponding learned models $\{m_i\}_{i=1}^n$, we embed these tasks in a space that captures the transfer relationships between the tasks. Two tasks that have high transferability should be close in the space; tasks that have low or negative transferability should be far apart.

We define transferability from task t_i to t_j as the change in performance on task t_j between learning with and without transfer from t_i's model. Although we focus on this definition of transferability, our approach is general enough to use other measures. This definition is very similar to the approach used by Thrun and O'Sullivan [3]. While their task clustering method simply looks at the change in performance for a specific number of training instances, we also consider the average transferability over the entire learning curve.

We model the space of transfer using a *model transfer graph*, with each task as a vertex in the graph. A pair of vertices are connected via an edge if they have positive transferability; this edge is weighted based on the amount of positive transferability between the tasks, which is in $(0, 1]$.

We *could* directly plot the models in $\mathbb{R}^\theta$, since each model can be characterized by its transferable parameter vector. However, this embedding ignores that the transferred knowledge must *improve performance* on the target task. Similarity between two parameter vectors does not imply that models using those vectors will have similar performance on a task. Therefore, it is important to measure similarity in the transfer space based on *transferability*.

The model transfer graph corresponds to a discrete approximation of the continuous transfer manifold, using transferability as the metric. The source tasks are known samples of various locations on the manifold, with each task t_i having an associated parameter vector v_i. Transfer to a new task, as described in Sect. 5, occurs by approximating the location of the new task on the manifold, then using the transfer function to determine the parameter vector to transfer.

Measuring Transferability

We measure transferability from task t_i to t_j as the direct change in performance between learning with and without transfer. For a task t_j, we can generate two learning curves for the task's model: one for learning t_j's model with transfer from t_i, and one for learning the model without transfer. For learning with transfer, we use logistic regression biased toward the parameter vector v_i that characterizes t_i's base model m_i. For learning without transfer, we use standard regularized logistic regression, which is biased toward the zero vector. Any performance measure that evaluates to a real number in $[0, 1]$ can be used to compute the performance (e.g., predictive accuracy, f-measure). In our experiments, we use predictive accuracy on the held-out test set.

Let $\mathit{performance}_j(q)$ be the performance on task t_j without transfer given q training instances, and let $\mathit{performance}_{i \to j}(q)$ be the performance on task t_j with transfer from t_i given q training instances from task t_j. Then, the transferability from task t_i to t_j is given by

$$\mathit{transfer}_{i \to j}(q) = \mathit{performance}_{i \to j}(q) - \mathit{performance}_j(q) \ . \tag{5}$$

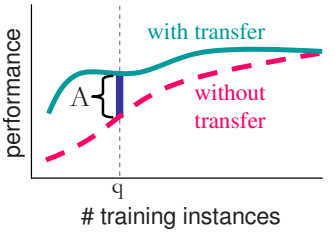


Fig. 1. The transferability measure: $transfer(q) = A$

Note that $transfer_{i \to j} \in [-1, 1]$, with positive transfer falling in $(0, 1]$.

This definition of transferability for a given amount of training data lends itself to a definition of overall transferability from task t_i to t_j. Specifically, we can average Eqn. 5 over a range of values for q, yielding a measure of the overall transferability from t_i to t_j. By considering $transfer_{i \to j}$ across the entire learning curve, we compute the expected amount of transfer for an arbitrary amount of training data. This computation assumes a uniform probability distribution over the amount of training data that will be available for a new task; it is a simple matter to scale this computation for a non-uniform probability distribution.

Defining the Model Transfer Graph

The spectral graph analysis techniques we use to analyze the transfer surface (Sect. 5.2) rely on the model transfer graph being undirected. A symmetric affinity measure is the most natural representation for the transfer surface. However, transfer by its nature is directed from source knowledge to a target task. Therefore, $transfer_{i \to j}$ is *not* guaranteed to be the same as $transfer_{j \to i}$.

We define the symmetric undirected transferability between tasks t_i and t_j to be the minimum of the two directed transferabilities:

$$transfer_{i,j}(q) = \min \left(transfer_{i \to j}(q), transfer_{j \to i}(q) \right) \quad . \tag{6}$$

The largest potential problem is overestimating the amount of transfer between two tasks, and using the minimum of the directed transferabilites ensures that our estimate of the transfer is as large as possible without being a potential overestimation. Using other forms of symmetrization, such as taking the average or maximum, could lead to overestimation. While this construction underestimates the amount of transfer, we show empirically that it performs well in Sect. 6.

We define the vertices of the model transfer graph to be the source tasks $V = \{t_i\}_{i=1}^{n}$, and the symmetric adjacency matrix A for q training instances as

$$A_{i,j}(q) = \begin{cases} 0 & \text{if } i = j, \\ \max \left(0, transfer_{i,j}(q)\right) & \text{otherwise.} \end{cases} \tag{7}$$

Since we need only model the positive transfer, this construction eliminates all negative edges from $A(q)$. We store multiple snapshots of the graph's adjacency

matrix $\{A(q_i)\}_{q_1}^{q_k}$ at various numbers of training instances. In the experiments, we sampled the learning curve every five percent of the training data, so $k = 20$ with successive q_i's in 5% increments. To transfer to a new task, we select the current picture of the transferability space for the given number $\hat{q}$ of target task training instances, and use that version of the model transfer graph $G(\hat{q}) = (V, A(\hat{q}))$.

5 Transfer to a New Task

From Sect. 4, we can construct a model transfer graph to represent the transfer relationships among the source tasks. In this section, we describe a procedure for using the graph to determine the parameters to transfer to a new task.

Given $\hat{q}$ training instances of a new target task t_{n+1}, we can extend the model transfer graph $G(\hat{q})$ to include t_{n+1}. We then learn a transfer function on the extended graph to determine the parameter vector to transfer to the new task. This process is equivalent to interpolating the position of t_{n+1} on the transfer manifold, and then determining the transfer function's value at that point.

5.1 Extending the Model Transfer Graph

Given a small sample ($\hat{q}$ instances) of the data from t_{n+1} (much less data than was given for any other task $t_1 \ldots t_n$), we approximate task t_{n+1}'s location in the graph by computing its transferability from every other task t_i: $\{\mathit{transfer}_{i \to n+1}(\hat{q})\}_{i=1}^{n}$. This yields a set of weighted edges[2] between t_{n+1} and all other tasks $t_1 \ldots t_n$, allowing us to localize t_{n+1} in the transfer graph. Let these weights be $\hat{w}_1 \ldots \hat{w}_n$, where $\hat{w}_i = \mathit{transfer}_{i \to n+1}(\hat{q})$.

The extended model transfer graph that includes task t_{n+1} can now be defined by $\hat{G} = (\hat{V}, \hat{A})$, where $\hat{V} = V \bigcup \{t_{n+1}\}$ and $\hat{A}$ is the $(n+1) \times (n+1)$ extended adjacency matrix given by

$$\hat{A} = \begin{bmatrix} A(\hat{q}) & \hat{w}^T \\ \hat{w} & 0 \end{bmatrix} \quad . \tag{8}$$

5.2 Learning the Transfer Function

Once the graph $G(\hat{q})$ has been extended to include the new target task, the next step is to learn the transfer function on $\hat{G}$ and use it to determine the knowledge to transfer in learning t_{n+1}. Each vertex i in the extended model transfer graph $\hat{G}$ has some associated transfer knowledge given by its parameter vector v_i. For the new target task t_{n+1}, this transfer knowledge is unknown, and the transfer function can estimate it automatically from the source tasks' parameter vectors.

The source tasks represent a known sample of the transfer surface, with the parameter vectors $\{v_i\}_{i=1}^{n}$ representing the transfer knowledge at these sample locations on the manifold. Each parameter vector v_i is in $\mathbb{R}^{\theta}$. We assume that there is some function that determines the transfer knowledge for a task based on

[2] We ignore the directionality of the edges, since the transfer is one-way only.

that task's location on the transfer surface. This *transfer function* $\hat{f} : \hat{V} \rightarrow \mathbb{R}^{\theta}$ is able to assign a parameter vector to each task located on the transfer surface.

The source tasks' parameter vectors $\{v_i\}_{i=1}^{n}$ represent known values of the transfer function $\hat{f}$ at various locations (given by the source tasks) on the transfer surface. Therefore, the source tasks' locations coupled with their parameter vectors provide training data for learning the transfer function $\hat{f}$. To transfer to a new task t_{n+1}, we can evaluate the learned transfer function at t_{n+1}'s location on the transfer surface to yield a parameter vector for t_{n+1}.

In order to ensure that the learned transfer function respects the transfer relationships between the tasks, we must model the transfer function in a manner that respects the model transfer graph's geometry. To do this, we define $\hat{f}$ using a set of basis functions for the graph determined by spectral graph theory.

Determining the Basis Functions

This section describes the spectral graph theory [14] techniques we use to derive the basis functions, which allow us to define a transfer function that will respect the geometry of the model transfer graph.

Let $G = (V, A)$ be the model transfer graph, which is an undirected connected weighted graph with a set of n vertices V and a weighted $n \times n$ adjacency matrix A. Recall that $A_{u,v} \neq 0$ implies that there is an edge depicting positive transferability between vertices u and v. We can denote the degree of vertex v by $d_v = \sum_{u=1}^{n} A_{u,v}$. Let T be the diagonal matrix where $T_{v,v} = d_v$.

Spectral graph theory allows us to define a set of basis functions for G based on the graph Laplacian, an operator defined by the Laplacian matrix. The combinatorial Laplacian matrix L for the graph is given by $L = T - A$ [14].

We can also define the *normalized Laplacian* $\mathcal{L} = I - T^{-\frac{1}{2}} A T^{-\frac{1}{2}}$, where I is the identity matrix [14]. While both forms of the Laplacian are applicable to our work, we found that the combinatorial Laplacian (hereafter referred to as just the *Laplacian*) worked better in our experiments and so we focus on it.

The Laplacian L is symmetric; therefore, its eigenvalues are all real and non-negative. The eigendecomposition of L yields $L = Q \Lambda Q^T$, where Λ is the diagonal matrix of eigenvalues $[\lambda_1 \ \ldots \ \lambda_n]$ and the columns of Q are the eigenvectors $[q_1 \ \ldots \ q_n]$. Let $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues, and let eigenvector q_i correspond to λ_i. Spectral graph theory tells us that the smallest eigenvalue λ_1 is always 0 (with multiplicity 1, since G is connected) and q_1 is constant over all vertices. The eigenvectors in Q form an orthonormal basis for L.

Spectral graph theory has connections to Riemannian manifolds, which we use to define the surface on which transfer occurs. The model transfer graph G represents a sample of the continuous transfer manifold $\mathcal{M}$, with the vertices as points on the manifold and the edges connecting points that are close to each other on the manifold (i.e., tasks that have high transferability).

Let g be a smooth function $g : \mathcal{M} \rightarrow \mathbb{R}^{\theta}$ representing the transfer function on the Riemannian transfer manifold $\mathcal{M}$ with Riemannian transferability metric ψ. The Laplace-Beltrami operator Δ is defined to be the divergence of the gradient of $\mathcal{M}$, and can act on g. Hodge theory [15] implies that g has a unique spectrum based on the eigenfunctions of the Laplace-Beltrami operator on $\mathcal{M}$.

Since we have only a sample of the transfer manifold given by the source tasks, we work with a discrete form f of the true transfer function g. The graph Laplacian is a discrete form of the continuous Laplace-Beltrami operator that acts on a function $f : V \to \mathbb{R}^\theta$ defined on G. Like the continuous g, f can also be characterized by the eigenfunctions of the Laplacian; therefore, Q forms a set of basis functions which we can use to define the transfer function f.

While this analysis has focused on the original model transfer graph G, we can similarly analyze the extended graph $\hat{G}$. As with G, we can take the eigenvectors $\hat{Q}$ of $\hat{G}$'s graph Laplacian $\hat{L}$ to form a set of basis functions that can characterize the extended transfer function $\hat{f}$ on $\hat{G}$.

Modeling the Transfer Function

Using the basis functions for $\hat{G}$, we can model the transfer function $\hat{f} : \hat{V} \to \mathbb{R}^\theta$. The eigenvectors $\hat{Q}$ form an orthonormal basis for the set of all functions on $\hat{G}$; therefore, $\hat{f} = \hat{Q}W$ for some $(n+1) \times \theta$ matrix W.

We use the known parameter vectors $\{v_i\}_{i=1}^n$ as samples of the function values on the graph, defining $f = [v_1 \dots v_n]^T$, where the v_i's are column vectors. We can similarly define the basis vectors for these sample points as the corresponding rows of $\hat{Q}$: $Q = \hat{Q}_{1\dots n,*}$. The matrix Q is $n \times (n+1)$, and f is $n \times \theta$.

We fit each column of W separately using regularized least-squares by solving:

$$W_{*,i} = \arg_w \min ||f_{*,i} - Qw||^2 + \left|\left|\sqrt{\hat{\Lambda}}w\right|\right|^2 \quad , \tag{9}$$

where $\sqrt{\hat{\Lambda}}$ serves as the regularization operator in this Tikhonov regularization problem. The operator acts as a weighted penalty on the function's average second-derivative, enforcing smoothness by scaling each eigenvector's weight by its corresponding eigenvalue λ_i, thereby increasing the regularization on higher-order eigenvectors to prevent overfitting with the high-frequency components.

We derive this expression by constraining the smoothness of $\hat{f}$—i.e., the L2 norm of the gradient of $\hat{f}$, given by $\langle \nabla \hat{f}, \nabla \hat{f} \rangle$:

$$\begin{aligned} \langle \nabla \hat{f}, \nabla \hat{f} \rangle &= \langle \hat{f}, \hat{L}\hat{f} \rangle \\ &= (\hat{Q}w)^T (\hat{L}\hat{Q}w) \\ &= w^T \hat{Q}^T (\hat{Q}\hat{\Lambda}\hat{Q}^T \hat{Q}w) \\ &= w^T I \hat{\Lambda} I w \\ &= w^T \hat{\Lambda} w \quad . \end{aligned}$$

Therefore, we can constrain the smoothness of $\hat{f}$ by penalizing the least-squares problem with $w^T \hat{\Lambda} w$, which is equivalent to the penalty $||\sqrt{\hat{\Lambda}}w||^2$ in Eqn. 9. The solution to this least-squares problem is given by

$$W = \left(Q^T Q + \hat{\Lambda}\right)^{-1} Q^T f \quad . \tag{10}$$

This process yields an $(n+1) \times \theta$ matrix for W, which can be used unaltered to form the extended transfer function $\hat{f} = \hat{Q}W$ that assigns a parameter vector

to each vertex in $\hat{G}$. The extended transfer function $\hat{f}$ approximates the known parameter vectors $\{v_i\}_{i=1}^n$ at the source task vertices. At the new target task t_{n+1}, $\hat{f}$ acts as a smoothed interpolent of the source tasks' knowledge at t_{n+1}'s location, respecting the graph geometry and transferability relationships.

By the $\hat{f}$ transfer function, the transferred parameters for the target task are given by $v_{n+1} = \hat{Q}_{n+1,*}W$, where $\hat{Q}_{n+1,*}$ is the $(n+1)^{th}$ row of $\hat{Q}$. We then transfer v_{n+1} in learning t_{n+1}'s model.

5.3 Creating a Reusable Transfer Function

In the procedure we defined in Sect. 5.2, the transfer function must be relearned for each new target task based on the geometry of the extended model transfer graph. The transfer graphs used in the experiments were small enough that we could directly compute the eigendecomposition of $\hat{L}$. However, for very large transfer graphs or for repeated transfer scenarios, this process of recomputing the transfer function becomes a source of inefficiency. In this section, we describe a method for creating a reusable form of the transfer function.

First, we construct the model transfer graph G using the source tasks as described in Sect. 4. For G, we can construct the transfer function $f : V \rightarrow \mathbb{R}^\theta$ for the source tasks by solving the least-squares problem $f = QW$ for an $n \times \theta$ matrix W, where $f = [v_1 \dots v_n]^T$ and Q is the matrix of eigenvectors with eigenvalues Λ of G's graph Laplacian. The solution is given by $W = \left(Q^T Q + \Lambda\right)^{-1} Q^T f$.

While this f operates on G, applying it directly to the extended graph $\hat{G}$ would not work, because there would be more than n eigenvectors. However, we can use the Nyström method to extend G's eigenvectors to new vertices without increasing the number of eigenvectors, thereby allowing us to reuse the learned transfer function f for multiple transfer scenarios.

The Nyström method [16,17,18] allows us to efficiently extend a graph's eigenvectors to include a new vertex. Let $\{\hat{w}_i\}_{i=1}^n$ be the transferability edge weights between a new task t_{n+1} and all the vertices V of the original model transfer graph G. The Nyström extension allows us to extend the eigenvectors Q of G's graph Laplacian to approximate the eigenvector values at the new task t_{n+1} as

$$q_i(t_{n+1}) = \frac{1}{\lambda_i} \sum_{j=1}^{n} \hat{w}_j \, q_i(t_j) \ , \tag{11}$$

where $q_i(t_j)$ is the i^{th} eigenvector applied to task t_j.

Using these extended eigenvectors, we can form an approximation to the true eigenvectors $\hat{Q}$ of $\hat{G}$, the extended model transfer graph that includes t_{n+1}. The Nyström approximation to the eigenvectors is given by $\tilde{Q}$, an $(n+1) \times n$ matrix. Since $\tilde{Q}$ and Q both contain n eigenvectors (recall that $\hat{Q}$ contained $n+1$ eigenvectors), we can approximate the transfer function $\hat{f}$ on $\hat{G}$ by $\tilde{f} = \tilde{Q}W$ using the same weight matrix W without relearning. Then, for the new task t_{n+1}, the transferred parameter vectors are given by $v_{n+1} = \tilde{Q}_{n+1,*}W$.

6 Evaluation

Our experiments examine transfer in two domains: letter and newsgroup recognition. The Letters data set [19] characterizes various fonts of each character using 16 features normalized to lie in $[0, 1]$. The Newsgroup experiments use the 20 newsgroups data [20], characterized by a binary vector of the 100 most discriminating words, as determined by Weka's string-to-wordvector filter [9]. Transfer is often not useful given large amounts of data, since there would be enough data to learn a model with high performance. Both original data sets are very large, so we randomly selected five percent of each to use in the experiments.

For Letters, we took the first 13 letters (A–M) and generated 13 binary tasks of each of these letters against the last 13 letters (N–Z), ensuring that each task had unique negative examples and equal class proportions. For example, the task of recognizing the letter C used 35 "C"s as positive examples and 35 random letters N–Z as negative examples. We chose this construction to yield tasks that would interfere as little as possible with each other. For example, if instead we had converted this data set into 26 one-versus-rest classification problems, there would be interference between the tasks, since one task's positive examples would appear as other tasks' negative examples, diminishing the possibility of transfer. The Newsgroups tasks are constructed similarly, using the first newsgroup[3] in each major category as negative examples for the tasks given by the 13 remaining newsgroups.

The base models for each task were learned from all available data. We then constructed model transfer graphs for both Letters and Newsgroups over 10 trials of 10-fold cross-validation over all available data on the source tasks, excluding the target task from the computations. The held-out fold was used for performance evaluation to generate the baseline and transfer learning curves.

For each target task, we used the task's training data to extend the transfer graph (again, computing the transfer over 10 trials of 10-fold cross-validation on the training data), learned the transfer function on the extended graph, and then used it to estimate the parameter vector for the target task. We evaluated the learned classifier with transfer on the task's held-out test data. This procedure was repeated and averaged over 20 trials of 10-fold cross-validation to generate the learning curves. Table 1 summarizes each transfer scenario used in the experiments.

Figures 2 and 3 compare the performance of our "graph transfer" approach against "hand-selected" transfer, an "average" transfer method, and the baseline of learning without transfer. The "hand-selected" transfer computes the average parameter vector over each target task's related source tasks given in Table 1. For Newsgroups, these related tasks were chosen as the other newsgroups with the same top-level category; the related Letters tasks were chosen based on visual similarity between the letters. The average transfer method simply averages the parameter vectors from all source tasks (including irrelevant tasks), corresponding to several current transfer approaches [1,2].

[3] The negative newsgroups are alt.atheism, comp.graphics, misc.forsale, rec.autos, sci.crypt, soc.religion.christian, and talk.politics.guns.

Table 1. Summary of transfer scenarios

20 Newsgroups		
Target task	**# instances**	**"Hand-selected" source tasks**
comp.windows.x	100	comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware
rec.sport.baseball	100	rec.motorcycles, rec.sport.hockey
sci.space	100	sci.electronics, sci.med
talk.politics.mideast	94	talk.politics.misc, talk.religion.misc

Letters		
Target task	**# instances**	**"Hand-selected" source tasks**
C	70	E, G
E	88	B, F
G	70	C
H	70	K, M
J	72	I, L
L	68	I, J

All of the Newsgroup transfer scenarios contain a mix of both relevant and irrelevant source tasks. The comp.windows.x task (Fig. 2(a)) has a higher proportion of relevant source tasks than the other Newsgroup scenarios, due both to the larger proportion of computer-related newsgroups and the broad applicability of computers to other subjects.

Our graph transfer method shows statistically significant improvement (with at least 95% confidence) over the average parameter vector on the Newsgroup tasks, demonstrating its ability to focus on information from relevant source tasks. The inclusion of irrelevant source tasks in computing the average parameter vector sometimes results in negative transfer, which our graph transfer avoids. These results support the use of localized estimates for the transfer parameters instead of averaging information from all source tasks without regard to transferability. It also shows that our approach can achieve performance near that of expensive "hand-selected" source tasks; in many cases, the performance of the graph transfer and hand-selected transfer are statistically indistinguishable.

Results on the letter transfer scenarios in Figs. 3(a)–(c) mirror the successes of our approach on the Newsgroup tasks. The letter-L transfer scenario (Fig. 3(d)) shows one case where we were unable to obtain clear improvement over the average parameter vector, although in many cases its increased performance over graph transfer is not statistically significant. This scenario also shows the pitfalls of hand-selecting source tasks, in this case based on visual similarities between the letters, in that these hand-selected tasks can unexpectedly result in negative transfer.

Figures 3(e)–(f) depict two extreme transfer scenarios that demonstrate the versatility of our approach. The complete model transfer graph for the letters domain showed that all letters had positive transfer (on average) to the letter-H task, with the exception of the letter-B task showing very slight negative

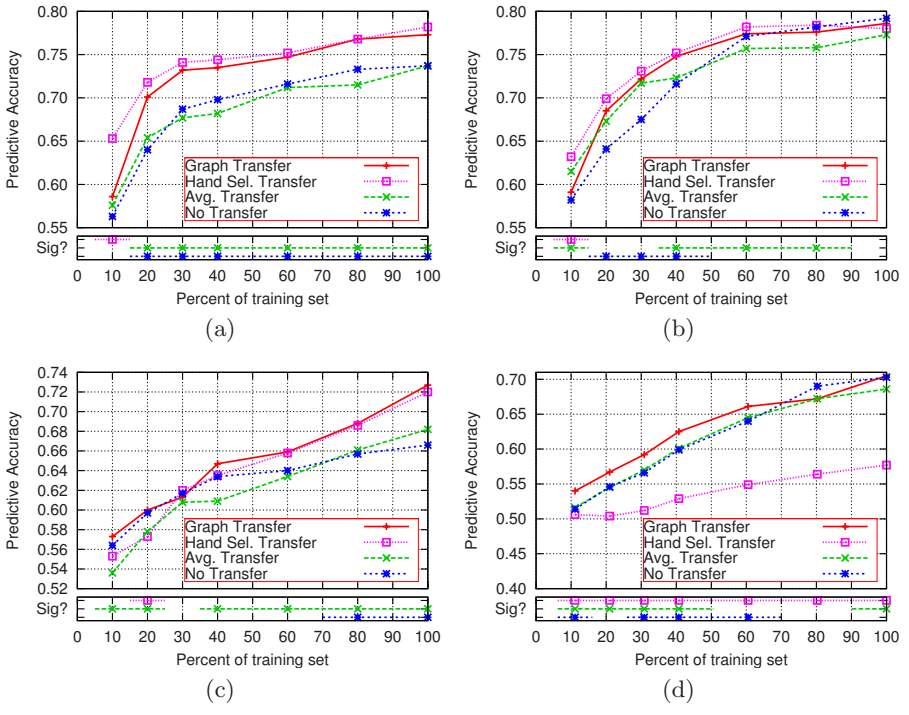


Fig. 2. Results of the Newsgroups transfer scenarios. The bottom portion of each graph depicts the range where our graph transfer approach's performance is statistically different at 95% confidence from each of the other three methods as measured by a pairwise t-test.

transfer. For the letter-H scenario with all relevant source tasks (Fig. 3(e)), graph transfer achieves performance that is statistically indistinguishable from the average parameter transfer, correctly combining information from all source tasks. The transfer scenario in Fig. 3(f) depicts the opposite transfer scenario, with only one source task showing very slight positive transfer to the letter-J task. In this case, it is clear that any transfer would decrease performance, and our graph transfer method shows the best performance of all the transfer methods.

In working with our graph transfer approach, we did observe a few cases where learning without transfer outperformed learning with transfer when given very little training data. However, these situations disappeared when averaged over many trials and folds, as shown in our results. Biased logistic regression relies on the training data to determine the amount of regularization from the given parameter vector. When given very little training data, the learning algorithm's estimation of the ideal amount of transfer may be inaccurate, so it could be outperformed by learning without transfer. It may also be the case that the graph transfer method was occasionally unable to accurately localize the target task

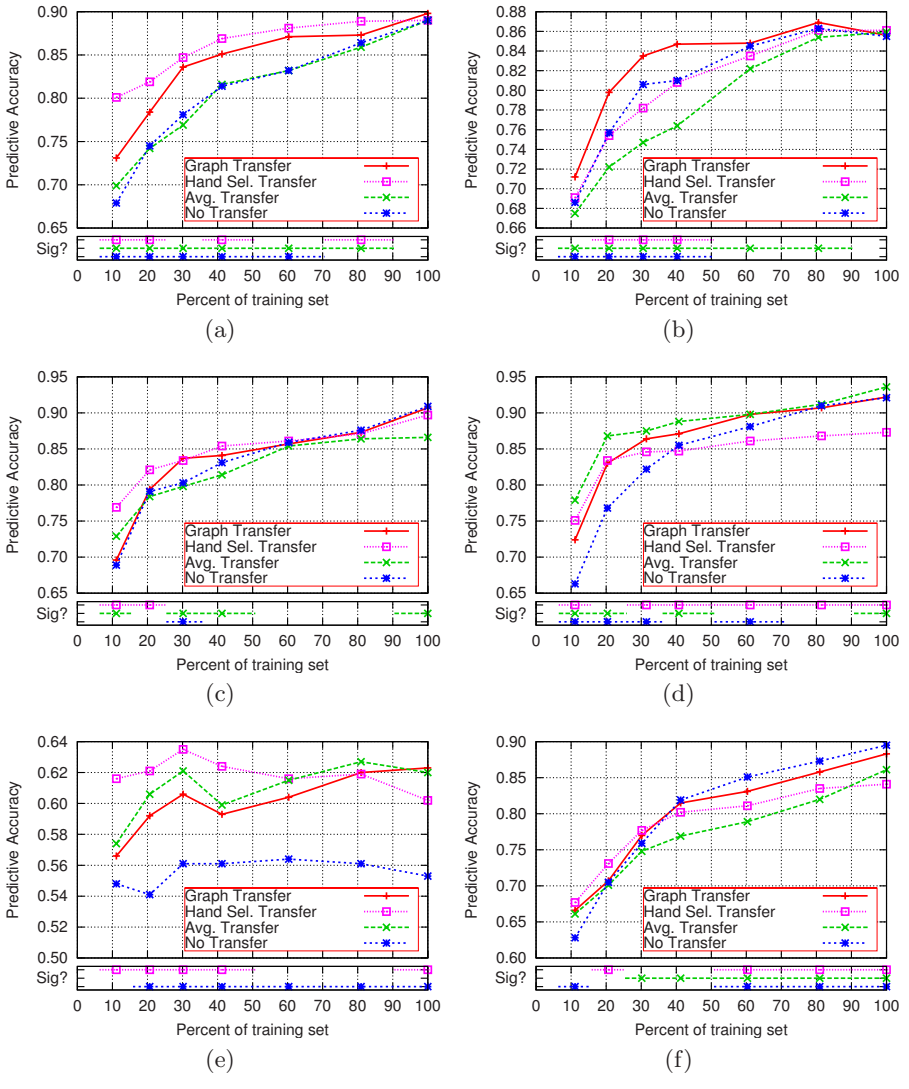


Fig. 3. Results of the Letter transfer scenarios. Figures (e) and (f) depict extreme transfer scenarios: (e) all relevant source tasks on a difficult problem, and (f) no relevant source tasks. For explanation of the lower significance graphs, see the caption to Fig. 2.

in the model transfer graph given very little data. In any case, these hindrances disappeared with the addition of slightly more training data.

We also explored a second transferability measure that was a normalized form of Eqn. 5. This measure defined transferability as the percentage improvement due to transfer against the best possible improvement. In an ideal transfer situation, the learned model's performance would immediately increase to the

maximum possible performance, which may be less than 1 due to noise in the data. The percentage improvement due to transfer would then be the ratio of Eqn. 5 to the maximum possible improvement. Using this normalized transferability measure yielded similar results to those we report here, so we omit these results due to space limitations. In some cases, the normalized transferability measure performed slightly worse than the *un*normalized measure. However, a more thorough analysis involving other transfer scenarios is required to conclude whether the unnormalized measure we use in this paper is truly better.

7 Conclusion and Future Work

This paper describes a novel method for inductive transfer based on modeling the transfer relationships between the source tasks. As shown by our results, using localized estimates of the transfer values results in superior performance on most problems. The shortcut of always using the average parameter vector works well when all of the source tasks are relevant for transfer to the target task, but this involves expensive hand-selection of the source tasks. Additionally, hand-selection relies on qualitative (and sometimes incorrect) judgments that the selected tasks will transfer well to the target task.

We are exploring several extensions to our approach. In this paper, we required transferability to be symmetric between two tasks. However, it has been our experience that often $transfer_{i \to j}$ is much greater than $transfer_{j \to i}$, showing that transfer is not always symmetrical in practice. We plan to extend our method to support directed edges in the transfer graph. Techniques for spectral analysis of directed graphs have only been recently developed [21,22] and using them in this transfer framework presents significant technical challenges that we leave to future work. Additionally, we are conducting a more extensive evaluation of this method, including applying this method to other domains.

Acknowledgments

Eric Eaton's and Marie desJardins' work was supported in part by NSF grant ITR-0325329; Terran Lane's work was supported by NSF grant IIS-0705681. We thank Adam Anthony, Blazej Bulka, and Tim Oates for feedback on this work, and Josh Neil, Eduardo Corona, and Curtis Storlie for discussions on regularization. We also thank the anonymous reviewers for their detailed feedback.

References

1. Marx, Z., Rosenstein, M.T., Kaelbling, L.P., Dietterich, T.G.: Transfer learning with an ensemble of background tasks. In: NIPS 2005 Workshop on Transfer Learning, Whistler, BC, Canada (2005)
2. Kienzle, W., Chellapilla, K.: Personalized handwriting recognition via biased regularization. In: Proceedings of the Twenty-Third International Conference on Machine Learning, Pittsburgh, PA (2006)

3. Thrun, S., O'Sullivan, J.: Clustering learning tasks and the selective cross-task transfer of knowledge. Technical Report CMU-CS-95-209, Carnegie Mellon University, Pittsburgh, PA (November 1995)
4. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: the TC algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 489–497. Morgan Kaufmann, San Francisco (July 1996)
5. Bakker, B., Heskes, T.: Task clustering and gating for Bayesian multitask learning. Machine Learning Research 4, 83–99 (2003)
6. Jordan, M., Jacobs, R.: Hierarchical mixtures of experts and the EM algorithm. Neural Computation 6(2), 181–214 (1994)
7. Pratt, L.Y.: Transferring Previously Learned Back-Propagation Neural Networks to New Learning Tasks. PhD thesis, Rutgers University (June 1993)
8. Mitchell, T.M., Thrun, S.B.: Learning analytically and inductively. In: Mind Matters: A Tribute to Allen Newell, pp. 85–110. Lawrence Erlbaum Associates, Mahwah (1996)
9. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools with Java Implementations. Morgan Kaufmann, San Francisco (2000)
10. Duffy, D.E., Santner, T.J.: On the small sample properties of norm-restricted maximum likelihood estimators for logistic regression models. Communications in Statistics: Theory and Methods 18, 959–980 (1989)
11. Le Cessie, S., Van Houwelingen, J.C.: Ridge estimators in logistic regression. Applied Statistics 41(1), 191–201 (1992)
12. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)
14. Chung, F.R.K.: Spectral Graph Theory. CBMS Regional Conference Series in Mathematics, vol. 92. American Mathematical Society, Providence, RI (1994)
15. Rosenberg, S.: The Laplacian on a Riemannian Manifold. Cambridge University Press, Cambridge (1997)
16. Baker, C.T.H.: The Numerical Treatment of Integral Equations. Clarendon Press, Oxford (1977)
17. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the Nyström method. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(2) (February 2004)
18. Drineas, P., Mahoney, M.W.: On the Nyström method for approximating a Gram matrix for improved kernel-based learning. Journal of Machine Learning Research 6, 2153–2175 (2005)
19. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
20. Rennie, J.: 20 Newsgroups data set, sorted by date (September 2003), `http://www.ai.mit.edu/~jrennie/20Newsgroups/`
21. Chung, F.: Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics 9, 1–19 (2005)
22. Chung, F.: The diameter and Laplacian eigenvalues of directed graphs. Electronic Journal of Combinatorics 13(4) (2006)

Mining Edge-Weighted Call Graphs to Localise Software Bugs

Frank Eichinger, Klemens Böhm, and Matthias Huber

Institute for Program Structures and Data Organisation (IPD),
Universität Karlsruhe (TH), Germany
{eichinger,boehm,huberm}@ipd.uka.de

Abstract. An important problem in software engineering is the automated discovery of noncrashing occasional bugs. In this work we address this problem and show that mining of weighted call graphs of program executions is a promising technique. We mine weighted graphs with a combination of structural and numerical techniques. More specifically, we propose a novel reduction technique for call graphs which introduces edge weights. Then we present an analysis technique for such weighted call graphs based on graph mining and on traditional feature selection schemes. The technique generalises previous graph mining approaches as it allows for an analysis of weights. Our evaluation shows that our approach finds bugs which previous approaches cannot detect so far. Our technique also doubles the precision of finding bugs which existing techniques can already localise in principle.

1 Introduction

Software quality is a big concern in industry. Almost any software displays at least some minor bugs after being released. Such bugs incur significant costs. A class of bugs which is particularly hard to handle is *noncrashing occasional bugs*, i.e., failures which lead to faulty results with some but not with any input data. Noncrashing bugs in general are already hard to find. This is because no stack trace of the failure is available. With occasional bugs, the situation is even more difficult, as they are harder to reproduce. Developers usually try to find and fix bugs by doing an in-depth code review along with testing and classical debugging. Since such reviews are very expensive, there is a need for tools which localise pieces of code that are more likely to contain a bug.

Research in the field of software reliability has been extensive, and various techniques have been developed for locating bugs. Static techniques require a large bug and version history database, which is not always available. Dynamic techniques using instrumentation often have a poor runtime behaviour. Another dynamic technique is the analysis of *call graphs*. Such a graph reflects the invocation structure of a particular program execution. Without any further treatment, a call graph is a *rooted ordered tree*. The `main()` method[1] of a program usually

[1] In this paper, we use *method* interchangeably with *function*.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 333–348, 2008.

is its root, and all methods invoked directly are its children. Figure 1(a) is an abstract example of such a call graph. Recent work [1, 2] deploys *graph mining* techniques on call graphs for bug localisation. [2] then derives a ranking of methods which are most probable to contain a bug. Generating such a ranking is not trivial. For instance, follow up bugs need to be identified. [2] does not identify follow up bugs at all, and [1] only generates a backtrace-like structure which helps the programmer to find follow up bugs.

Graph mining is a relatively new discipline in data mining, and innovative algorithms have been developed in recent years [3, 4, 5, 6]. Various algorithms deal with the problem of *mining frequent subgraphs*, i.e., discovering all subgraphs which are frequent in a set of graphs. A difficulty with graph mining on raw call graphs is that the algorithms do not scale. Therefore, reduction techniques are developed and applied first. Such techniques are not obvious: They involve a trade-off between loss of information and the size of the resulting graphs. An important piece of information included in the raw call graphs is the call frequency of all methods. The reduction techniques in [1, 2] loose this information. But it eases detection of bugs which affect the number of invocations of a method, *call frequency affecting bugs*. Various reasons for such bugs exist, e.g., wrongly specified conditions. Note that it is not only loop conditions which cause these bugs, but any wrongly specified condition leading to method calls within in a loop. This is because branches taken within a loop or not affect the frequency of a certain method call. As iterations are elementary in programming languages, a wide range of bugs is call frequency affecting. To find such bugs, we take call frequencies into account and analyse their differences in correct and failing executions.

Graph mining research has focused on structural and categorical techniques, and the various graph miners available target at different kinds of graphs. Almost all algorithms handle categorical data in node and edge labels. However, little attention has gone into the analysis of quantitative information, and no algorithm is available for mining *weighted graphs*. Since we want to analyse call graphs where weighted edges represent call frequencies, we must come up with a solution. Further, finding a suitable combination of call graph reduction, graph mining and the analysis of call frequencies is challenging.

In this work, we use conventional mining techniques for unweighted graphs in conjunction with feature selection algorithms to analyse numerical edge weights. More specifically, we first trace program executions and classify them as correct or failing using a test oracle. We for our part do this by comparing execution results to a fault-free reference. These correct results are typically available, as test suites providing such information are widely used in quality assurance [7]. We represent the program traces as call graphs and reduce these graphs by deleting multiple method calls caused by iterations and introducing edge weights representing call frequencies. We then mine the reduced graphs before taking the edge weights into account: By applying an entropy based feature selection algorithm to the weights of the different edges, we calculate the likelihood of both every method invocation as well as of every method containing a bug. For a final

ranking, we combine these likelihoods with another score based on structural properties of the graph mining results. The rationale is that this ranking is given to a software developer who can do a code review of the suspicious methods.

In other words, solving the problems mentioned so far requires innovations at different levels of analysis. Our contributions are as follows:

Reduction of software call graphs. We propose using a new variant of reduced call graphs and present a technique to accomplish this reduction. The reduced graphs keep much information, while they are relatively small. For example, [2] would reduce one call graph from our evaluation from 9,946 to 50 edges. With our technique, there are just 31 edges, while keeping more information and, ultimately, giving way to better results.

Mining weighted graphs. We present an approach which combines numerical analysis of edge weights with conventional graph mining. Our approach distinguishes between occurrences of edges which appear more than once within a subgraph. This allows for a detailed analysis. To our knowledge, a technique which analyses weights in the postprocessing of graph mining has not been described before. This particular contribution is not limited to call graph analysis, but can be transferred to any domain where weighted graphs are present.

Combination of numerical and structural techniques. There exist frequent subgraphs which occur both in call graphs of correct and of failing executions, as well as frequent subgraphs occurring in failing executions only. We analyse the edge weights for subgraphs from the first class, whereas we generate purely structural evidence from subgraphs from the second class. Our approach then combines these different kinds of evidence, and we demonstrate its usefulness.

Evaluation. We show that our approach is particularly well suited to discover call frequency affecting bugs. Unlike previous techniques, it also detects follow up bugs. With regard to bugs existing techniques can already localise, our approach doubles the precision of finding them. Furthermore, it finds bugs on the granularity of method invocations instead of the level of methods.

In this work, we do not reduce recursive calls and concentrate on iterations. The reduction of recursions is not obvious and is beyond the scope of this study.

Paper outline: Section 2 reviews related work. Section 3 presents an overview of graph mining with call graphs. Section 4 discusses graph reduction techniques. Section 5 describes how to calculate probabilities of containing bugs based on reduced graphs. Section 6 features an evaluation. Section 7 concludes.

2 Related Work

A lot of research has been done in the field of software reliability. Approaches range from static code analysis and mining of software repositories and bug databases [8, 9, 10] to dynamic program verification. The latter focus on the data flow [11, 12] or, like all call graph based techniques, on the control flow [13, 14]. In the following, we will first discuss the application of data mining techniques

in this context – bug localisation is just one application. Then we concentrate on two graph mining based approaches [1, 2] which are most related to our work. Finally, we describe some related work in the area of mining weighted structures.

2.1 Mining Software Metrics and Invariants

[8] maps post-release failures from a bug database to defects in static source code. Using standard complexity measures from software engineering, the source code is mined with regression models, which can then predict post-release failures for new software entities. A similar study uses decision trees to predict failure probabilities [9]. [10] uses regression techniques to predict the likelihood of bugs based on static usage relationships between software components. All approaches mentioned require a large collection of bugs and version history.

Dynamic program slicing [11] gives hints which parts of a program might have contributed to a faulty execution, without ranking the locations in question. This is done by discovering all statements that actually affect the variables involved. Advanced techniques like [12] perform dynamic data flow focused analysis by instrumenting the source code to gain program invariants. These are used as features of correct and failing executions which are analysed with regression techniques. This leads to potentially faulty pieces of code. A similar approach, but with a focus on the control flow, is [13]. It instruments condition statements and calculates a ranking based on its evaluation frequencies. The instrumentation based approaches mentioned either suffer from poor runtime behaviour or miss bugs if only sampled parts of the software are instrumented.

[14] is a technique which uses tracing and visualisation. It relies on a simple ranking of program components based on the information which components are executed more often in failing program executions. This ranking serves as a basis for more sophisticated rankings in [2] as well as in our approach.

2.2 Call Graph Based Fault Detection

The approach from Liu et al. [1]: This first study which applies graph mining techniques to dynamic call graphs considers so called *software behaviour graphs*. These are reduced call graphs, augmented with some temporal information. Section 4 will provide more information on these graphs. The behaviour graphs representing correct and failing program executions are mined with a variant of the *CloseGraph* algorithm [5]. This step results in frequent subgraphs which are used as binary features characterising a program execution: A boolean feature vector represents every execution. In this vector, every element indicates if a certain subgraph is included in the corresponding behaviour graph. Using those feature vectors, a support vector machine is learned which decides if a program execution is correct or failing. More precisely, for every method, two classifiers are learned: one based on behaviour graphs including the respective method, one based on graphs without it. If the precision rises significantly when adding graphs containing a certain method, this method is deemed more likely to contain a bug. Experiments with five out of 130 bugs from the Siemens Programs [15]

demonstrate good classification performance, but do not evaluate the precision of the bug localisation. Furthermore, the authors do not generate a ranking of methods suspected to contain a bug. As we do so, our approach can not be compared directly. However, in Sect. 6, we compare the reduction techniques.

The approach from Di Fatta et al. [2]: In this work, a reduction technique is again applied to the raw call trees first (see Sect. 4 for details). The next step is similar to the one described before: A collection of reduced call graphs representing correct and failing program executions is analysed with graph mining. The authors use the tree miner *FREQT* [16] to find all frequent subtrees. The call trees analysed are large and lead to scalability problems of the algorithm. Hence the authors limit the size of the subtrees searched to values up to 4. Then the authors identify which resulting subgraphs are frequent in the set of failing program executions, but not frequent in the set of correct ones. This set of subgraphs is called *specific neighbourhood*. For all methods invoked within subgraphs which are part of the specific neighbourhood, a probability of containing a bug is calculated based on support values. Like [1], [2] does not put attention on call frequencies.

2.3 Subsumption

Frequent subgraph mining is a generalisation of previous structural knowledge discovery techniques such as mining of itemsets, sequences and trees [17]. Early work [18] in the area of itemsets has introduced the problem of *weighted structure mining*. Itemsets can be seen as graphs consisting of nodes only. In [18], these nodes are weighted and are discretised during preprocessing. Corresponding techniques are applied to graphs in transportation networks [19] and image analysis [20]. Such discretisation leads to a loss of information, as we will discuss in Sect. 5.2. In [21], we have already analysed tuples of weights of sequences as a postprocessing step of sequence mining. This allows for a more detailed analysis of weights in different structural contexts. This current work is in the field of weighted structure mining as well – it analyses edge weights subsequent to a graph mining step.

[22] is a preliminary and much shorter version of this paper. It directly combines its results with those of [2] in order to find a wider range of bugs. It requires two costly graph mining steps. This current work avoids this.

3 Call Graph Mining Overview

Before we focus on reduction and ranking techniques in Sections 4 and 5, we now give an overview of the procedure of localising bugs with graph mining. Note that [2] follows this general procedure as well. Algorithm 1 first assigns a class (*correct*, *failing*) to every program trace (Line 4), using a test oracle. Then every trace is reduced (Line 5), which leads to smaller call graphs. Techniques to do so are discussed in Sect. 4. Now frequent subgraphs are mined (Line 7). For this step, several algorithms, e.g., tree mining or graph mining in different variants,

Algorithm 1. Generic graph mining based bug localisation procedure

1: **Input:** a collection of program traces $t \in T$
2: $G = \emptyset$ // initialise a collection of reduced graphs
3: **for all** traces $t \in T$ **do**
4: assign a $class \in \{correct, failing\}$ to t
5: $G = G \cup \{reduce(t)\}$
6: **end for**
7: $SG = frequent\_subgraph\_mining(G)$
8: calculate $P(m)$ for all methods m; based on SG

can be used. The last step is calculating a likelihood of containing a bug. This can either be fine granular for every method invocation or, more coarse grained, for every method (as shown in Line 8). The calculation of the likelihood is based on the frequent subgraphs mined and facilitates a ranking of the methods, which can then be given to the software developer.

4 Call Graph Reduction

In the related work on call graph mining (see Sect. 2.2), two different approaches exist which lead to reduced call graphs (as specified in Line 5 of Algorithm 1). The reduction technique used in [1] projects every node representing the same method in the call graph to a single node in the reduced graph. We call this technique *total reduction*. Note that this may give way to the existence of loops and limits the size of the graph (in terms of nodes) to the total number of methods in the program analysed. As an example, the raw ordered call tree in Fig. 1(a) would result in the reduced graph displayed in Fig. 1(b). In addition to the reduction, so called *temporal edges* are inserted between all methods which are executed consecutively and are invoked by the same method. This technique integrates the temporal order from the raw ordered call trees into the graph representations. Technically, temporal edges are directed edges having another label, e.g., "temporal", compared to other edges which are labelled, say, "call". Figure 1(c) serves as an example of a graph using the reduction technique and temporal edges (dotted), called *software behaviour graph*. This reduction is rather severe, e.g., from several millions of nodes to several hundreds. It allows graph

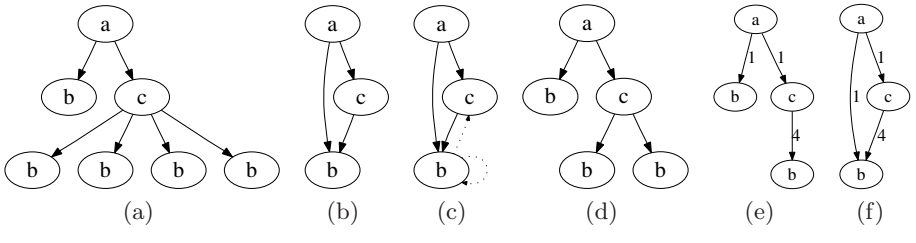


Fig. 1. Variants of reduced call graphs

mining based bug localisation even with larger software projects. In contrast, much information about the program execution is lost. This concerns frequencies of the execution of methods as well as information on different structural patterns within the graphs. In particular, the information in which context a certain substructure is executed is lost (see Sect. 5.2). Furthermore, the temporal edges increase the size of the graphs significantly. An increased precision of fault detection by using temporal edges has not been evaluated.

The approach in [2] keeps more information. It omits substructures of subsequent executions, which are invoked more than twice in a row from the same node. See Fig. 1(d) for an example. This reduction ensures that many equal substructures called within a loop do not lead to call graphs of an extreme size. In contrast, the information that some substructure is executed several times is still encoded in the graph structure, but without exact numbers. Compared to [1], much more information about a program execution is kept, compromised by a call graph which is generally much larger. For example, graphs are reduced from several millions of nodes to several ten thousand nodes.

In our approach, we try to overcome the shortcomings of both approaches and try to keep most of the information available. We reduce subtrees executed iteratively by deleting all but the first one and inserting the call frequencies as edge weights. This makes the graphs relatively small and keeps a lot of information. An example is given in Fig. 1(e). The introduction of edge weights allows for a detailed analysis. If, e.g., a bug is hidden in a loop condition, this might lead to hundreds of iterations of the loop, compared to just a few in correct program executions. Note that, with both previous graph representations, the graph of the correct and of the failing execution is reduced to exactly the same structure in this case. In our approach, the edge weights would be significantly different. Analysis techniques can then discover this.

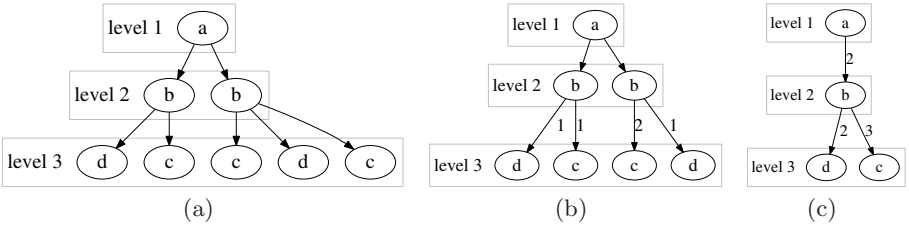


Fig. 2. A raw call tree, its first and second transformation step

For our graph reduction approach, which implements the functionality specified in Line 5 of Algorithm 1, we organise the call tree into n horizontal levels. The root node is in *level* 1, all other nodes are in subsequent levels, increasing with the distance to the root. See Fig. 2(a). A naïve approach to reduce our example call tree in Fig. 2(a) would be to start at *level* 1 with Node a. There, one would find two child subtrees having a different structure – one could not merge anything. Therefore, we work level by level, starting from *level* $n-1$, as

Algorithm 2. Subtree reduction algorithm

```
1: Input: a call tree organised in n levels
2: for level = n − 1 to 1 do
3:    for each node in level do
4:       merge all identical child-subtrees of node, sum up corresponding edge weights
5:    end for
6: end for
```

described in Algorithm 2. In our example in Fig. 2(a), we have to start in *level* 2. The left node b has two different children. Thus, nothing can be merged there. In the right b, the two children c are merged by adding the edge weights of the merged edges, yielding the tree in Fig. 2(b). In the next level, *level* 1, we process the root node a. Here, the structure of the two succeeding subtrees is the same. We merge them, resulting in the tree in Fig. 2(c).

Our reduction technique obviously is not lossless. The size of the resulting graphs would be prohibitive. With large pieces of software, graph mining may not scale with the size of the call graphs, even if the reduction technique applied is effective. To deal with this problem, it seems promising to use graphs of coarser granularity: Instead of using methods as nodes in call graphs, it is possible to, say, use classes as a coarser abstraction. Such call graphs would have classes as nodes and inter class method calls as edges. Obviously, such a coarser abstraction would lead to less detailed bug localisation as well.

5 The Ranking Framework

So far, we have discussed how to reduce call graphs. Now we will describe our framework for deriving a ranking of potentially buggy method invocations (edges) and methods (nodes) from such graphs. Before we focus on the individual components in the following subsections, we give an overview of our framework. At first, we apply frequent subgraph mining to the reduced call graphs, without considering the weights for now (Sect. 5.1). This corresponds to Line 7 in Algorithm 1. We then partition the set of frequent subgraphs just mined and consider two sets separately: (1) the set of subgraphs which occur in both correct and failing executions SG_{cf} and (2) the set of subgraphs which only occur in failing executions SG_f[2]. We use SG_{cf} to build a ranking based on the differences in edge weights in correct and failing executions (Sect. 5.2). This ranking can be based on method invocations or on individual methods. As subgraphs in SG_f are never contained in correct executions, it is not possible to analyse differences based on SG_f. In contrast, SG_f provides crucial information about the graph structures in failing executions. Therefore, we derive a score based on the support in SG_f (Sect. 5.3) and combine it with the edge weight based one mentioned above (Sect. 5.4). This combination is in Line 8 in Algorithm 1.

[2] In preliminary experiments, we have also evaluated the influence of subgraphs which occur in correct executions only. It has turned out that such graphs do not help to localise bugs.

5.1 Graph Mining Step

After having reduced the call graphs gained from correct and failing program executions, we search for frequent closed subgraphs SG in the graph dataset G using the *CloseGraph* algorithm [5]. For this step, we employ the *ParSeMiS* graph mining suite[3]. Closed mining reduces the number of graphs in the result set significantly and increases the performance of the mining algorithm (studying its effects on result quality is beyond the scope of this article). Furthermore, the usage of a general graph mining algorithm instead of a tree miner allows for comparative experiments with other graph reduction techniques (see Sect. 6). After the graph mining step, we partition SG and derive the set of subgraphs which occur in correct and failing executions SG_{cf} and the set of subgraphs which occur in failing executions only SG_f.

5.2 Entropy Based Scoring

We now focus on frequent subgraphs which occur in both correct and failing executions (SG_{cf}). Our goal is to develop an approach which discovers which edge weights of call graphs from a program are most significant to discriminate between *correct* and *failing*. To do so, one possibility is to consider different *edge types*, e.g., edges having the same calling method m_s (start) and the same method called m_e (end). However, edges of one type can appear more than once within one subgraph and, of course, in several different subgraphs. Therefore, we analyse every edge in every such location, which we refer to as a *context*. To specify the exact location of an edge in its context within a certain subgraph, we do not use the method names, as they may occur more than once. Instead, we use a unique id for the calling node (id_s) and another one for the method called (id_e). All ids are valid within its subgraph. To sum up, we reference an edge in its context in a certain subgraph sg with the following tuple: (sg, id_s, id_e). A certain bug does not affect all method calls (edges) of the same type, but method calls of the same type in the same context. Therefore, we assemble a feature table with every edge in every context as columns and all program executions (represented by their reduced call graphs) as rows. The table cells contain the respective edge weights. The following table serves as an example:

	$a \to b$ (sg_1, id_1, id_2)	$a \to b$ (sg_1, id_1, id_3)	$a \to c$ (sg_2, id_1, id_2)	$\cdots$	Class
g_1	445	21	7	$\cdots$	*failing*
g_2	0	0	4	$\cdots$	*correct*
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

The first column corresponds to the first subgraph (sg_1) and the edge from id_1 (method a) to id_2 (method b). The second column corresponds to the same subgraph (sg_1) but to the edge from id_1 (method a) to id_3 (method b). The third column represents an edge from id_1 to id_2 in the second subgraph (sg_2). Note

[3] `http://www2.informatik.uni-erlangen.de/Forschung/Projekte/ParSeMiS/`

that method b occurs twice in sg_1 and that ids have different meanings in sg_1 and sg_2. The last column contains the class *correct* or *failing*. The rows correspond to all reduced call graphs $g_1, ..., g_n \in G$ available. If a certain subgraph is not contained in a call graph, the corresponding cells have value 0, like g_2 which does not contain sg_1. Graphs (rows) can contain a certain subgraph not just once, but several times at different locations. In this case, we use aggregates in the corresponding cells of the table. As minimum values would ignore bugs resulting in increased numbers and maximum values would ignore bugs leading to reduced numbers, we use the average. In the example, sg_2 is embedded at two locations in g_1. In one location the edge from id_1 to id_2 has the weight 6, in the other one weight 8.

The table structure described allows for a detailed analysis of edge weights in different contexts within a subgraph. All following steps in this subsection are described in Algorithm 3. After assembling the table, we employ a standard feature selection algorithm to score the columns of the table and thus the different edges. We use an entropy based algorithm from the *Weka* data mining suite [23] which calculates the information gain *InfoGain* [24] (with respect to the class of the executions, *correct* or *failing*) for every column (Line 2 in Algorithm 3). The information gain is a value between 0 and 1 which we interpret as a likelihood of being responsible for bugs. Columns with an information gain of 0, e.g., the edges always have the same weights in both classes, are discarded immediately (Line 3 in Algorithm 3).

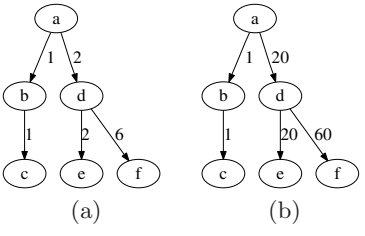


Fig. 3. Follow up bugs

Call graphs of failing executions frequently contain bug-like patterns which are caused by a preceding bug. We call such patterns *follow up bugs* and remove them from our ranked list of features. Figure 3 illustrates a follow up bug: (a) represents a bug free version, (b) contains a bug in method a where it calls method d. Here, this method is called 20 times instead of twice. Following our reduction technique, this leads to the same (or a proportional) increase in the number of calls in method d. In our entropy based ranking, the edges $d \rightarrow e$ and $d \rightarrow f$ inherit the score from $a \rightarrow d$ if the scaling of the weights is proportional. Thus, we interpret these two edges as follow up bugs and remove them from our ranking. More formally, we remove edges if the edge leading to its direct parent within the same subgraph has the same entropy score (Line 4 in Algorithm 3). In case of more than one bug in a program, this way of follow up bug detection might

Algorithm 3. Procedure to calculate $P_e(m_s, m_e)$ and $P_e(m)$

1: **Input:** a set of edges $e \in E$ representing edges in their context, $e = (sg, id_s, id_e)$
2: assign every $e \in E$ its information gain *InfoGain*
3: $E = E \setminus \{e \mid e.InfoGain = 0\}$
4: // remove follow up bugs:
$E = E \setminus \{e \mid \exists p : p \in E, p.sg = e.sg, p.id_e = e.id_s, p.InfoGain = e.InfoGain\}$
5: $E_{(m_s,m_e)} = \{e \mid e \in E \wedge e.id_s.label = m_s \wedge e.id_e.label = m_e\}$
6: $P_e(m_s, m_e) = \max_{e \in E_{(m_s,m_e)}} (e.InfoGain)$
7: $E_m = \{e \mid e \in E \wedge e.id_s.label = m\}$
8: $P_e(m) = \max_{e \in E_m} (e.InfoGain)$

not find all such bugs, but preliminary experiments have shown that it does detect common cases efficiently. We leave aside the pathological case that this technique classifies a real bug as follow up bug. This is acceptable, since the probability of a certain entropy value is the same for every bug. Therefore, it is very unlikely that two unrelated bugs lead to exactly the same entropy value, which would lead to a 'false positive' classification.

Now we calculate likelihoods of containing a bug for every method invocation (described by a calling method m_s and a method called m_e). We call this score $P_e(m_s, m_e)$ as it is based on entropy. To do so, we first determine sets $E_{(m_s,m_e)}$ of edges $e \in E$ for every method invocation in Line 5 of Algorithm 3. In Line 6, we use the max() function to calculate $P_e(m_s, m_e)$, the maximum of all edges (method invocations) in $E_{(m_s,m_e)}$. This is necessary, as in general there are many edges in E with the same method invocation. This is because an invocation can occur in different contexts. With the max() function, we assign every invocation the score from the context ranked highest. Lower scores for the same invocation are less important, and we ignore them.

At this point, the ranking does not only provide the score for a method invocation, but also the subgraphs where it occurs and the locations within it. This information might be important for a software developer. We report this information additionally. As we also want to compare our results to those of [2] which does not provide information on the invocation level, we also calculate $P_e(m)$ for every calling method m in Lines 7 and 8 of Algorithm 3. The explanation is analogous to the one of the calculation of $P_e(m_s, m_e)$ in Lines 5 and 6.

This subsection has presented a technique relying on the analysis of edge weights in different contexts. As we will see in the evaluation (Sect. 6), the consideration of different contexts is key for good results. As contexts are defined based on the subgraphs mined, such a differentiated analysis is only possible subsequent to graph mining, but not during preprocessing (see Sect. 2.3).

5.3 Structural Scoring

Our entropy based scoring (Sect. 5.2) cannot detect bugs which are not call frequency affecting, as it analyses call frequencies only. At the same time, it

does not consider subgraphs which occur in failing executions only (SG_f). As some bugs result in such subgraphs, these are essential to detect bugs as well. Therefore, we calculate the score $P_s(m)$ for individual methods based on the support in SG_f. This score is another likelihood of containing a bug, as it refers to the frequency of method invocations in failing executions.

$$P_s(m) = supp(m, SG_f) \tag{1}$$

where $supp(m, SG_f)$ is the fraction of graphs in SG_f containing a node m.

5.4 Combination

Now we calculate the overall likelihood $P(m)$ of containing a bug for every method m, based on the average of the normalised values for $P_e(m)$ (see Sect. 5.2) and $P_s(m)$ (see Sect. 5.3). Normalisation is necessary: While both values are in the [0, 1]-range, their maximum can be very different. Normalisation keeps us from overemphasising one of the two rankings. $P(m)$ is the basis for the ranking of all methods m, which is used to locate bugs:

$$P(m) = \frac{P_e(m)}{2 \max\limits_{n \in t \in T}(P_e(n))} + \frac{P_s(m)}{2 \max\limits_{n \in t \in T}(P_s(n))} \tag{2}$$

where n is a method in a program trace t in the collection of all traces T.

6 Evaluation

To evaluate bug localisation techniques, the *Siemens Programs* [15] are often used [1, 2, 13] as a reference suite of C programs artificially instrumented with different bugs. More specifically, it usually is just a small subset of this benchmark which is used. For example, [2] just considers three of the seven Siemens Programs, [1] only five different bugs out of 130 available in total. As most bug detection techniques are limited to certain classes of bugs, these techniques cannot find every element of a standard suite of bugs. Our approach, as well as the two most related ones [1, 2], focus on noncrashing occasional bugs.

As we rely on Java software, we use a well known Java diff tool (taken from [25]) and instrument it with fourteen different bugs[4]. To do so, we have examined the Siemens Programs and have identified five types of bugs which are most frequent within them. Our programs contain these bugs and also the kinds of bugs used in [1]. Most of these bugs are call frequency affecting. The Siemens Programs mostly contain bugs in single lines and just a few programs with more than one bug. To mimic the Siemens Programs as close as possible, we have instrumented only two out of fourteen versions (Bugs 7 and 8) with more than one bug. We give an overview of the kinds of bugs used in the following table:

[4] We provide the software versions containing the bugs used at `http://www.ipd.uka.de/~eichi/papers/eichinger08mining/`

Version	Description
Bug 1, Bug 10	Wrong variable used
Bug 2, Bug 11	Additional or-condition
Bug 3	`>=` instead of `!=`
Bug 4, Bug 12	`i+1` instead of `i` in array access
Bug 5, Bug 13	`>=` instead of `>`
Bug 6	`>` instead of `<`
Bug 7	A combination of Bug 2 and Bug 4 (in the same line)
Bug 8	`i+1` instead of `i` in array access + additional or condition
Bug 9, Bug 14	Missing condition

Every version has been executed exactly 100 times with different input data. The results have been classified as correct or failing executions with a test oracle based on a bug free reference version. Based on this data, we carry out four experiments:

1. The *conventional method ranking*, following [2], including its graph reduction technique, using the same method scoring and the same mining parameters for support and maximum subgraph size.
2. The *total reduction method ranking*, the total reduction from [1] with edge weights representing call frequencies (see Fig. 1(f)) together with the combined scoring (Sect. 5.4).
3. The *entropy-based method ranking*, our reduction technique (Sect. 4) together with the entropy based scoring (Sect. 5.2) but without the combination.
4. The *combined method ranking* (Sect. 5.4), our reduction technique (Sect. 4) together with the entropy based scoring (Sect. 5.2) and the structural scoring (Sect. 5.3).

To keep the comparison focused, we leave aside the temporal order inside the call graphs. We have found graphs with temporal edges as used in [1] too large (in terms of edges) to be mined efficiently. Nevertheless, our study is fair since we leave aside that temporal information with all alternatives.

All experiments produce ordered lists of methods. A software developer doing a code review would start with the top ranked method in such a list. The maximum number of methods which have to be checked to find the bug is, therefore, the line number of the faulty method in the ranked list. Sometimes two or more subsequent lines have the same score. As the intuition is to count the maximum number of methods which have to be checked, all lines with the same score have the number of the last line with this score.

In order to evaluate the accuracy of the results, the line of the ranking where the first instrumented bug is found needs to be identified. If the first instrumented bug is, e.g., reported in the third line, this is a fairly good result. A software developer only has to do a code review of maximally three out of 25 methods from our target program. Furthermore, in the entropy-based and the combined method ranking, there usually is more information available where a bug is located within a method and in which context it appears. Thus, our comparison is conservative, i.e., it does not demonstrate the full capabilities of our approach.

We present the results (the number of the first line in which a bug is found) of the four experiments for all fourteen bugs in the following table. For versions with two bugs, we concentrate on finding the first of the two bugs contained. Value 25 refers to a bug which is not discovered with the respective approach.

Experiment \ Bug Version Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Conventional method ranking (1)	3	25	1	4	6	4	3	3	1	6	4	4	25	4
Total reduction method ranking (2)	1	5	1	4	3	5	5	2	25	2	5	4	6	3
Entropy-based method ranking (3)	3	3	1	1	1	3	3	1	25	2	3	3	3	3
Combined method ranking (4)	3	2	1	1	1	2	2	1	8	2	2	3	3	3

Comparing the results from (1) and (3), the entropy-based approach almost always performs as good or better than the conventional one. This shows that analysing numerical call frequencies is adequate to locate bugs. Bugs 2, 9 and 13 illustrate that both approaches alone cannot find certain bugs. Bug 9 can not be found by comparing call frequencies (3), as a condition has been modified which now always leads to the invocation of a certain method. Therefore, the call frequency is always the same (not call frequency affecting). Bugs 2 and 13 are not found with the purely structural conventional approach (1). Both are typical call frequency affecting bugs: Bug 2 is in a condition inside a loop and leads to more invocations of a certain method. In Bug 13, a modified for-condition slightly changes the call frequency of a method inside the loop. With the reduction technique used in (1), this leads in Bug 2 and Bug 13 to the same graph structure both with correct and with failing executions. Thus, it is not possible to identify structural differences.

To ease presentation, we first describe the combined approach (4) before we discuss (2). Experiment (4) is intended to take important structural information into account as well and therefore to improve the results from (3). We do achieve this goal: We retain the already good results from (3) in nine cases and improve them in five. In particular, (4) finds Bug 9, which is not possible with (3) alone. Therefore, the combination of numerical and structural techniques turns out to be superior. When calculating averages of the improvement of certain approaches in the following, we leave aside Bugs 2, 9 and 13. This is because high values would have a too strong influence on the results. Note that this lets our approach look somewhat worse, as it does find all three bugs. Bugs 7 and 8 illustrate that our approach analyses versions with more than one bug successfully as well. In (4), both bugs are ranked at Position 1 and 2. This is not worse than versions with one bug only.

Experiment (2) is intended to evaluate the graph reduction technique in [1]. Except for the graph reduction technique, the approach is identical to the one in (4). In almost all cases, (2) performs worse than (4). This confirms that our graph reduction technique is reasonable and that it is worth to keep more structural information than the total reduction does.

Summing up, our experiments show that weighted graph mining on call graphs reduced with our method is appropriate for precise software bug localisation. Even if previous approaches are able to detect call frequency affecting bugs, our

approach can detect them with a much higher degree of precision. This is because it explicitly analyses call frequencies. Furthermore, our approach finds bugs in settings with more than one bug and doubles the precision of [2] on average.

7 Conclusions

In this work we have addressed the problem of localising noncrashing occasional software bugs. This localisation is important as such bugs are hard to detect manually and cause significant costs. Our approach is dynamic and control flow centred as it relies on call graphs of program executions. We have presented a novel technique to reduce such graphs. It keeps the size of the resulting graphs relatively small while keeping more important information. In particular, it introduces edge weights representing call frequencies. As none of the recently developed graph mining algorithms analyse weighted graphs, we have developed a combined approach which does so. It consists of conventional frequent subgraph mining and subsequently scoring of numerical edge weights using an entropy based algorithm. Our experiments do not just show a doubled precision of bug localisation. They also show that our approach detects bugs which previous approaches can not find in principle. We demonstrate that the numerical information kept with our call graph reduction technique is important for good results. We have shown that our combination of structural and numerical mining techniques is key for precise localisations.

Future work will address recursive method invocations. Another direction is mining of call graphs with constraint based and approximative techniques.

References

[1] Liu, C., Yan, X., Yu, H., Han, J., Yu, P.S.: Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs. In: Proc. of the 5th Int. Conf. on Data Mining (SDM) (2005)

[2] Di Fatta, G., Leue, S., Stegantova, E.: Discriminative Pattern Mining in Software Fault Detection. In: Proc. of the 3rd Int. Workshop on Software Quality Assurance (SOQUA) (2006)

[3] Borgelt, C., Berthold, M.R.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In: Proc. of the 2nd Int. Conf. on Data Mining (ICDM) (2002)

[4] Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. of the 2nd Int. Conf. on Data Mining (ICDM) (2002)

[5] Yan, X., Han, J.: CloseGraph: Mining Closed Frequent Graph Patterns. In: Proc. of the 9th Int. Conf. on Knowledge Discovery and Data Mining (KDD) (2003)

[6] Nijssen, S., Kok, J.N.: A Quickstart in Frequent Structure Mining Can Make a Difference. In: Proc. of the 10th Int. Conf. on Knowledge Discovery and Data Mining (KDD) (2004)

[7] Harrold, M.J., Gupta, R., Soffa, M.L.: A Methodology for Controlling the Size of a Test Suite. ACM Transactions on Software Engineering and Methodology (TOSEM) 2(3), 270–285 (1993)

[8] Nagappan, N., Ball, T., Zeller, A.: Mining Metrics to Predict Component Failures. In: Proc. of the 28th Int. Conf. on Software Engineering (ICSE) (2006)
[9] Knab, P., Pinzger, M., Bernstein, A.: Predicting Defect Densities in Source Code Files with Decision Tree Learners. In: Proc. of the Int. Workshop on Mining Software Repositories (MSR) at ICSE (2006)
[10] Schröter, A., Zimmermann, T., Zeller, A.: Predicting Component Failures at Design Time. In: Proc. of the 5th Int. Symposium on Empirical Software Engineering (2006)
[11] Korel, B., Laski, J.: Dynamic Program Slicing. Information Processing Letters 29(3), 155–163 (1988)
[12] Liblit, B., Aiken, A., Zheng, A.X., Jordan, M.I.: Bug Isolation via Remote Program Sampling. ACM SIGPLAN Notices 38(5), 141–154 (2003)
[13] Liu, C., Yan, X., Han, J.: Mining Control Flow Abnormality for Logic Error Isolation. In: Proc. of the 6th Int. Conf. on Data Mining (SDM) (2006)
[14] Eagan, J., Harrold, M.J., Jones, J.A., Stasko, J.: Technical Note: Visually Encoding Program Test Information to Find Faults in Software. In: Proc. of the Symposium on Information Visualization (INFOVIS) (2001)
[15] Hutchins, M., Foster, H., Goradia, T., Ostrand, T.: Experiments on the Effectiveness of Dataflow- and Controlflow-Based Test Adequacy Criteria. In: Proc. of the 16th Int. Conf. on Software Engineering (ICSE) (1994)
[16] Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., Arikawa, S.: Efficient Substructure Discovery from Large Semi-structured Data. In: Proc. of the 2nd Int. Conf. on Data Mining (SDM) (2002)
[17] Han, J., Cheng, H., Xin, D., Yan, X.: Frequent Pattern Mining: Current Status and Future Directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)
[18] Srikant, R., Agrawal, R.: Mining Quantitative Association Rules in Large Relational Tables. In: Proc. of the Int. Conf. on Management of Data (SIGMOD) (1996)
[19] Jiang, W., Vaidya, J., Balaporia, Z., Clifton, C., Banich, B.: Knowledge Discovery from Transportation Network Data. In: Proc. of the 21st Int. Conf. on Data Engineering (ICDE) (2005)
[20] Nowozin, S., Tsuda, K., Uno, T., Kudo, T., Bakir, G.: Weighted Substructure Mining for Image Analysis. In: Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR) (2007)
[21] Eichinger, F., Nauck, D.D., Klawonn, F.: Sequence Mining for Customer Behaviour Predictions in Telecommunications. In: Proc. of the Workshop on Practical Data Mining at ECML/PKDD (2006)
[22] Eichinger, F., Böhm, K., Huber, M.: Improved Software Fault Detection with Graph Mining. In: Proceedings of the 6th Int.Workshop on Mining and Learning with Graphs (MLG) at ICML (2008)
[23] Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2005)
[24] Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
[25] Darwin, I.F.: Java Cookbook. O'Reilly, Sebastopol (2004)

Hierarchical Distance-Based Conceptual Clustering*

A. Funes[1,2], C. Ferri[2], J. Hernández-Orallo[2], and M. J. Ramírez-Quintana[2]

[1] Universidad Nacional de San Luis, Ejército de los Andes 950, 5700 San Luis, Argentina
afunes@unsl.edu.ar
[2] DSIC, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, España
{cferri,jorallo,mramirez}@dsic.upv.es

Abstract. In this work we analyse the relation between hierarchical distance-based clustering and the concepts that can be obtained from the hierarchy by generalisation. Many inconsistencies may arise, because the distance and the conceptual generalisation operator are usually incompatible. To overcome this, we propose an algorithm which integrates distance-based and conceptual clustering. The new dendrograms can show when an element has been integrated to the cluster because it is near in the metric space or because it is covered by the concept. In this way, the new clustering can differ from the original one but the metric traceability is clear. We introduce three different levels of agreement between the clustering hierarchy obtained from the linkage distance and the new hierarchy, and we define properties these generalisation operators should satisfy in order to produce distance-consistent dendrograms.

Keywords: conceptual clustering, hierarchical clustering, generalisation, distances.

1 Introduction

Distances and generalisations are the underlying concepts to two different approaches for machine learning. Similarity, which is a broader concept than distance, is the basis for many inductive inference techniques, since similar elements are expected to behave similarly. Distances do not only formalise the notion of similarity between cases or individuals, but provide the additional properties of metric spaces, which are advantageously exploited by many techniques, known as distance-based.

Generalisation is also another key concept in machine learning. Any inductive learning involves some kind of generalisation. Unlike distance-based methods, some approaches are based on the idea that a generalisation or pattern discovered from old data can be used to describe new data covered by this pattern. These techniques are known as model-based.

Distance-based techniques are quite intuitive and flexible, in the sense that we only need to define a distance function for the data we are working with. However, distance-based methods do not provide a pattern or explanation which justifies the decision made

* This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN under grant TIN2007-68093-C02 and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022). A. Funes was supported by a grant from the Alfa Lernet project and the UNSL.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 349–364, 2008.

for a given individual. In particular, distance-based clustering systems arrange elements into groups based on a numerical measure of similarity between elements. Therefore, the resulting clusters lack conceptual descriptions making them difficult to interpret. For instance, it is helpful to know that a given molecule belongs to a cluster because it is similar to the elements of the cluster according to a certain distance measure, but it would even be more interesting to know what chemical properties are shared by all the molecules in the cluster.

A well-known approach for distance-based clustering is hierarchical clustering [1, 2]. In hierarchical distance-based clustering, data are split into clusters during several partition steps forming a hierarchy of clusters from a single cluster containing all the elements to n clusters containing just one element. Depending on how the hierarchy is built, hierarchical clustering can be classified as agglomerative (bottom-up) or divisive (top-down).

A different approach to clustering is conceptual clustering defined by Michalski [3, 4]. Conceptual clustering overcomes the cluster interpretation problem by forming clusters that can be described by properties involving relations on a selected set of attributes. A conceptual clustering system accepts a set of object descriptions and produces a partition over the observations. These descriptions can be viewed as cluster generalisations, which are expressed as patterns common to all the elements of the cluster.

In this work we present a general approach for clustering in such a way that we use a distance to construct the cluster hierarchy while also producing patterns. The core of the approach is an algorithm for Hierarchical Distance-based Conceptual Clustering (HDCC). The key issue here, which has been neglected by other conceptual clustering methods that use distances, is whether the hierarchy induced by a distance and the discovered patterns are consistent, i.e. are all the elements covered by a pattern close with respect to the underlying distance? To answer the question, first we need to clearly show when this happens. This has led to a new graphical representation of the resulting dendrogram (that we have named conceptual dendrogram). We also need to analyse a priori whether the inconsistencies will appear or not. This has given rise to the development of three levels of consistency between distances and generalisations and the corresponding properties which ensure (in a higher or lower degree) that the conceptual clustering also reflects the distribution of examples in the metric space. This means that if for a given problem we are able to prove these properties, we will know beforehand that the resulting hierarchy of patterns is at the same time consistent with the distance and the concepts expressed by each pattern in the hierarchy.

The main contribution of this work is a practical and general way to integrate hierarchical distance-based and conceptual clustering smoothly. Additionally, the algorithm is also a general way to construct an n-ary generalisation operator from binary generalisation operators in a metric space. Our approach is general in the sense that it can be applied to any distance, pattern language and generalisation operator. Consequently, this idea is directly applicable to structured data. One possible instantiation would provide us with the descriptions or generalisations for clusters of first order atoms obtained by the application of Plotkin´s least general generalisation operator (*lgg*) at the same time that the process of clustering uses a distance for atoms, e.g. the distance defined in [5]. Another direct instantiation would be for example the clustering of lists using regular patterns and the edit distance.

The work is organised as follows. In Section 2 some necessary previous concepts are summarized. Our proposal (HDCC) is presented in Section 3. In Section 4 we show theoretical results about some generalisation operator properties. In section 5 we present some experiments which compare our method to traditional conceptual clustering. Finally, Section 6 closes the paper with the conclusions and future work.

2 Preliminaries

Intuitively, the generalisation of a finite set of elements E in a metric space (X, d) could be extensionally defined as a set that contains E. However, this kind of extensional definition gives no insight on the concept or pattern that the elements in the generalisation share. We say that a pattern $p \in \mathcal{L}$, where $\mathcal{L}$ is the pattern language, is an intensional way of representing a set of elements of X, which are denoted by $Set(p)$.

First we introduce the definition of binary generalisation operators over a metric space and then we extend this concept to patterns in Definition 2.

Definition 1. *Let (X, d) be a metric space and $\mathcal{L}$ a pattern language. A* binary generalisation operator *is a function $\Delta: X \times X \rightarrow \mathcal{L}$ such that given $x_1 \in X$, $x_2 \in X$, $\Delta(x_1, x_2) = p$, where $p \in \mathcal{L}$ and $x_1 \in Set(p)$ and $x_2 \in Set(p)$.*

Fig. 1 (Left) shows five possible generalisations of two points in the metric space $(\Re^2, d)$, where d is the Euclidean distance.

Fig. 1. (Left) Five possible generalisations of two points in $\Re^2$. **(Right)** A generalisation of two patterns p_1 and p_2 in $\Re^2$.

Definition 2. *Let (X, d) be a metric space and $\mathcal{L}$ a pattern language. A* pattern binary generalisation operator *is a function $\Delta^*: \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$ such that given $p_1 \in \mathcal{L}$ and $p_2 \in \mathcal{L}$, $\Delta^*(p_1, p_2) = p$, where $p \in \mathcal{L}$ and $Set(p_i) \subseteq Set(p)$ $(i \in \{1,2\})$.*

Definition 2 establishes that a generalisation of two patterns must describe at least all the elements described by both patterns. In Fig. 1 (Right) we show a possible generalisation for two patterns p_1 and p_2 in $\mathcal{L}$, where $\mathcal{L}$ is the set of all axis-parallel rectangles.

Note that when $\mathcal{L} = X$, as it happens, e.g. with *lgg* for atoms, the operator Δ^* and Δ can be the same.

3 Hierarchical Distance-Based Conceptual Clustering Algorithm

The approach to clustering we propose is based on one of the most known and simple bottom-up distance-based algorithms, the agglomerative hierarchical clustering. It

builds a hierarchy of clusters from individual elements by progressively merging clusters. Clusters are joined based on the distance between them, referred as the linkage distance. Usually, the linkage distance is determined by the maximum distance between elements of each cluster (i.e. complete linkage distance, d^c_L), by the minimum distance between elements of each cluster (i.e. single linkage distance, d^s_L), by the mean distance between elements of each cluster (i.e. average linkage distance, d^a_L), or by the minimum distance between the cluster prototypes (i.e. prototype linkage distance, d^p_L), among others. In the rest of the paper we will only consider these four functions, d^c_L, d^s_L, d^a_L, and d^p_L. We use d_L to refer any of them.

In traditional agglomerative hierarchical clustering, the process of clustering starts at the leaves of the tree where each leaf corresponds to a one-element cluster. Then it joins the two closest clusters into a new cluster that becomes the parent of the formers into the hierarchy. Now the new cluster and the rest minus the two closest ones compose the new set of clusters. This process is repeated until eventually the set of clusters is formed by only one cluster containing all the elements.

A problem appears if we want a pattern or description for each cluster. Since the clustering process is driven by the underlying distance, a discovered pattern obtained by generalisation may describe the elements of a cluster but it might describe other elements of the metric space that are not included into the cluster. This can lead to an inconsistency between the clusters described by the patterns and those resulting from the hierarchical algorithm. To illustrate the problem let us consider the example for lists of symbols given in Fig. 2 (Left). The elements belong to the metric space (X, d) where X is the set of all the finite list of symbols on the alphabet $\Sigma = \{a, b\}$ and d is the edit distance or Levenshtein distance [6] considering the cost of a replacement as the cost of a supression plus an insertion. The figure shows four elements (*aa*, *aab*, *abb*, *aabbbbbb*) and the distances between them.

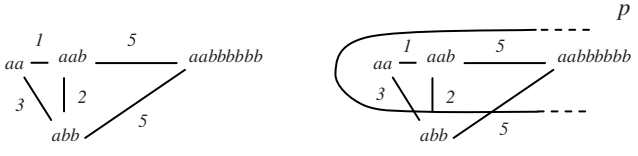


Fig. 2. (Left) Four examples of lists in (Σ^*, d). (Right) The coverage of pattern $p = aa*$.

According to the hierarchical clustering algorithm with single linkage and taking into account the distances between the examples, the resulting clusters are those shown in Fig. 3 (Left). Let us suppose that the chosen generalisation operator produces the pattern $aa*$ for the cluster $\{aa, aab\}$. Clearly, there is a metric inconsistency between the elements described by $aa*$ (*aa*, *aab*, *aaa*, *aaba*, *aabb*,…) and the clusters induced by the distance, since $aa*$ covers *aabbbbbb* but it does not cover *abb*, which is closer (see Fig. 2 (Right)).

With this idea, the proposed approach to hierarchical distance-based conceptual clustering (HDCC) makes a generalisation operator and a distance work together by

achieving a simple adaptation to the hierarchical base algorithm. This adaptation consists in merging to each new cluster all those clusters covered by its generalisation. In this way, the final patterns provide a description common to all the elements that are close according to the underlying distance but also of those that although not close enough to be part of the cluster are covered by the pattern. To represent the resulting clustering we use an extended dendrogram that we have named *conceptual dendrogram*. A conceptual dendrogram provides not only with the traditional information about what elements are in each cluster but it also gives a description of the common properties of their elements in the form of a pattern. A solid line links the clusters merged by the distance, while a dashed line links those merged by a pattern. Fig. 3 (Right) shows the conceptual dendrogram for the current example. The pattern $p = aa*$ covers the cluster {*aa*, *aab*, *aabbbbbb*}, which has been formed considering in first place the distance between the clusters and in second place the coverage of the resulting pattern *aa**.

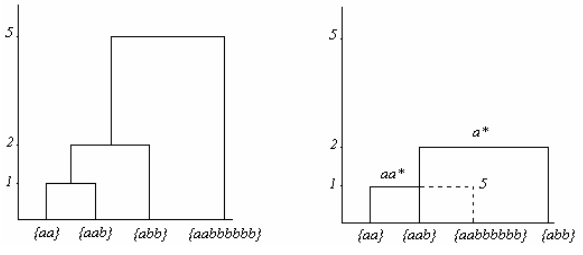


Fig. 3. (Left) Traditional dendrogram. **(Right)** Conceptual dendrogram.

To overcome the inconsistency problem between the distance and the generalisation operator mentioned above, HDCC performs a coverage-reorganisation process that consists in merging to the new cluster C with pattern p all those clusters in the hierarchy that are included in $Set(p)$. Hence these conceptually-added clusters can play a very different role in the construction of the hierarchy. Note that this process is performed during the construction of the hierarchy, and not as a post-process. A post-processing over the original dendrogram would not yield a distance-consistent explanation of the hierarchy and it would imply a much more complex, costly and thorough reorganization of the hierarchy.

Table 1 shows a pseudo code for HDCC. The output is a tree T where each node is a cluster with its corresponding pattern and linkage distance (shown on the Y-axis). The HDCC is in fact a n-ary generalisation operator.

The following simple example illustrates how the HDCC algorithm works under single linkage. Let us suppose the evidence is the set of points in $\Re^2$ shown in Fig. 4 (Left) while the generalisation operator and the pattern language are the same used in the example shown in Fig. 1 (Right).

Fig. 4 (Right) shows the resulting conceptual dendrogram. The clusters {a, b}, {a, b, c}, {a, b, c, d} and {a, b, c, d, e} have been formed driven by the distance. However, as we can see in Fig. 4 (Center) the cluster {i} has been merged to {a, b, c, d, e}

Table 1. Hierarchical distance-based conceptual clustering

Input: $E=\{e_1, e_2,..., e_n\} \subseteq X$ and a distance d, with (X, d) a metric space; $\Delta^*: \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$ a pattern binary generalisation operator; $\Delta: X \times X \rightarrow \mathcal{L}$ a binary generalisation operator; $d_L: 2^X \times 2^X \times (X \times X \rightarrow \Re) \rightarrow \Re$ a linkage distance.

Output: A tree T of clusters and generalisations.

1. $S \leftarrow \{\{e_1\}, \{e_2\}, ..., \{e_n\}\}$.
2. Insert tuple $(\{e_i\}, \Delta(e_i, e_i), 0)$ as a leaf of T, for all $\{e_i\}$ in S.
3. While $S \neq \{E\}$ do
   3.1. Compute $d_L(C_i, C_j, d)$ between each pair of clusters $C_i, C_j \in S$ with $i<j$, using the distance d.
   3.2. Compute the pattern p_{Cxy} of cluster C_{xy} as $\Delta^*(p_{Cx}, p_{Cy})$, where $C_{xy} = C_x \cup C_y$, p_{Cx}, p_{Cy} are the patterns of C_x and C_y, respectively, and C_x and C_y are the closest clusters in S according to d_L.
   3.3. $S \leftarrow S \cup \{C_{xyz}\}$ and $C_{xyz} = C_{xy} \cup C_z$ and $C_z = \{ e \mid e \in C_i \wedge C_i \in S \wedge C_i \subseteq Set(p_{Cxy}) \}$
   3.4. Insert $(C_{xyz}, p_{Cxy}, d_{LCxy})$ in T as the parent node of (C_x, p_{Cx}, d_{LCx}), (C_y, p_{Cy}, d_{LCy}) and of nodes (C_i, p_{Ci}, d_{LCi}) where $C_i \in S$ and $C_i \subseteq Set(p_{Cxy})$.
   3.5. $S \leftarrow S - \{C_i\}$ for all C_i s.t. $C_i \subseteq Set(p_{Cxy})$.
4. Return T.

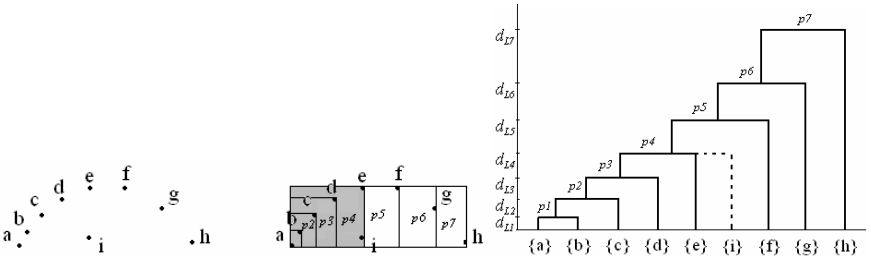


Fig. 4. (Left) A set of points in $\Re^2$. **(Center)** The discovered patterns *p1*,..., *p7*. The shadowed area shows the evidence covered by *p4*. **(Right)** Conceptual dendrogram.

by the pattern p_4 that covers both. Note that $\{i\}$ would have been the last merged cluster by d^s_L in the traditional dendrogram.

4 Consistency Between Distances and Generalisation Operators

The exact shape of the conceptual dendrogram and whether it has dashed links depends not only on the distance d and the generalisation operators used but also on the linkage distance d_L. We can talk of several degrees of consistency between distances and generalisations on the basis of the similarity between a conceptual dendrogram and the traditional one. The more similar the dendrograms are the more consistent the

distance is wrt. the generalisation operator. Next we present three different conditions to ensure that the generalisation operator produces distance-consistent dendrograms.

4.1 Equivalent Dendrograms

In some cases, the conceptual dendrogram is isomorphic to the traditional dendrogram. This happens when the discovered patterns do not cover any other cluster besides those linked by the distance, i.e. each new cluster is formed only by merging the closest clusters or it is composed of only one element (i.e. it is a leaf cluster). Therefore, we say that a conceptual dendrogram is *equivalent to a traditional dendrogram* if for each cluster C which is not a leaf all its children are linked at the same distance l. This is formalised in Definition 3.

Definition 3. *Let T be the tree resulting from* HDCC. *T is* equivalent to a traditional dendrogram *iff* $\forall\ (C, p, l) \in T : (|C| = 1 \vee (\forall\ (C_i, p_i, l_i)$ *child of* $(C, p, l), \exists\ (C_j, p_j, l_j)$ *child of* (C, p, l) *in T such that* $d_L(C_i, C_j, d) = l))$.

If we want equivalent dendrograms, each time HDCC determines the two closest clusters C_1 and C_2 with linkage distance l, the corresponding pattern p should not cover any other cluster C whose distances l_1 and l_2 to C_1 and C_2 respectively are greater than l. Note that l_1 and l_2 can not be lower than l since in this case HDCC would have merged this cluster to C_1 or C_2 before. We say that generalisation operators that generate patterns whose coverage satisfies this condition are *strongly bounded by d_L*. Intuitively, a pattern binary generalisation operator is strongly bounded by d_L when for any pair of patterns p_1, p_2, and any pair of sets C_1 and C_2 covered by each, the linkage distances from the new elements covered by the generalisation of p_1 and p_2 to C_1 and C_2 are equal or lower than the linkage distance between C_1 and C_2, i.e. the new elements covered by the generalisation of p_1 and p_2 fall into the balls of radius $d_L(C_1, C_2, d)$ and centre in the linkage points of C_1 and C_2. The linkage points are, in the case of d^s_L, the two closest elements in C_1 and C_2; the two most distant elements in d^c_L; the prototypes in the case d^p_L and the centroids in the case of d^a_L (assuming the metric space is continuous). This concept is formalised in Definition 4.

Definition 4. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language and d_L a linkage distance. A* pattern binary generalisation operator Δ^* *is strongly bounded by d_L iff* $\forall\ p_1, p_2 \in \mathcal{L}, C_1 \subseteq Set(p_1), C_2 \subseteq Set(p_2), C \subseteq Set(\Delta^*(p_1, p_2)) - (Set(p_1) \cup Set(p_2)) : d_L(C, C_1, d) \leq d_L(C_1, C_2, d) \vee d_L(C, C_2, d) \leq d_L(C_1, C_2, d)$.

Fig. 5 (Left) shows clusters $\{a, b, c\}$ and $\{d, e, f\}$ formed under single linkage. The patterns used are unions of axis-parallel rectangles. A is the generalisation of $\{d, e\}$, $A \cup C$ of $\{d, e, f\}$, B of $\{a, b\}$; $B \cup D$ of $\{a, b, c\}$ and $A \cup C \cup B \cup D \cup E$ of $\{a, b, c, d, e, f\}$. The union of the circles determines the area where the new elements in the generalisation of $A \cup C$ and $B \cup D$ should be if Δ^* is strongly bounded by d^s_L.

Definition 5 gives the same property for a binary generalisation operator. A binary generalisation operator is strongly bounded by d_L when for any pair of elements e_1 and e_2, the linkage distances from $\{e_1\}$ and $\{e_2\}$ to any cluster $\{e\}$ covered by the generalisation of e_1 and e_2 is lower than the linkage distance between $\{e_1\}$ and $\{e_2\}$, i.e. e must fall into the balls of radius $d_L(\{e_1\}, \{e_2\}, d)$ and centre in e_1 and e_2.

Fig. 5. (Left) The union of the circles shows the region where the new elements in the generalisation of $A \cup C$ and $B \cup D$ should be. **(Right)** Maximum coverage of a binary generalisation operator Δ strongly bounded by d_L.

Definition 5. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language and d_L a linkage distance. A* binary generalisation operator *Δ is strongly bounded by d_L iff $\forall\, e, e_1, e_2 \in X$: if $e \in Set(\Delta(e_1, e_2))$ then $d_L(\{e\}, \{e_1\}, d) \leq d_L(\{e_1\}, \{e_2\}, d) \vee d_L(\{e\}, \{e_2\}, d) \leq d_L(\{e_1\}, \{e_2\}, d)$.*

Fig. 5 (Right) shows the area in $\Re^2$ that a pattern p is allowed to cover when Δ is strongly bounded by d_L. Note that when we generalise only two elements the linkage distance d_L is equal to the distance d between the elements for any d_L.

The linkage distance d_L used by HDCC affects the boundedness property of generalisation operators. Given a distance d, a generalisation operator could be strongly bounded under a given d_L but not under a different one. For example, the pattern generalisation operator Δ^* shown in Fig. 4 (Center) is not strongly bounded by d^s_L but it is strongly bounded by d^c_L. We can easily see that it is not strongly bounded by d^s_L because, for instance, the point i covered by the pattern p_4 is outside the balls with centre in d and e and radius $d^s_L(\{a, b, c, d\}, \{e\}, d)$. Note that $d^s_L(\{a, b, c, d\}, \{e\}, d) = d(d, e)$. However, Δ^* is strongly bounded by d^c_L since each pattern covers a rectangle that is determined by the two most distant points e_1 and e_2 in $C1$ and $C2$, and this rectangle is always in the intersection of the two balls $B(e_1, l)$ and $B(e_2, l)$, where $l = d^c_L(C1, C2, d) = d(e_1, e_2)$ and e_1, e_2 are the linkage points in $C1$ and $C2$.

Proposition 1. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language for X, Δ a binary generalisation operator, Δ^* a pattern binary generalisation operator and d_L a linkage distance. For any evidence $E \subseteq X$, the conceptual dendrogram T resulting from HDCC(E, X, d, Δ^*, Δ, d_L) is equivalent to the traditional dendrogram if the generalisation operators Δ and Δ^* are strongly bounded by d_L.*

Proof. There are two different cases to consider in *T:* (*a*) the leaves and (*b*) the internal nodes.

Case (*a*): In the first step HDCC builds n clusters $\{e\}$ and their corresponding generalisations as $\Delta(e, e)$. Since Δ is strongly bounded by d_L we have by Definition 5 $\forall\, e'$, $e \in E$: if $e' \in Set(\Delta(e, e))$ then $d_L(\{e'\}, \{e\}, d) \leq d_L(\{e\}, \{e\}, d)$. Since $d_L(\{e\}, \{e\}, d) = 0$ and d_L is positive, then $d_L(\{e'\}, \{e\}, d) = 0$. The only element e' that satisfies this is $e' = e$. Therefore, after a pattern is computed no other element can be added to the cluster by HDCC. Therefore, in this case, T is equivalent to the traditional dendrogram by Definition 3.

Case (*b*): In the following steps, each new node (C, p, l) in T is formed by merging in first place the two clusters (C_1, p_1, l_1), (C_2, p_2, l_2) whose linkage distance l is the lowest and p is computed as $\Delta^*(p_1, p_2)$. Since Δ^* and Δ are generalisation operators we have that $C \subseteq Set(p)$ and $C_1 \subseteq Set(p_1)$ and $C_2 \subseteq Set(p_2)$.

- If $\Delta^*(p_1, p_2)$ does not cover any other cluster in addition to C_1 and C_2, we have $C = C_1 \cup C_2$ and $d_L(C_1, C_2, d) = l$.
- Let us suppose there is another child (C_3, p_3, l_3) of (C, p, l) such that $C_3 \subseteq Set(\Delta^*(p_1, p_2))-(Set(p_1) \cup Set(p_2))$. Since Δ^* is strongly bounded, by Definition 4 we have $d_L(C_3, C_1, d) \leq d_L(C_1, C_2, d) \vee d_L(C_3, C_2, d) \leq d_L(C_1, C_2, d)$. However, $d_L(C_3, C_1, d)$ must be equal to $d_L(C_1, C_2, d)$ and $d_L(C_3, C_2, d)$ must be equal to $d_L(C_1, C_2, d)$ otherwise HDCC should have merged before C_1 and C_3 or C_2 and C_3 than C_1 and C_2.

Therefore, $d_L(C_i, C_j, d) = l$ for any child (C_i, p_i, l_i), (C_j, p_j, l_j) of (C, p, l) and consequently, in both cases, T is equivalent to the traditional dendrogram by Definition 3. □

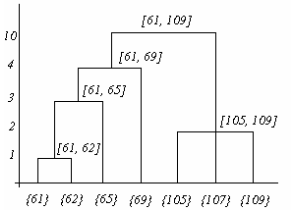


Fig. 6. An equivalent conceptual dendrogram

Fig. 6 shows a simple example of a conceptual dendrogram that is equivalent to the traditional dendrogram under single linkage. $\mathcal{L}$ is the set of the finite closed intervals in $\Re$, and d the absolute difference. $\Delta^*(p_1, p_2)$ is the interval $[min, max]$, where min is the minimum value of the lower bounds of p_1 and p_2, and max is the maximum of the upper bounds. $\Delta(e_1, e_2)$ is $[min(e_1, e_2), max(e_1, e_2)]$. It is easy to see that (a) $\Delta^*$and (b) Δ are strongly bounded by d^s_L:

- (a) Each generalisation of two patterns p_1 and p_2 is a new interval p that only covers the elements covered by p_1 and p_2 and those that are in between of them. If e_1 and e_2 are the linkage points in C_1 and C_2 that have determined the single linkage distance $l = d^s_L(C_1, C_2, d)$, the new elements in the interval p must be included into the two balls $B(e_1, l)$ or $B(e_2, l)$, i.e. the intervals $[e_1 - l, e_1 + l]$ or $[e_2 - l, e_2 + l\,]$. Since e_1 and e_2 are the closest elements in C_1 and C_2, we have that $p_1 = [a, e_1]$ and $p_2 = [e_2, b]$ and $p = [a, b]$. Clearly the new elements in p, i.e. the elements in the interval $]e_1, e_2[$, are included in $[e_1 - l, e_1 + l]$ and also in $[e_2 - l, e_2 + l\,]$ because $l = |e_2 - e_1|$.
- (b) Δ is strongly bounded by d^s_L too because any element in $Set(\Delta(e_1, e_2))$ will be always in between of e_1, e_2. Note that (a) and (b) holds for any d_L here considered.

The condition for having equivalent dendrograms, however, is too strong for many datatypes and generalisation operators given that it forces to minimal generalisations.

In fact a pattern generalisation operator $\Delta^*(p_1, p_2)$ whose coverage $Set(\Delta^*(p_1, p_2))$ is equal to $Set(p_1) \cup Set(p_2)$ is strongly bounded because there is no new elements in $\Delta^*(p_1, p_2)$, so the only set C which must satisfy $(d_L(C, C_1, d) \leq d_L(C_1, C_2, d) \vee d_L(C, C_2, d) \leq d_L(C_1, C_2, d))$ is $C = \emptyset$ and since the d_L from a cluster to $\emptyset$ is zero, the condition holds for any $p_1, p_2 \in \mathcal{L}$, $C_1 \subseteq Set(p_1)$, $C_2 \subseteq Set(p_2)$. The same happens with Δ when $\Delta(e_1, e_2)$ is defined as $\{e_1, e_2\}$.

4.2 Order-Preserving Dendrograms

Sometimes for a given pair of generalisation operators Δ and Δ^*, a distance d and a linkage distance d_L, the conceptual dendrogram –although not equivalent to the traditional one– can just preserve the order in which clusters are merged by d_L, i.e. a discovered pattern will never cover a farther cluster leaving out a closer one. In that case, we say that the conceptual dendrogram is *order-preserving*.

More specifically, an order-preserving conceptual dendrogram is one where for any node (C, p, l) in the hierarchy, its children are linked at the same distance l or they are linked by the pattern at a linkage distance lower than the linkage distance from any other cluster in the hierarchy not covered by the pattern. This concept is formalised by Definition 6.

Definition 6. *Let (X, d) be a metric space and T the tree resulting from HDCC. T is* order-preserving *iff* $\forall (C, p, l), (C_i, p_i, l_i) \in T, \exists (C_j, p_j, l_j) \in T$ *with* (C_i, p_i, l_i) *and* (C_j, p_j, l_j) *children of* (C, p, l) *such that* $d_L(C_i, C_j, d) = l \vee (d_L(C_i, C_j, d) < d_L(C', C_i, d) \wedge d_L(C_i, C_j, d) < d_L(C', C_j, d))$, *for all* $(C', p', l') \in T$, $C' \not\subset Set(p)$.

To obtain an order-preserving conceptual dendrogram, any time HDCC merges two clusters C_1 and C_2 with patterns p_1 and p_2, any other cluster C covered by the generalisation of p_1 and p_2 that has not been linked by the distance d_L must have lower linkage distances to C_1 and C_2 than the linkage distances to C_1 and C_2 from any other cluster C' not covered by the pattern. This is formalized by the property we call *weak boundedness* and that is given by Definition 7. Analogously, Definition 8 establishes the same property for binary generalisation operators.

Definition 7. *Let (X, d) be a metric space,* $\mathcal{L}$ *a pattern language and* d_L *a linkage distance. A pattern binary generalisation operator* Δ^* *is* weakly bounded by d_L *iff* $\forall p_1, p_2 \in \mathcal{L}$, $C_1 \subseteq Set(p_1)$, $C_2 \subseteq Set(p_2)$, $C \subseteq Set(\Delta^*(p_1, p_2)) - (Set(p_1) \cup Set(p_2))$, $C' \not\subset Set(\Delta^*(p_1, p_2))$: $(d_L(C, C_1, d) \leq d_L(C_1, C_2, d) \vee d_L(C, C_2, d) \leq d_L(C_1, C_2, d)) \vee (d_L(C, C_1, d) < d_L(C', C_1, d) \wedge d_L(C, C_2, d) < d_L(C', C_2, d))$.

Definition 8. *Let (X, d) be a metric space,* $\mathcal{L}$ *a pattern language and* d_L *a linkage distance. A binary generalisation operator* Δ *is* weakly bounded by d_L *iff* $\forall e, e', e_1, e_2 \in X$: *if* $e \in Set(\Delta(e_1, e_2))$ *and* $e' \notin Set(\Delta(e_1, e_2))$ *then* $d_L(\{e\}, \{e_1\}, d) \leq d_L(\{e_1\},\{e_2\}, d) \vee d_L(\{e\}, \{e_2\}, d) \leq d_L(\{e_1\},\{e_2\}, d) \vee ((d_L(\{e\}, \{e_1\}, d) < d_L(\{e'\},\{e_1\}, d) \wedge d_L(\{e\}, \{e_2\}, d) < d_L(\{e'\},\{e_2\}, d))$.

Proposition 2. *Let (X, d) be a metric space,* $\mathcal{L}$ *a pattern language,* d_L *a linkage distance,* Δ *a binary generalisation operator, and* Δ^* *a pattern binary generalisation operator.*

(a) If Δ^* is strongly bounded by d_L then Δ^* is weakly bounded by d_L.
(b) If Δ is strongly bounded by d_L then Δ is weakly bounded by d_L.

Proof. Part (a) of Proposition 2 follows immediately from definitions of strongly and weakly bounded operators. Any pattern generalisation operator that is strongly bounded by the linkage distance is also weakly bounded given that Definition 7 relaxes the condition in Definition 4. The same holds for part (b) since Definition 8 relaxes the condition in Definition 5. □

As before, we want to show that the weakly bounded property is a sufficient condition to preserve the order.

Proposition 3. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language, Δ a binary generalisation operator, Δ^* a pattern binary generalisation operator and d_L a linkage distance. For any evidence $E \subseteq X$, the conceptual dendrogram T resulting from HDCC(E, X, d, Δ^*, Δ, d_L) is order-preserving if the generalisation operators Δ and Δ^* are weakly bounded by d_L.*

Proof. There are two different cases to consider in *T:* (*a*) the leaves and (*b*) the internal nodes.

Case (*a*)*:* In the first step HDCC builds n nodes $(\{e\}, \Delta(e, e), l)$ with $l = 0$. If $\Delta(e, e)$ covers any other element this is merged to $\{e\}$.

Since Δ is weakly bounded by d_L we have by Definition 8 $\forall\ e, e', e_1 \in E$*:* if $e \in Set(\Delta(e_1, e_1))$ and $e' \notin Set(\Delta(e_1, e_1))$ then $d_L(\{e\}, \{e_1\}, d) \leq d_L(\{e_1\},\{e_1\}, d) \vee (d_L(\{e\}, \{e_1\}, d) < d_L(\{e'\},\{e_1\}, d)$. Since $d_L(\{e_1\}, \{\ e_1\}, d) = 0$ and d_L is positive we have $\forall\ e, e', e_1 \in E$*:* if $e \in Set(\Delta(e_1, e_1))$ and $e' \notin Set(\Delta(e_1, e_1))$ then $d_L(\{e\},\{e_1\}, d) = 0 \vee d_L(\{e\},\{e_1\}, d) < d_L(\{e'\},\{e_1\}, d)$. Therefore, T is order-preserving by Definition 6.
Case (*b*)*:* In the following steps, each node (C, p, l) in T is formed by merging (in first place) the two clusters (C_1, p_1, l_1), (C_2, p_2, l_2) whose linkage distance l is the lowest and p is computed as $\Delta^*(p_1, p_2)$. Since Δ^* and Δ are generalisation operators we have that $C \subseteq Set(p)$ and $C_1 \subseteq Set(p_1)$ and $C_2 \subseteq Set(p_2)$.

- If $\Delta^*(p_1, p_2)$ does not cover any other cluster different to C_1 and C_2, we have $C = C_1 \cup C_2$ and $d_L(C_1, C_2, d) = l$.
- Let us suppose there is another child (C_3, p_3, l_3) of (C, p, l) such that $C_3 \subseteq Set(\Delta^*(p_1, p_2)) - (Set(p_1) \cup Set(p_2))$. Since Δ^* is weakly bounded, by Definition 7 we have $d_L(C_3, C_1, d) \leq d_L(C_1, C_2, d) \vee d_L(C_3, C_2, d) \leq d_L(C_1, C_2, d) \vee (d_L(C_3, C_1, d) < d_L(C', C_1, d) \wedge d_L(C_3, C_2, d) < d_L(C', C_2, d))$ for all $C' \not\subset Set(\Delta^*(p_1, p_2))$. By reasoning as in Proposition 1 we have, $d_L(C_3, C_1, d) = d_L(C_3, C_2, d) = d_L(C_1, C_2, d) = l \vee (d_L(C_3, C_1, d) < d_L(C', C_1, d) \wedge d_L(C_3, C_2, d) < d_L(C', C_2, d))$.

Therefore, in both cases, T is order-preserving by Definition 6. □

The conceptual dendrogram of Fig. 3 (Right) is not order-preserving under the single linkage. Δ^* is not weakly bounded by d^s_L since the pattern *aa** has linked first the cluster {*aabbbbbb*}, which is farther from {*aa*} and {*aab*} than {*abb*}.

Fig. 7 shows an example of an order-preserving dendrogram for nominal data using d^s_L. We have used a distance similar to the one defined in [14], a distance induced by a relationship R, where R is a partial order. R is defined as xRy if x is a y. Fig. 7 (Left) shows part of a relationship R as a tree hierarchy. The distance between two elements is the sum of costs associated to each edge of the shortest path connecting them. The cost of an edge of level i is $w_i = 1/2^i$. $\Delta(e_1,e_2)$ is defined as the minimun ancestor of e_1 and e_2 if $e_1 \neq e_2$ otherwise is equal to e_1, and $\Delta^*(p_1, p_2)$ is defined anlogously. In Fig. 7 (Top right) we can see the traditional dendrogram, and the corresponding conceptual dendrogram in Fig. 7 (Bottom right). The evidence is formed only by elements in the leaves of R. The internal nodes are generalisations.

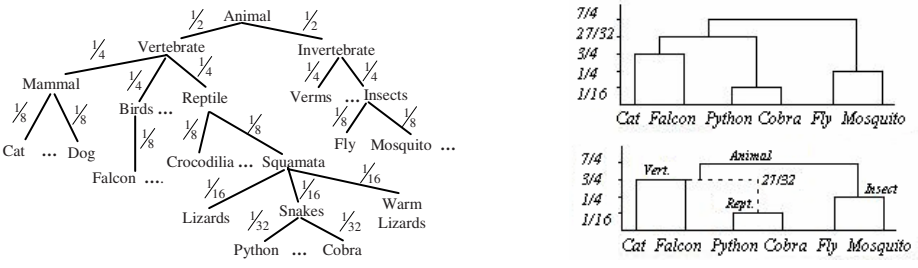


Fig. 7. (Left) Relationship R as a tree**. (Top right)** Traditional dendrogram. **(Bottom right)** Not equivalent but order-preserving conceptual dendrogram.

Note that a pattern generalisation operator that covers all the space (the maximal operator $\Delta^*(p_1, p_2) = p$ where $\text{Set}(p) = X$) is trivially weakly bounded because we cannot find a cluster C such that $C \not\subset Set(\Delta^*(p_1, p_2))$.

4.3 Acceptable Generalisation Operators

There are some generalisation operators that although not (weakly) bounded lead to dendrograms which are consistent with the distance in a broader sense. The idea is that a pattern should not cover new elements whose distance to the old elements is greater than the greatest distance between the old elements. We refer to the operators that produce this kind of patterns as *acceptable*. In this case, the dendrograms can differ significantly.

Definition 9. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language and $d^c_L(\, . \, , \, . \, , \, . \,)$ the complete linkage distance. A pattern binary generalisation operator Δ^* is* acceptable *iff* $\forall \; p_1, \; p_2 \in \mathcal{L}, \; e \in Set(\Delta^*(p_1, \; p_2)), \; \exists \; e' \in Set(p_1) \cup Set(p_2) : d(e, \; e') \leq d^c_L(Set(p_1),Set(p_2), \; d)$.

In Fig. 8 (Left) the union of the circles whose centres are in $Set(p_1) \cup Set(p_2)$ and radius equal to $d^c_L(Set(p_1), \; Set(p_2), \; d)$ determines the maximum coverage for a pattern produced by an acceptable generalisation operator for the evidence $\{a, \; b, \; c, \; d\}$ in $\Re^2$.

Note that a pattern generalisation operator is acceptable independently of the linkage distance used. It only depends on the distance d between the two most distant

elements in the clusters. We use d^c_L in the definition to simplify the notation. Definition 10 gives the same concept applied to binary generalisation operators.

Definition 10. *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language. A binary generalisation operator Δ is* acceptable *iff* $\forall e, e_1, e_2 \in X$ *: if* $e \in Set(\Delta(e_1, e_2))$ *then* $d(e, e_1) \leq d(e_1, e_2)$ *or* $d(e, e_2) \leq d(e_1, e_2)$.

We can see from Definition 10 and Definition 5 that any binary generalisation operator Δ is acceptable if and only if it is strongly bounded by the linkage distance since the linkage distance d_L is equal to d when applied to single sets for any of the linkage distances here considered.

The pattern binary generalisation operator Δ^* used in Fig. 6 is acceptable, since all the elements in p will always fall between the bounds of p_1 and p_2 and consequently at a distance lower than the two most distant elements in p_1 and p_2. Δ is also acceptable because, as we showed for the example of Fig. 6, it is strongly bounded by the linkage distance.

The good thing is that $\Delta(e, e) = \{e\}$ is strongly bounded, and hence acceptable. This operator can usually be expressed in most $\mathcal{L}$. So, only Δ^* must be analysed in most cases and, additionally, it is independent from d_L. Results obtained for acceptability will be then extensible to whatever linkage function.

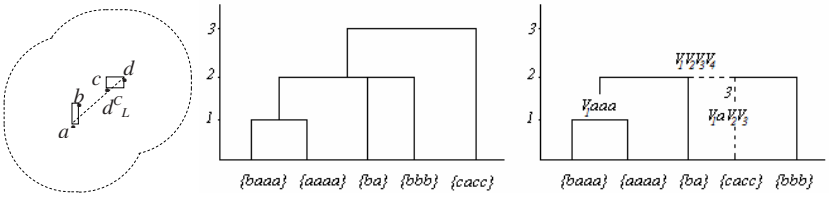


Fig. 8. (Left) The coverage of an acceptable Δ^* for the evidence $\{a, b, c, d\}$ in $\Re^2$**. (Center)** Traditional dendrogram. **(Right)** Conceptual dendrogram.

Fig. 8 (Center) shows the traditional dendrogram for the evidence {*baaa*, *aaaa*, *ba*, *bbb*, *cacc*} and Fig. 8 (Right) the corresponding conceptual dendrogram, both using d^s_L. The metric space is (Σ^*, d), where Σ^* is the space of states of lists formed from Σ included the empty list λ, $\Sigma = \{a, b, c\}$ and d is the edit distance. The pattern language $\mathcal{L}$ is given by all the finite lists from the alphabet $\Sigma' = \Sigma \cup V \cup \{\lambda\}$ where $V = \{V_1, V_2, \ldots, V_n\}$ is a set of variables. The variables are used to generalise symbols in $\Sigma \cup \{\lambda\}$. The generalisation of two lists is given by $\Delta(l_1, l_2) = p$ where p is formed by the patterns given by the optimal alignments of l_1 and l_2 and whose length is given by the length of the longest pattern l_1 or l_2. Variables represent the symbols that do not match. For instance, $\Delta(aabaaa, ababaa) = aV_1V_2V_3aa$. $\Delta^*(p_1, p_2)$ is computed analogously, e.g. $\Delta^*(aV_1V_2V_3aa, baa) = V_1V_2V_3V_4aa$. Although Δ and Δ^* are not bounded under the single linkage distance, they are acceptable given that each pattern covers elements whose distances are at most the number of variables in the pattern and this is precisely the maximum distance possible between two elements in $Set(p)$, in particular to the elements in $Set(p_1) \cup Set(p_2)$.

5 Experimental Results

One question that arises from the previous proposal is whether the new conceptual clustering, coming from the on-line re-arrangement of the dendrogram might undermine cluster quality (in the cases where the dendrograms are not equivalent, naturally). In order to bring some light on this, the experiments below compare HDCC against the traditional version of the hierarchical clustering algorithm. We constructed 100 artificial datasets by drawing points from a finite mixture of k Gaussian distributions in $\Re^2$ whose means are randomly located in $[0, 100]^2$ with a standard deviation of 1. Although k represents the actual number of *gaussians* in a dataset, note that there might be overlapping between *gaussians*, so having fewer clusters. We set $k = 3$, and each dataset was formed by 600 points (200 points were drawn from each of the 3 Gaussian distributions). The experiments were conducted under single and complete linkage and using two different language patterns $\mathcal{L}_1$ and $\mathcal{L}_2$. $\mathcal{L}_1$ is the language of axis-parallel rectangles and $\mathcal{L}_2$ is the language of circles.

Fig. 9 shows the discovered patterns in $\mathcal{L}_1$ and $\mathcal{L}_2$ for one dataset with 600 points drawn from three Gaussian distributions, one using d^s_L (Left) and the other using d^c_L (Right). Note that the rectangles obtained incrementally by HDCC fit the points as well as an n-ary operator. This is not the case in $\mathcal{L}_2$ where the discovered patterns are more general than using an n-ary operator. However, as we can see in Table 2, it does not affect the clustering quality because they are built incrementally and HDCC in each step only merges those clusters that are completely covered by the pattern.

To assess the quality of the clustering we employed a measure, S, that reflects the mean scattering over the k clusters (see eq. (1)). The lower S is the better the clustering is. Table 2 shows S averaged over n different experiments. Note that n can take values less than 100 in HDCC since the resulting hierarchies do not always have a clustering of k clusters (several clusters may be joined by a discovered cluster pattern in one step).

$$S = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\sum_{j=1 l=j+1}^{m}\sum^{m} d(\vec{x_j}, \vec{x_l})^2}\ . \tag{1}$$

The experiments show that not only quality is not degraded, but for $\mathcal{L}_2$ HDCC sometimes outperforms the traditional algorithm under single linkage. Similar results were obtained with points in $[0, 10]^2$. Logically, different results might be obtained using non-convex or complex-shaped patterns.

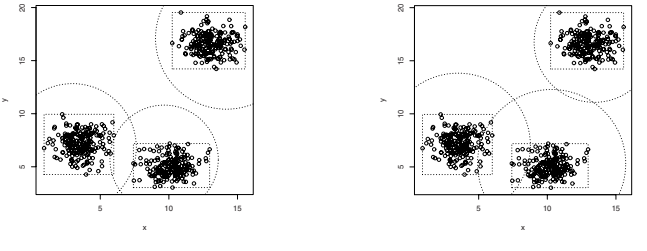


Fig. 9. Discovered patterns in L_1 and L_2 using d^s_L **(Left)** and using d^c_L **(Right)**

Table 2. Values of S for traditional and conceptual dendrograms for k =3

		Traditional S n	Conceptual S n	Dendrogram Relation
$\mathcal{L}_1$	Single	292,293 (100)	292,295 (100)	Not equivalent
	Complete	281,393 (100)	281,393 (100)	Equivalent
$\mathcal{L}_2$	Single	292,293 (100)	281,549 (98)	Not equivalent
	Complete	281,393 (100)	281,727 (100)	Not equivalent

6 Conclusions

We have presented a general approach for hierarchical conceptual clustering based on distances and generalisation operators. It puts together the flexibility of hierarchical distance-based clustering and the interpretability of conceptual clustering. For instance, a user can choose any part of a dendrogram, get a description also learning whether all the covered elements are close wrt. the underlying metric.

Several clustering algorithms that generate concept descriptions can be found in the literature. On the one hand we have those coming from traditional conceptual clustering such as CLUSTER/2 [4], COBWEB [7] and GCF [8]. On the other hand we have those that, using a subset of first-order logic as representation language, apply traditional distance-based clustering algorithms. In this second group we can find KBG [9], C 0.5 [10], COLA-2 [11], and TIC [12, 13] among others. Our proposal is different to all the conceptual clustering methods which also use a distance in the way that it is general to any datatype (any generalisation operator and distance can be used). Moreover, we present graphical extensions to see the divergence between the distance and the generalisation operator a posteriori, but also conditions that can be checked a priori to ensure that the resulting conceptual dendrograms are consistent with the underlying distance. Our work is related to [14] where the author analyses the relationship between distances and generalisations and proposes a framework where these two paradigms can be integrated in a consistent way. In [14] the analysis is achieved on generalisation operators defined on a metric space and not over a language of patterns as it is done here.

Additionally, as we have said, HDCC can be seen as an n-ary operator constructed over binary operators by only applying the binary operators at most n times, where n is the number of examples. This is an interesting property for machine learning areas which have well-established binary generalisations operators, such as ILP.

The instantiation of HDCC to propositional clustering is direct, when datatypes are nominal or numerical. We have shown in [15] that the common generalisation operators for nominal data (extensional set) and numerical data (intervals) are strongly bounded in the metric spaces defined by the distance functions commonly used for these datatypes (discrete distance and difference distance). Hence in this case the distance-based conceptual dendrograms are equivalent to classical distance-based dendrograms, independently of the linkage distance. The problem is also analysed when the tuple is composed of both nominal and numerical data, and the generalisation operators are extended accordingly. Examples of this have been shown in the experiments section in this paper.

Things are more diverse (and interesting) when applying the proposal to structured datatypes. We have seen several examples in this paper when the conditions hold for the complete linkage but not for the single linkage, or only one of the degrees (the weakest one, acceptability) is met. We are currently working on the establishment of operative combinations of distances and generalisation operators for lists and sets.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Survey 31(3), 264–323 (1999)
2. Berkhin, P.: A Survey of Clustering Data Mining Techniques, Grouping Multidimensional Data, pp. 25–71. Springer, Heidelberg (2006)
3. Michalski, R.S.: Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. Policy Analysis and Information Systems 4(3), 219–244 (1980)
4. Michalski, R.S., Stepp, R.E.: Learning from Observation: Conceptual Clustering. In: Michalski, et al. (eds.) Machine Learning: An Artificial Intelligence Approach, pp. 331–363. TIOGA Publishing Co. (1983)
5. Ramon, J., Bruynooghe, M., Van Laer, W.: Distance measures between atoms. CompulogNet Meeting ComputingLogic & Machine Learning, 35–41 (1998)
6. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10, 707–710 (1966)
7. Fisher, D.: Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning 2, 139–172 (1987)
8. Talavera, L., Béjar, J.: Generality-Based Conceptual Clustering with Probabilistic Concepts. IEEE Transactions on Pattern Analysis & Machine Inteligence 23(2) (2001)
9. Bisson, G.: Conceptual Clustering in a First Order Logic Representation. In: European Conference on Artificial Intelligence, pp. 458–462 (1992)
10. De Raedt, L., Blockeel, H.: Using Logical Decision Trees for Clustering. In: Džeroski, S., Lavrač, N. (eds.) ILP 1997. LNCS, vol. 1297, pp. 133–140. Springer, Heidelberg (1997)
11. Emde, W.: Inductive learning of characteristic concept descriptions. In: Proc. 4th Intl Workshop on Inductive Logic Programming (ILP 1994) (1994)
12. Blockeel, H., De Raedt, L.: Top-down induction of first order logical decision trees. Artificial Intelligence 101, 285–297 (1998)
13. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the 15th Intl. Conference on Machine Learning, pp. 55–63 (1998)
14. Estruch, V.: Bridging the gap between distance and generalisation:Symbolic learning in metric spaces. PhD thesis, DSIC-UPV (2008), `http://www.dsic.upv.es/~vestruch/thesis.pdf`
15. Funes, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Technical Report, DSIC (2008), `http://www.dsic.upv.es/~flip/#Papers`

Mining Frequent Connected Subgraphs Reducing the Number of Candidates

Andrés Gago Alonso[1,2], José Eladio Medina Pagola[1],
Jesús Ariel Carrasco-Ochoa[2], and José Fco. Martínez-Trinidad[2]

[1] Data Mining Departament,
Advanced Technologies Application Center (CENATAV),
7a ♯ 21812 e/ 218 y 222, Rpto. Siboney, Playa, CP: 12200, La Habana, Cuba
{agago,jmedina}@cenatav.co.cu
[2] National Institute of Astrophysics, Optics and Electronics (INAOE),
Luis Enrique Erro No. 1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico
{ariel,fmartine}@inaoep.mx

Abstract. In this paper, a new algorithm for mining frequent connected subgraphs called gRed (graph Candidate Reduction Miner) is presented. This algorithm is based on the gSpan algorithm proposed by Yan and Jan. In this method, the mining process is optimized introducing new heuristics to reduce the number of candidates. The performance of gRed is compared against two of the most popular and efficient algorithms available in the literature (gSpan and Gaston). The experimentation on real world databases shows the performance of our proposal overcoming gSpan, and achieving better performance than Gaston for low minimal support when databases are large.

1 Introduction

Nowadays, due to the rapid scientific and technological advances, there are great creation, storage and data distribution capacities. This situation has boosted the necessity of new tools to transform this big amount of data into useful information or knowledge for decision makers. When these data are complex and structured, this transformation requires techniques that usually have high time and memory requirements. Examples of these techniques are those related to frequent subgraph mining; i.e., the process of finding subgraphs that occur frequently in a collection of graphs.

Frequent subgraph mining has become an important topic in data mining researches with wide applications [3], including mining substructures from chemical compound databases, XML documents, citation networks, biological networks, etc. As consequence several algorithms have been proposed to find all frequent connected subgraphs in collections of labeled graphs [6,7,11,2,4,8].

Labeled graphs can be used to model relations among data in the aforementioned applications because labels can represent attributes of entities and relations among themselves. For example in chemistry, the different kinds of atoms

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 365–376, 2008.

and bonds in a chemical compound can be modeled by vertex and edge labels respectively.

The first frequent subgraph miner called AGM was introduced by Inokuchi *et al.* for unconnected graphs [5]. This algorithm was followed by the FSG algorithm [7] and AcGM [6] (an adaptation of AGM), for mining frequent connected subgraphs. These algorithms have the same setup as the original Apriori algorithm for mining frequent itemsets [1].

To avoid supposed overheads incurred in the earlier algorithms, new pattern growth based algorithms such as gSpan [11], MoFa [2], FFSM [4] and Gaston [8] were developed. In [10] these algorithms were compared in a common framework. In this experimentation, the four algorithms were competitive among themselves, although Gaston and MoFa were the fastest and slowest algorithms respectively, in almost all tests. On the other hand, gSpan was the best algorithm regarding its memory requirements. The embedding structures used by MoFa, FFSM and Gaston could cause problems if not enough memory is available or if the memory throughput is not high enough.

In this paper, a new pattern growth algorithm called gRed (graph Candidate Reduction Miner) is introduced. This algorithm is based on the gSpan scheme; but using novel properties of the DFS code that allows to reduce the number of candidates for optimizing the mining process.

The basic outline of this paper is as follows. Section 2 is dedicated to the related work, it includes the basic concepts introduced by gSpan algorithm. The details of gRed algorithm are discussed in the section 3. The experimental results are presented in section 4. Conclusions of the research and some ideas about future directions are exposed in section 5.

2 Preliminaries

In gRed, each candidate graph is represented by its minimum DFS (Depth First Search) code. This kind of canonical representation, based on DFS graph traversal, was introduced in gSpan [11]. Some concepts introduced in gSpan are required for understanding our algorithm; therefore, we include them in section 2.1.

2.1 Basic gSpan Concepts

The basic concepts we will use in this paper are the following.

DFS Tree. A DFS tree T is constructed when a DFS traversal in a graph $G = \langle V, E \rangle$ is performed. G can have different DFS trees because there are more than one DFS traversal. Each DFS traversal (DFS tree) defines a unique order among all the vertices; therefore, we can number each vertex according to this DFS order. The root and the right most vertex in T are v_0 and v_n respectively. The right most path is the straight path from v_0 to v_n in T. The forward edge set $F(T)$ contains all the edges in T, and the backward edge set $B(T)$ contains the edges which are not in T.

Procedure `MainLoop`(D, δ, S)

Input: D - database, δ - support threshold
Output: S - mining results

```
1 Remove infrequent vertices and edges;
2 S ← S^1 ← all frequent 1-edge codes
3 forall code s ∈ S^1 do
4 |   Initialize the TID list s.L by the graphs which contains the edge of s;
5 |   SubgraphMining(D,s,δ,S);
6 |   D ← D \ e;
7 |   if |D| < δ then break;
8 end
```

Procedure `SubgraphMining`(D, s, δ, S)

```
1 if not isMin(s) then return;
2 S ← S ∪ {s};
3 Enumerate(D, s, RE);
4 forall edge extension e ∈ RE do
5 |   if s ◇ e.support ≥ δ then SubgraphMining(D, s ◇ e, δ, S);
6 end
```

Procedure `Enumerate`(D, s, RE)

```
1 forall graph g ∈ s.L do
2 |   Enumerate the next occurrence of s in g;
3 |   forall right most extension e of s do
4 |   |   s ◇ e.L ← s ◇ e.L ∪ {g};
5 |   |   RE ← RE ∪ {e};
6 |   end
7 |   if any occurrence of s in g are not covered then goto line 2;
8 end
```

Fig. 1. General description of gSpan algorithm

DFS Code. The DFS code is a sequence of edges built from the DFS Tree. This sequence is obtained considering the destination vertices in $F(T)$ according to the DFS order. The backward edges from a vertex are inserted just before its forward edges; if the vertex has several backward edges, these are included in the DFS order of their destination vertices. Multiple edges between same vertices are ordered according to the lexicographic order ($\prec_l$) of its labels. These considerations for building the sequence define the following linear order ($\prec_e$) between two edges.

DFS lexicographic order. For simplicity, each edge can be represented by a 5-tuple, $(i, j, l_i, l_{(i,j)}, l_j)$ where i and j are the subindices of the vertices (v_i and v_j), l_i and l_j are the labels of these vertices respectively, and $l_{(i,j)}$ is the label of the edge. If $i < j$ it is a forward edge; otherwise it is a backward edge.

In summary, the inequality $e_1 \prec_e e_2$ holds (assume that $e_1 = (i_1, j_1, \ldots)$ and $e_2 = (i_2, j_2, \ldots)$) if and only if one of the following statements is true:

- $e_1, e_2 \in F(T)$ and $j_1 < j_2$ or $i_1 > i_2 \wedge j_1 = j_2$;
- $e_1, e_2 \in B(T)$ and $i_1 < i_2$ or $i_1 = i_2 \wedge j_1 < j_2$;
- $e_1 \in B(T)$, $e_2 \in F(T)$ and $i_1 < j_2$;
- $e_1 \in F(T)$, $e_2 \in B(T)$ and $j_1 \leq i_2$;
- $i_1 = i_2$, $j_1 = j_2$ and $e_1 \prec_l e_2$.

The lexicographic order $\prec_l$ compares the edges e_1 and e_2 regarding the last three components in each 5-tuple. The vertex label l_i gets first priority, the edge label $l_{(i,j)}$ gets the second, and the vertex label l_j gets the third to determine the order between two edges.

The order $\prec_e$ can be also extended to a lexicographic order ($\prec_s$) to compare two edge sequences (two DFS codes). Let $s_1 = (a_1, a_2, \ldots, a_m)$ and $s_2 = (b_1, b_2, \ldots, b_n)$ be two DFS codes. We say that $s_1 \prec_s s_2$ if one of the following conditions is true:

$$\exists t, \forall k < t, a_k = b_k, \text{ and } a_t \prec_e b_t \; ; \tag{1}$$

$$m < n \text{ and } \forall k \leq m, a_k = b_k \; . \tag{2}$$

Minimum DFS Code. It is defined as the minimum sequence according to the order $\prec_s$ among all DFS codes of the same graph.

Rightmost path extension. Given a DFS code s and an edge e, e is a rightmost path extension of s if e connects the rightmost vertex with another vertex in the rightmost path (backward extension); or it introduces a new vertex connected from a vertex of the rightmost path (forward extension). In such cases, the DFS code $s' = s \diamond e$ is the code obtained extending s by e; s' is called a child of s or s is called a parent of s'.

gSpan guarantees the completeness of mining results only working with the minimum DFS codes, pruning non minimal children in the solution space. Fig. 1 describes the pseudo-code of gSpan. This pseudo-code is an integration of the algorithm descriptions presented in [11,12].

All pattern growth algorithms generate duplicated candidates during the enumeration process. In gSpan, the duplicated candidates are non-minimal codes. Instead of calculating the minimum DFS code of s from all possible DFS codes, picking up the smallest one and comparing it against s, gSpan defines a more efficient function `isMin`(s) in line 1 of `SubgraphMining`. A heuristic search was designed using the DFS lexicographic order. Whenever some prefix of a DFS is generated and it is less than s, then s is not minimal and the search concludes.

For support calculation and candidate enumeration, gSpan uses a TID list. The TID list (Transaction ID list) contains the ID of each graph in the database that holds the corresponding subgraph. In the procedure `Enumerate`, $s.L$ is used to determine the possible extension set for s, performing subgraph isomorphism tests to find all the embeddings of s in each graph in $s.L$. In line 5 of `SubgraphMining`, the support of $s \diamond e$ is the length of $s \diamond e.L$.

3 The gRed Algorithm

The gRed algorithm can also be described using the pseudo-code of Fig. 1. Only the procedures `SubgraphMining` and `Enumerate` are changed by new procedures `gRed-SubgraphMining` and `gRed-Enumerate` respectively. Novel properties of the DFS codes introduced in the following section are used by the new procedures for optimizing the mining process.

3.1 DFS Codes

Suppose that $s = e_0, e_1, \ldots, e_m$ is a minimum DFS code. The set $RE(s)$ of all rightmost path extensions (see section 2.1) of s can be partitioned into three sets $B(s)$ (backward extensions), $FV(s)$ (forward extensions from the rightmost vertex) and $FN(s)$ (forward extensions from a non rightmost vertex in the right most path), $RE(s) = B(s) \cup FV(s) \cup FN(s)$ (see Fig. 2). Thus, the forward extensions can be represented by $F(s) = FV(s) \cup FN(s)$.

Let v_i be a vertex in the right most path, let v_n be the right most vertex in s and let $F_i(s)$ be the forward extension set from vertex v_i. If $v_i \neq v_n$ we can use $FN_i(s)$ to refer to $F_i(s)$. Similarly, we use $B_i(s)$ to denote the set that contains the backward extensions to the destination vertex v_i. For example, in Fig. 2 the right most path is (v_0, v_4, v_6); therefore, we have $FV(s) = F_6(s)$, $FN(s) = F_0(s) \cup F_4(s)$ and $B(s) = B_0(s) \cup B_4(s)$.

If $v_i \neq v_n$, we denote f_i as the forward edge from v_i lying in the right most path. For each edge e, we use e^{-1} to refer to the reverse edge of e. For example in Fig. 2, we have $f_4 = (4, 6, B, \_, C)$ and $f_4^{-1} = (6, 4, C, \_, B)$.

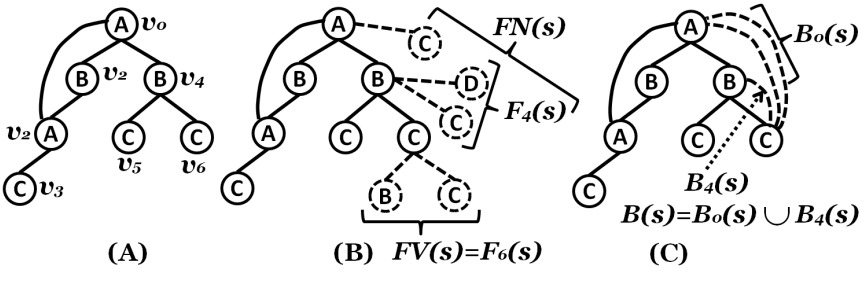


Fig. 2. Partitions in the rightmost path extensions, (A) example of a DFS tree, (B) the forward extensions $FV(s) \cup FN(s)$ and (C) the backward extensions $B(s)$

The following two results are sufficient conditions for a child to be non minimal. Both statements can be used by the procedure `gRed-Enumerate` to filter the possible extensions from a DFS code s.

Proposition 1. *Let s be a minimal DFS code. If $e \in FN_i(s)$ and $e \prec_l f_i$, then $s' = s \diamond e$ is a non-minimal child of s.*

Proof. Let h be an integer number such that $e_h = f_i$. The edges e_h and e start from v_i and they are forward edges in s'. Therefore, we can perform a DFS traversal visiting first e and then e_h or vice versa. If e is visited immediately before than e_h, the resulting DFS code has the following format $s'_1 = e_0, \ldots, e_{h-1}, e, e'_h, \ldots, e'_m$, where e'_j is e_j with another subindices for each $j \geq h$. The codes s' and s'_1 have the same prefix $e_0, \ldots, e_{h-1}$ and we are considering that $e \prec_l e_h$; therefore, by the condition (1), $s'_1 \prec_s s'$. Thus, we conclude that s' is a non-minimal child of s. □

The proposition 1 allows to characterize the non-minimality of certain forward extensions. Similar results for backward extensions are showed in proposition 2.

Proposition 2. *Let s be a minimal DFS code. If $e \in B_i(s)$ and $e^{-1} \prec_l f_i$, then $s' = s \diamond e$ is a non-minimal child of s.*

Proof. Similarly to the proof of the proposition 1, h is the integer number such that $e_h = f_i$. We can perform a DFS traversal visiting first e^{-1} and then e_h or vice versa. If e^{-1} is visited immediately before than e_h, the resulting DFS code have the following format $s'_1 = e_0, \ldots, e_{h-1}, e^{-1}, e'_h, \ldots, e'_m$. The codes s' and s'_1 have the same prefix $e_0, \ldots, e_{h-1}$ and we assume that $e^{-1} \prec_l e_h$; therefore, $s'_1 \prec_s s'$. Thus, we conclude that s' is a non-minimal child of s. □

In gSpan, a duplicate test (minimality test) is performed for each frequent child of s, i.e. the number of such tests is $|RE(s)|$. Let $RE_0(s)$ be the extension set obtained from $RE(s)$ by removing the extensions whose non-minimality is guaranteed according to the propositions 1 and 2. The algorithm gRed only considers the extensions in $RE_0(s)$.

It is not always necessary to perform the duplicate test for each child of s in $RE_0(s)$. The following propositions allow to avoid some minimality tests in the context of the DFS codes. The procedure `gRed-SubgraphMining` uses this properties to speedup the mining process.

Proposition 3. *Let s be a minimal DFS code and let $e, e' \in F_i(s)$ be two forward extensions of s by the same vertex v_i. Then, the following statements are true:*

1. *if $s \diamond e$ is a minimal child and $e \preceq_l e'$, then $s \diamond e'$ is a minimal child;*
2. *if $s \diamond e$ is a non-minimal child and $e' \preceq_l e$, then $s \diamond e'$ is a non-minimal child.*

Proof. Let us proof separately each case. We assume that s is the edge sequence $e_0, e_1, \ldots, e_m$.

In the first case, we have that $s \diamond e$ is a minimal child and $e \preceq_s e'$. Suppose that $s \diamond e'$ is a non-minimal child, then there is at least one code $s_1 = a_0, a_1, \ldots, a_{m+1}$ such that $s_1 \prec_s s \diamond e'$. Using the definition of $\prec_s$ (see section 2.1), there is an integer t, $0 \leq t \leq m$ such that $a_k = e_k$ for all $k < t$, and $a_t \prec_e e_t$. As it can be noticed, $t < m+1$ because s is a minimal DFS code. Thus, by the condition (1), $s_1 \prec_s s$.

Since e and e' start from the same vertex, we can replace the edge representing e' in s_1 by the edge e. Assume s_1 as the code obtained when replacing of e' by e in

s_1; this code is a valid one for the graph coded by $s \diamond e$ and we have $s'_1 \prec_s s_1 \prec_s s$. Using the condition (2), $s'_1 \prec_s s \prec_s s \diamond e$. Then, $s \diamond e$ is a non-minimal child of s, representing a contradiction. Therefore, the initial assumption ($s \diamond e'$ is a non-minimal child) must be false. Thus, we conclude the proof for the first case.

In the second case, we have that $s \diamond e$ is a non-minimal child and $e' \preceq_s e$. Then, there is at least one code $s_1 = a_0, a_1, \ldots, a_{m+1}$ such that $s_1 \prec_s s \diamond e'$. Let t be the integer such that $0 \leq t \leq m$, $a_k = e_k$ for all $k < t$, and $a_t \prec_e e_t$. Thus, by the condition (1), we have $s_1 \prec_s s$. Since e and e' start from the same vertex, we can replace the edge representing e in s_1 by the edge e'. The resulting code (assume it is s'_1) is a valid DFS code for the graph coded by $s \diamond e'$ and we have $s'_1 \prec_s s_1 \prec_s s \prec_s s \diamond e'$. Therefore, $s \diamond e'$ is a non-minimal child. □

Proposition 4. *Let s be a minimal DFS code and let $e, e' \in B_i(s)$ be two backward extensions of s with destination vertex v_i. Then, the following statements are true:*

1. *if $s \diamond e$ is a minimal child and $e \preceq_l e'$, then $s \diamond e'$ is a minimal child;*
2. *if $s \diamond e$ is a non-minimal child and $e' \preceq_l e$, then $s \diamond e'$ is a non-minimal child.*

Proof. The proof is similar to that given for proposition 3. □

The last two propositions state that the minimality tests are not required in some elements of $B_i(s)$ or $F_i(s)$. The extensions in $RE_0(s)$ are sorted according to the aforementioned order $\prec_e$, firstly the elements in $B_0(s), \ldots, B_n(s)$, and finally the elements in $F_n(s), \ldots, F_0(s)$. Each set $B_i(s)$ or $F_i(s)$ is ordered internally by the lexicographic order $\prec_l$. Under the proposition 3, we only need to find the minimal extension $e \in F_i(s)$ such that its corresponding predecessor extensions in $F_i(s)$ are non-minimal. The successors of e in $F_i(s)$ are also minimal extensions; therefore, the minimality tests are not required. Similar observations might be indicated for each $B_i(s)$ using the proposition 4.

These four propositions were not used in the original gSpan algorithm described in [11,12]. In the following section we illustrate how the aforementioned properties are used to design the gRed algorithm.

3.2 The Algorithm

The Fig. 3 outlines the pseudo-code of the gRed algorithm. Note that D represents the graph database, δ is the minimum support threshold and S contains the mining result.

`gRed-MainLoop` is quite similar to the procedure `MainLoop` of gSpan. It starts by removing all infrequent vertices and edges. Next, for each frequent edge its TID list is initialized before the `gRed-SubgraphMining` is invoked. At the end of each iteration the edge is dropped from the database, i.e. it will not be used as possible extensions in the next iterations.

The procedure `gRed-SubgraphMining` recursively generates all candidate codes (graphs), this process is done while the generated code is frequent. Firstly, for each minimum DFS code s its extension set $ER_0(s)$ is calculated using the procedure `gRed-Enumerate`. The propositions 3 and 4 are used in `gRed SubgraphMining`

Procedure gRed-MainLoop(D, δ, S)

Input: D - database, δ - support threshold
Output: S - mining results

```
1  Remove infrequent vertices and edges;
2  S ← S^1 ← all frequent 1-edge codes
3  forall code s ∈ S^1 do
4  |  Initialize the TID list s.L by the graphs which contains the edge of s;
5  |  gRed-SubgraphMining(D,s,δ,S);
6  |  D ← D \ e;
7  |  if |D| < δ then break;
8  end
```

Procedure gRed-SubgraphMining(D, s, δ, S)

```
1  S ← S ∪ {s};
2  gRed-Enumerate(D, s, RE);
3  forall extensions set E, might be RE.B_i or RE.F_i do
4  |  Scan the first elements in E according to the order ≺_e, removing the
   |  non-minimal extensions;
5  |  forall extension e ∈ E do
6  |  |  if s ◇ e.support ≥ δ then gRed-SubgraphMining(D, s ◇ e, δ, S);
7  |  end
8  end
```

Procedure gRed-Enumerate(D, s, RE)

```
1  forall graph g ∈ s.L do
2  |  Enumerate the next occurrence of s in g;
3  |  forall right most extension e of s do
4  |  |  i ← e.i;
5  |  |  f_i ← the forward edge starting in i and lies in the right most path;
6  |  |  if e is a forward edge and e ≥_l f_i then        /* See proposition 1 */
7  |  |  |  s ◇ e.L ← s ◇ e.L ∪ {g};
8  |  |  |  RE.F_i ← RE.F_i ∪ {e};
9  |  |  end
10 |  |  if e is a backward edge and e^-1 ≥_l f_i then    /* See proposition 2 */
11 |  |  |  s ◇ e.L ← s ◇ e.L ∪ {g};
12 |  |  |  RE.B_i ← RE.B_i ∪ {e};
13 |  |  end
14 |  end
15 |  if any occurrence of s in g are not covered then goto line 2;
16 end
```

Fig. 3. General description of gRed algorithm

for reducing the number of expensive minimality tests regarding gSpan and guaranteeing the completeness in the mining result. The minimality test in line 4 is performed using the same `isMin`(s) function used in gSpan (see section 2.1). In

gRed the `isMin`(s) function is used only for the first extensions of each $B_i(s)$ or $F_i(s)$ while in gSpan the test is performed for every child of s.

In the procedure `gRed-Enumerate`, all occurrences of s in each graph of the TID list $s.L$ are enumerated. Thus, all the possible extensions of s in the database are generated. Using the propositions 1 and 2, some non-minimum extensions are filtered. For the specific sub-graph isomorphism testing procedure of line 2, we use the same enumerating engine proposed by gSpan in [12].

4 Experimental Results

All the experiments were done using an Intel Core 2 Duo PC at 2.2 GHz with 2 GB of RAM. We compare gRed only against gSpan and Gaston; both were implemented in a common Java framework [10] which is distributed under GNU license. Our implementation of gRed is compatible with this framework. The SUN Java Virtual Machine (JVM) 1.5.0 was used to run the algorithms.

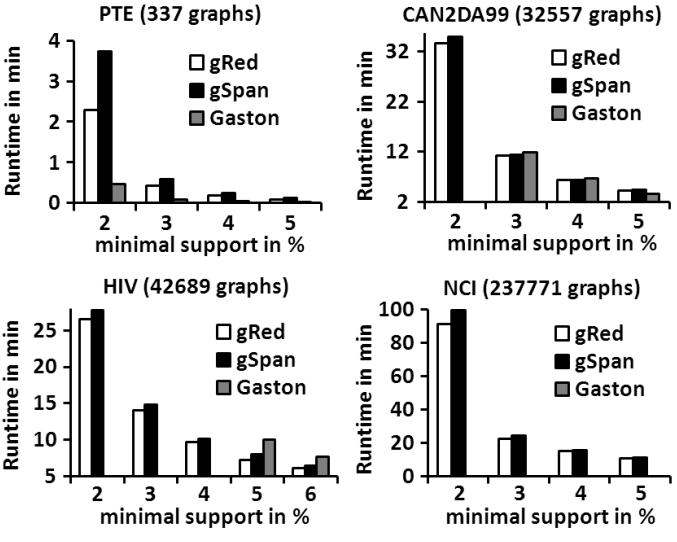


Fig. 4. Runtime with datasets PTE, CAN2DA99, HIV and NCI varying the minimum support

First of all, in order to show that the algorithm implementations get approximately the same results as those published in [8,10] we retested the algorithms as in those works. The performance was retested using PTE [9] and CAN2DA99[1] datasets. The results in both datasets are included adding the performance of gRed.

[1] http://dtp.nci.nih.gov/docs/cancer/cancer_data.html

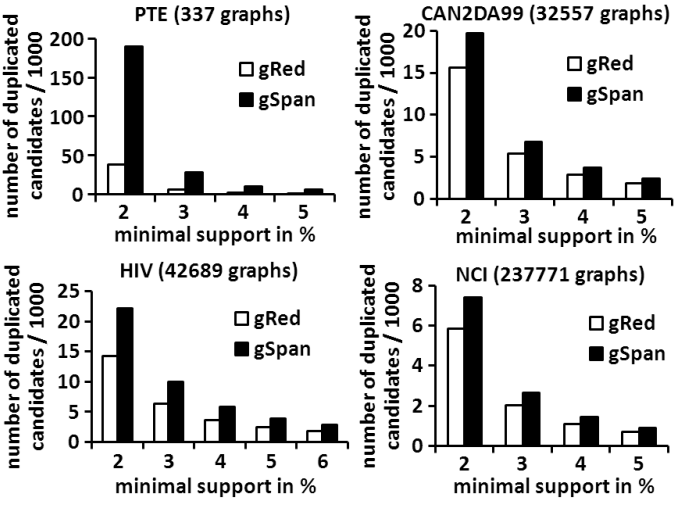


Fig. 5. The number of duplicated candidates found in datasets PTE, CAN2DA99, HIV and NCI varying the minimum support

The NCI[2] and HIV[3] datasets are used to determine how the algorithms scale when the database size increases. Commonly, these four datasets have been used in different works for performance evaluations [10].

The runtime for the algorithms was recorded varying the support threshold for the four datasets. A comparison of gRed, gSpan and Gaston regarding its execution times are showed in Fig. 4. In order to illustrate how the algorithms scale with a lot of candidates, only low support thresholds were considered. The runtime rises for Gaston on large databases (CAN2DA99, HIV and NCI) for these minimal supports. Besides, Gaston was unable to complete the execution for low minimal supports (less than 3% in CAN2DA99, 5% in HIV and 6% in NCI) due to memory requirements. Gaston needed much more memory than the other tested algorithms (see Fig. 6). However, in the smallest database (PTE), the best results were achieved by Gaston. The best runtimes on the large databases were obtained by gRed and gSpan for the evaluated support thresholds.

As we can see, gRed beats gSpan in all tests. It is known that much of the time consumption in gSpan is used by subgraph isomorphism tests during the candidate enumeration process. Since gRed also uses this kind of tests, it has similar behavior. However, gRed shows significant improvements when the database is large. In PTE the improvement achieved by gRed is even greater since the number of duplicate candidates declines considerably (see Fig. 5).

[2] http://cactus.nci.nih.gov/ncidb2/download.html
[3] http://dtp.nci.nih.gov/docs/aids/aids_data.html

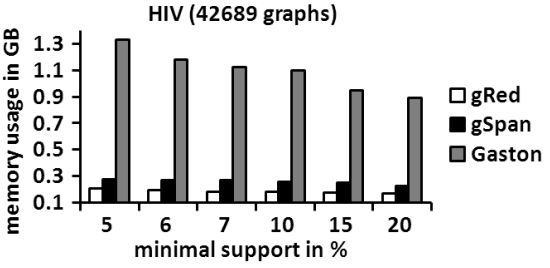


Fig. 6. Memory usage on the HIV dataset varying the minimum support

The algorithms gRed and gSpan are also compared regarding the number of duplicated candidates in Fig. 5. The pruning strategies used to minimize the number of duplicates in Gaston is very different from those used by gRed and gSpan. Gaston is not included in this comparison to highlight the differences between gRed and gSpan. The number of duplicates in all cases were significantly reduced by gRed; even in PTE for minimal support = 2, it reduces almost 60% of the duplicates regarding gSpan. This improvement corresponds to the runtimes of Fig. 4; nevertheless, the runtime improvement was not even greater because subgraph isomorphism tests are time-consuming. This result suggests that, if gRed is combined with the evaluation strategies of the other algorithms (for example the embedding structures used in Gaston), we might achieve better runtime scores.

The memory consumption was recorded varying the support threshold on the HIV dataset (see Fig. 6). We choose HIV in order to show an example of the memory problem of Gaston for low minimal support in correspondence to the runtimes of Fig. 4. The improvement of gRed regarding memory requirement can be appreciated in Fig. 6.

5 Conclusions

In this paper, a new algorithm called gRed, for frequent connected subgraph mining, was introduced. Novel properties of the DFS code, that allow to reduce the number of candidates during the mining process, were studied and implemented. In this research, we show that the DFS code has not been sufficiently studied and new properties can be found to improve the mining process.

We compared gRed against two reported algorithms. The experimentation showed that our proposal overcome gSpan in every tests. In the experiments, gRed and gSpan achieved better performance evaluations than Gaston for low minimal support when databases are large.

As future work, we are going to develop hybrid approaches of gRed in combination with evaluation strategies of other algorithms like Gaston.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB 1994), Santiago, Chile, pp. 487–499 (1994)
2. Borgelt, C., Berthold, M.R.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In: Proceedings of the 2002 International Conference on Data Mining (ICDM 2002), Maebashi, Japan, pp. 211–218 (2002)
3. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent Pattern Mining: Current Status and Future Directions. In: Data Mining and Knowledge Discovery (DMKD 2007), 10th Anniversary Issue, vol. 15(1), pp. 55–86 (2007)
4. Huan, J., Wang, W., Prins, J.: Efficient Mining of Frequent Subgraph in the Presence of Isomorphism. In: Proceedings of the 2003 International Conference on Data Mining (ICDM 2003), Melbourne, FL, pp. 549–552 (2003)
5. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
6. Inokuchi, A., Washio, T.: Nishimura and K., Motoda, H.: A Fast Algorithm for Mining Frequent Connected Subgraphs, Technical Report RT0448. In IBM Research, Tokyo Research Laboratory, pp. 10 (2002)
7. Kuramochi, M., Karypis, G.: Frequent Subgraph Discovery. In: Proceedings of the 2001 International Conference on Data Mining (ICDM 2001), San Jose, CA, pp. 313–320 (2001)
8. Nijssen, S., Kok, J.: A Quickstart in Frequent Structure Mining can Make a Difference. In: Proceedings of the 2004 ACM SIGKDD International Conference on Kowledge Discovery in Databases (KDD 2004), Seattle, WA, pp. 647–352 (2004)
9. Srinivasan, A., King, R.D., Muggleton, S.H., Sternberg, M.: The Predictive Toxicologic Evaluation Challenge. In: Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI 1997), pp. 1–6. Morgan-Kaufmann, San Francisco (1997)
10. Wörlein, M., Meinl, T., Fischer, I., Philippsen, M.: A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 392–403. Springer, Heidelberg (2005)
11. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proceedings of the 2002 International Conference on Data Mining (ICDM 2002), Maebashi, Japan, pp. 721–724 (2002)
12. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining, Expanded Version, UIUC Technical Report, UIUCDCS-R-2002-2296 (2002)

Unsupervised Riemannian Clustering of Probability Density Functions

Alvina Goh and René Vidal

Center for Imaging Science, Johns Hopkins University
3400 North Charles Street, Baltimore, MD 21218, USA

Abstract. We present an algorithm for grouping families of probability density functions (pdfs). We exploit the fact that under the square-root re-parametrization, the space of pdfs forms a Riemannian manifold, namely the unit Hilbert sphere. An immediate consequence of this re-parametrization is that different families of pdfs form different submanifolds of the unit Hilbert sphere. Therefore, the problem of clustering pdfs reduces to the problem of clustering multiple submanifolds on the unit Hilbert sphere. We solve this problem by first learning a low-dimensional representation of the pdfs using generalizations of local nonlinear dimensionality reduction algorithms from Euclidean to Riemannian spaces. Then, by assuming that the pdfs from different groups are separated, we show that the null space of a matrix built from the local representation gives the segmentation of the pdfs. We also apply of our approach to the texture segmentation problem in computer vision.

Keywords: Manifold learning, manifold clustering, probability density functions.

1 Introduction

Over the past few decades, there has been a huge explosion in the amount of readily available information. In order to be able to efficiently process this abundance of data, probability density functions (pdf) are often employed to model complex datasets. This has led to the development of various metrics to measure the similarities between different pdfs. *Information geometry* refers to the study of the intrinsic geometric structures in the manifold of pdfs. The Riemannian structure of the space of pdfs was first introduced in [1]. Since then, there have been major breakthroughs in the theory of information geometry [2], and also in the development of computational tools for clustering that utilize the geometric structure of the Riemannian manifold [3,4,5].

An area in which probability density functions are commonly used in computer vision is texture analysis. Segmentation of different textures remains important for present-day applications and is the focus of considerable effort in the field. For example, it is vital to be able to perform automatic segmentation of different land cover from remotely sensed images in environmental management, as it is tedious for photo-interpreters to classify landscape manually. It is well-known that by convolving the image with a set of filters, it is possible to obtain a spectral pdf of the image which indicates the texture characteristics. Therefore, given a set of images with different textures, it is often desirable to be able to automatically segment the textures into similar classes

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 377–392, 2008.

by clustering the pdfs into different families. For arbitrary textures, we do not necessarily know the description of the pdfs, though it is reasonable to assume that similar textures will have similar pdfs.

The question we address in this paper is the following. Given a set of pdfs, how do we develop a computationally simple framework that allows us to group the pdfs into similar families? Since these pdfs are determined from data, they are non-parametric in nature. Therefore, in order to compute distances between two arbitrary pdfs, we impose a Riemannian structure on the manifold of pdfs. Unlike traditional clustering methods, we will not assume that the pdfs within each group are centered around a collection of cluster centers in the manifold. Instead, we will assume that the different groups of pdfs form different submanifolds in the Riemannian space. Therefore, we aim to develop a framework that exploits the Riemannian structure of the space of pdfs to cluster a given set of arbitrary pdfs into similar groups.

Our clustering framework makes use of nonlinear dimensionality reduction techniques. Nonlinear dimensionality reduction (NLDR) refers to the problem of finding a low-dimensional representation for a set of points lying in a nonlinear manifold embedded in a high-dimensional space. Existing NLDR techniques can be categorized into two main groups: global and local techniques. Global techniques attempt to preserve global properties of the data lying in a submanifold, similar to what Principal Component Analysis (PCA) [6] attempts to preserve for data lying in a linear subspace. Two of the best-known examples of this family of algorithms are ISOMAP [7] and Kernel PCA (KPCA) [8]. Local techniques are however based on the preservation of local properties which are obtained from the small neighborhoods around the datapoints. The key idea of such techniques is that by preserving the local properties of the data, one can also retain the global properties of the data. Locally linear embedding (LLE) [9], Laplacian eigenmaps (LE) [10] and Hessian LLE [11] fall under this category of algorithms. In this paper, we chose to use local NLDR techniques such as LLE, LE and HLLE. We will show that the segmentation of the data can be obtained from the null space of a matrix built from the local representation. This is a property that local NLDR methods offer but global NLDR methods such as ISOMAP do not.

Paper contributions The main contribution of this paper is the development of a framework for clustering different families of pdfs. By choosing the square-root representation, we reduce the problem to one of clustering data lying in different submanifolds of a unit sphere. As in our previous work [12], we learn a local representation of the data using generalization of the three NLDR techniques, namely Laplacian Eigenmaps (LE) [10], Locally Linear Embedding (LLE) [9], and Hessian LLE (HLLE) [11], from Euclidean to Riemannian spaces. We show that the null space of a matrix built from the local representation gives the segmentation of the pdfs. Our method is computationally simple and performs automatic segmentation without requiring user interaction.

2 Review of Local Nonlinear Dimensionality Reduction Methods in Euclidean Spaces

In this section, we review three local NLDR algorithms. Let $X = \{\boldsymbol{x}_i \in \mathcal{M}\}_{i=1}^n$ be a set of n data points sampled from a d-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$, $d \ll D$.

We assume that the n points are k**-connected**, i.e., for any two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$ there is an ordered sequence of points in X having $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as endpoints, such that any two consecutive points in the sequence have at least one k-nearest neighbor in common. The goal of dimensionality reduction is to find a set of vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$, such that nearby points remain close and distant points remain far.

Locally Linear Embedding (LLE) [9] assumes that the local neighborhood of a point in the manifold can be well approximated by the affine subspace spanned by the k-nearest neighbors of the point, and finds a low-dimensional embedding of the data based on these affine approximations. **Laplacian Eigenmaps (LE)** [10] are based on computing the low dimensional representation that best preserves *locality* instead of *local linearity* in LLE. **Hessian LLE (HLLE)** [11] bears substantial resemblance to LLE and LE, with the main difference being that the local neighborhood is represented by the tangent space at each point and the Laplacian matrix is replaced by the Hessian matrix. The main steps of these local NLDR algorithms are as follows:

1. *Nearest neighbor search:* For each data point $\boldsymbol{x}_i \in X$, find its k nearest neighbors (kNN) $\{\boldsymbol{x}_{i_j}\}_{j=1}^k$ according to the Euclidean distance.
2. *Construction of similarity matrix:* Construct a weighted graph whose elements encode the local geometry of the data. Define a similarity matrix M based on these weights. M is symmetric and positive semidefinite.
3. *Sparse eigenvalue problem*: Obtain the embedding coordinates, i.e., the columns of $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d}$, from the d (generalized) eigenvectors of the matrix M associated with its second to $(d+1)$-th smallest (generalized) eigenvalues. The vector of all ones, $\mathbf{1} \in \mathbb{R}^n$, is a eigenvector of M associated with eigenvalue 0.

We will now describe the construction of M for each NLDR algorithm in more detail.

Calculation of M in LLE

1. *Weight matrix*: Find a matrix of weights $W \in \mathbb{R}^{n \times n}$ whose entries W_{ij} minimize the reconstruction error

$$\varepsilon(W) = \sum_{i=1}^{n} \| \sum_{j=1}^{n} W_{ij} \boldsymbol{x}_j - \boldsymbol{x}_i \|^2 = \sum_{i=1}^{n} \text{dist}^2(\widehat{\boldsymbol{x}}_i, \boldsymbol{x}_i) \tag{1}$$

   subject to the constraints (i) $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$ and (ii) $\sum_{j=1}^n W_{ij} = 1$. In (1), $\widehat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \sum_{j=1}^n W_{ij} \overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j}$ is the linear interpolation of $\boldsymbol{x}_i$ and its kNN. The solution to this problem can be computed as

$$\begin{bmatrix} W_{i\,i_1} & W_{i\,i_2} & \cdots & W_{i\,i_k} \end{bmatrix} = \frac{\mathbf{1}^\top C_i^{-1}}{\mathbf{1}^\top C_i^{-1} \mathbf{1}} \in \mathbb{R}^{1 \times k}, \tag{2}$$

   where $\mathbf{1} \in \mathbb{R}^n$ is the vector of all ones, and $C_i \in \mathbb{R}^{k \times k}$ is the local Gram matrix at $\boldsymbol{x}_i$, i.e., $C_i(j,l) = (\boldsymbol{x}_j - \boldsymbol{x}_i) \cdot (\boldsymbol{x}_l - \boldsymbol{x}_i)$.
2. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the error

$$\phi(Y) = \sum_{i=1}^{n} \| \mathbf{y}_i - \sum_{j=1}^{n} W_{ij} \mathbf{y}_j \|^2 = \text{trace}(Y^\top M Y), \tag{3}$$

subject to the constraints (i) $\sum_{i=1}^{n} \mathbf{y}_i = \mathbf{0}$ and (ii) $\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i \mathbf{y}_i^\top = I$. The solution to this optimization problem is given by the d eigenvectors of $M = (I - W)^\top (I - W)$ associated with its second to $(d+1)$-th smallest eigenvalues.

Calculation of M in LE

1. *Weight matrix*: Construct a matrix of weights $W \in \mathbb{R}^{n \times n}$

$$W_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2) \tag{4}$$

subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. The entries of W, W_{ij}, measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

2. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the error

$$\phi(Y) = \sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij} = \text{trace}(Y^\top M Y) \tag{5}$$

subject to the constraints (i) $Y^\top D\mathbf{1} = \sum_{i=1}^{n} D_{ii}\mathbf{y}_i = \mathbf{0}$ (weighted low-dimensional coordinates centered at the origin) and (ii) $Y^\top DY = I$ (weighted low-dimensional coordinates having unit covariance). In Eq. (5), $M = D - W$ is the graph Laplacian matrix and D is a diagonal matrix whose entries are given by $D_{ii} = \sum_j W_{ij}$. The solution to this optimization problem is given by the d generalized eigenvectors of (M, D) associated with its second to $(d+1)$-th smallest generalized eigenvalues.

Calculation of M in HLLE

1. *Tangent coordinates*: For each data point $\boldsymbol{x}_i$, let $\{\boldsymbol{x}_{i_j}\}_{j=1}^k$ be its kNN. Form the D by D covariance matrix $\text{cov}(\boldsymbol{x}_i) = \frac{1}{k}\sum_{j=1}^{k}(\boldsymbol{x}_{i_j} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{i_j} - \bar{\boldsymbol{x}}_i)^\top$, where $\bar{\boldsymbol{x}}_i$ is the mean of the kNN. Perform an eigenanalysis of the matrix $\text{cov}(\boldsymbol{x}_i)$ to obtain the d eigenvectors $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$. The tangent coordinates of the kNN are given by the d columns of the $k \times d$ matrix V given below, where $p = 1, \ldots, k$ and $q = 1, \ldots, d$

$$V_{pq} = (\boldsymbol{x}_{i_p} - \bar{\boldsymbol{x}}_i)^\top \mathbf{u}_q = \langle \boldsymbol{x}_{i_p} - \bar{\boldsymbol{x}}_i, \mathbf{u}_q \rangle. \tag{6}$$

2. *Objective function*: The embedding vectors are obtained based on the null vectors of a matrix M that indicates the Hessian quadratic cost. While we refer the reader to [11] for details on the estimation of M, the basic principle is as follows. We first locally estimate a Hessian operator h^i at each point $\boldsymbol{x}_i$ in the manifold in a least squares sense. In particular, consider a smooth function $f : \mathcal{M} \to \mathbb{R}$. We evaluate the function at all kNN of a point $\boldsymbol{x}_i$ in the manifold and stack these entries into a vector $\mathbf{f}_i$. It can be shown that $h^i \mathbf{f}_i$ approximates the entries of the Hessian, whose (p, q)-th entry is given by $\frac{\partial^2 f}{\partial V_p \delta V_q}$. These local estimates are then used to obtain an empirical estimate of the (i, j)-th entry of M as

$$M_{i,j} = \sum_l \sum_r ((h^l)_{r,i}(h^l)_{r,j}). \tag{7}$$

The embedding coordinates are then found by selecting a basis for the space spanned by d eigenvectors of M associated with its second to $(d+1)$-th smallest eigenvalues

with the restriction that it provides an orthonormal basis to a specific fixed neighborhood $\mathcal{N}$. Let U denote the $n \times d$ matrix associated with the second to $(d+1)$-th smallest eigenvectors where $U_{l,r}$ is the l-th entry in the r-th eigenvector of M. The embedding coordinates is obtained as $UR^{-\frac{1}{2}}$, where $R_{r,s} = \sum_{j \in \mathcal{N}} U_{j,r} U_{j,s}$.

3 Clustering Submanifolds of a Riemannian Space

In this section, we present an algorithm for clustering and dimensionality reduction on Riemannian manifolds. We first present a brief summary of the theory of Riemannian manifolds in §3.1. For a more complete description, we refer the reader to [13]. We then illustrate how to extend existing NLDR algorithms to Riemannian manifolds in §3.2 by adopting the framework in [12]. Finally, in §3.3, we show that by making use of the mappings generated by NLDR, the problem of manifold clustering reduces to a central clustering problem, as proved in [12].

3.1 Review of Riemannian Manifolds

The NLDR techniques presented in §2 are applicable only in the presence of *one* manifold with *unknown structure*. Every operation is approximated by the corresponding Euclidean operation as the metric is unknown. However, for Riemannian manifolds with well-studied geometries, closed-form formulae for Riemannian operations are often available. The question now is to extend NLDR techniques for Riemannian manifolds in a way that takes into consideration the appropriate Riemannian structure. For this purpose, we adopt the framework developed in our previous work [12]. In this section, we will give an overview of Riemannian theory and show how the various operations such as interpolation on the manifold and computation of the mean and principal components are carried out.

A differentiable manifold $\mathcal{M}$ of dimension d is a topological space that is homeomorphic to the Euclidean space $\mathbb{R}^d$. Fig. 1 shows an example of a two-dimensional manifold, a smooth surface living in $\mathbb{R}^3$. The tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$ is the vector space that contains the tangent vectors to all 1-D curves on $\mathcal{M}$ passing through $\boldsymbol{x}$. A *Riemannian metric* on a manifold $\mathcal{M}$ is a bilinear form which associates to each point $\boldsymbol{x} \in \mathcal{M}$, a differentiable varying inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{x}}$ on the tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$. The norm of a vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$ is denoted by $\|\boldsymbol{v}\|_{\boldsymbol{x}}^2 = \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{x}}$. The Riemannian distance $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lying in the manifold is defined as the minimum length over all possible smooth curves on the manifold between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The geodesic curve from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$, γ, is the smooth curve with minimum length.

Given a tangent vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$, locally there exists a unique geodesic $\gamma_{\boldsymbol{v}}(t)$ starting at $\boldsymbol{x}$ with initial velocity $\boldsymbol{v}$, and this geodesic has constant speed equal to $\|\boldsymbol{v}\|_{\boldsymbol{x}}$. The *exponential map*, $\exp_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{M} \rightarrow \mathcal{M}$ maps a tangent vector $\boldsymbol{v}$ to the point in the manifold that is reached at time 1 by the geodesic $\gamma_{\boldsymbol{v}}(t)$. The inverse of $\exp_{\boldsymbol{x}}$ is the *logarithm map* and denoted by $\log_{\boldsymbol{x}} : \mathcal{M} \rightarrow T_{\boldsymbol{x}}\mathcal{M}$. For two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the manifold $\mathcal{M}$, the tangent vector to the geodesic curve from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ is defined as $\boldsymbol{v} = \overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j} = \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)$, and the exponential map takes $\boldsymbol{v}$ to the point $\boldsymbol{x}_j = \exp_{\boldsymbol{x}_i}(\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j))$. In addition, $\gamma_{\boldsymbol{v}}(0) = \boldsymbol{x}_i$ and $\gamma_{\boldsymbol{v}}(1) = \boldsymbol{x}_j$. The Riemannian distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \| \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \|_{\boldsymbol{x}_i}$. Linear

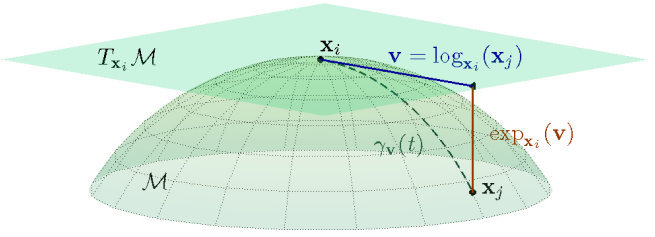


Fig. 1. An example of a two-dimensional manifold. The tangent plane at $\boldsymbol{x}_i$, together with the exponential and logarithm maps relating $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, are also shown.

geodesic interpolation makes use of the exponential and logarithm maps and is given by $\widehat{\boldsymbol{x}} = \exp_{\boldsymbol{x}_i}(w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$, $w \in [0,1]$. Finally, the Riemannian metric, exponential and logarithm maps depend on the point $\boldsymbol{x}$ under consideration, hence the subscripts reflecting this dependency.

We will now briefly summarize how to calculate the mean and principal components of data points lying in a manifold. As defined by Fréchet in [14] and used in several recent works [15,16], the intrinsic mean $\overline{\mathbf{x}}$ is defined as the solution to the following minimization problem

$$\overline{\mathbf{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{M}} \sum_{i=1}^{n} \operatorname{dist}(\boldsymbol{x}, \boldsymbol{x}_i)^2 = \arg\min_{\boldsymbol{x}\in\mathcal{M}} \sum_{i=1}^{n} \|\log_{\boldsymbol{x}}(\boldsymbol{x}_i)\|_{\boldsymbol{x}}^2. \tag{8}$$

Note that, unlike in the Euclidean case, in general there is no closed form for $\overline{\mathbf{x}}$. Moreover, there is no guarantee that $\overline{\mathbf{x}}$ exists or is unique. However, one can show the existence and uniqueness of $\overline{\mathbf{x}}$ [17] by assuming that the data lie in a small enough neighborhood, i.e., the maximum distance between any $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is small enough. Furthermore, $\overline{\mathbf{x}}$ can be computed as shown in Algorithm 1.

Algorithm 1. (Intrinsic Mean)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$, a predefined threshold ϵ, maximum number of iterations T,

1. Initialize $t = 1$, $\overline{\mathbf{x}}_1 = \boldsymbol{x}_i$ for a random i, $\boldsymbol{v} \neq 0 \in T_{\overline{\mathbf{x}}_1}\mathcal{M}$.
2. While $t \leq T$ or $\|\boldsymbol{v}\|_{\overline{\mathbf{x}}} \geq \epsilon$,
   (a) Compute tangent vector $\boldsymbol{v} = \frac{1}{n}\sum_{i=1}^{n} \log_{\overline{\mathbf{x}}_t}(\boldsymbol{x}_i)$.
   (b) Set $\overline{\mathbf{x}}_{t+1} = \exp_{\overline{\mathbf{x}}_t}(\boldsymbol{v})$

Given $\overline{\mathbf{x}}$, the calculation of principal components on a Riemannian manifold is not as straightforward as in the Euclidean case. It involves projecting a point onto a geodesic curve, which is also defined as a minimization problem for which existence and uniqueness are not ensured [15]. Again, by making the assumptions that the data lie in a small neighborhood, the projection can be shown to be unique. In [15], it is shown that finding principal components boils down to doing PCA in the tangent vectors $\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i) \in T_{\overline{\mathbf{x}}}\mathcal{M}$

Table 1. Comparison of Euclidean and Riemannian operations, where $\{\boldsymbol{x}_i\}_{i=1}^n$, are data points

Operation	Euclidean	Riemannian
Subtraction $\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\boldsymbol{x}_j - \boldsymbol{x}_i$	$\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)$
Addition $\boldsymbol{x}_j$	$\boldsymbol{x}_i + \overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\exp_{\boldsymbol{x}_i}(\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$
Distance $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$	$\|\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}\| = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|$	$\|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i} = \sqrt{\langle \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j), \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\rangle_{\boldsymbol{x}_i}}$
Mean $\overline{\mathbf{x}}$	$\sum_{i=1}^n \overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i} = 0$	$\sum_{i=1}^n \log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i) = 0$
Covariance $\mathrm{cov}(\boldsymbol{x})$	$\frac{1}{n}\sum_{i=1}^n (\overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i})(\overrightarrow{\overline{\mathbf{x}}\boldsymbol{x}_i})^\top$	$\frac{1}{n}\sum_{i=1}^n (\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i))(\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i))^\top$
Linear interpolation $\widehat{\boldsymbol{x}}$	$\boldsymbol{x}_i + w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$	$\exp_{\boldsymbol{x}_i}(w\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j})$

Algorithm 2. (Principal Geodesic Analysis)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$,

1. Compute intrinsic mean $\overline{\mathbf{x}}$ as in Algorithm 1.
2. Calculate the tangent vectors $\boldsymbol{v}_i = \log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i)$about $\overline{\mathbf{x}}$.
3. Construct the sample covariance matrix $\mathrm{cov}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n \boldsymbol{v}_i\boldsymbol{v}_i^\top$.
4. Perform eigenanalysis of the matrix $\mathrm{cov}(\boldsymbol{x})$, with the eigenvectors $\{\mathbf{u}_i\}_{i=1}^d$ giving the principal directions. $\{\mathbf{u}_i\}_{i=1}^d$ forms an orthonormal basis for $T_{\overline{\mathbf{x}}}\mathcal{M}$.

about the mean $\overline{\mathbf{x}}$. This algorithm, known as Principal Geodesic Analysis (PGA), is summarized in Algorithm 2. Table 1 compares the standard operations in Euclidean and Riemannian spaces.

3.2 Extending NLDR to Riemannian Manifolds

Notice that the information about the local geometry of the manifold is essential only in the first two steps of each algorithm and therefore, modifications are made only to these two stages. The key issues are how to select the kNN and how to compute the matrix M representing the local geometry. As shown in [12], the former is straightforward, while the latter requires some thought. Given M, calculating the low-dimensional representation remains the same as in the Euclidean case. We let $X = \{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^n$ be a set of n data points sampled from a known Riemannian manifold.

Selection of the Riemannian kNN The first step of any NLDR algorithm is the computation of the kNN associated with each data point. We define the kNN of $\boldsymbol{x}_i$ by incorporating the Riemannian distance, i.e., *the kNN of $\boldsymbol{x}_i$ are the k data points $\boldsymbol{x}_j$ that minimize* $\|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i}$.

Riemannian Calculation of M for LLE The second step of LLE is to compute the matrix of weights $W \in \mathbb{R}^{n\times n}$. In order to do so, we will answer two main questions: 1) how does one express a point as a linear combination of its neighbors? and 2) what is the reconstruction cost? First of all, we know that from §3.1 that

$$\widehat{\boldsymbol{x}}_{Riem,i} = \exp_{\boldsymbol{x}_i}(\sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)). \tag{9}$$

is the geodesic linear interpolation of $\boldsymbol{x}_i$ by $\{\boldsymbol{x}_j\}_{j=1}^n$. Now, instead of minimizing the Euclidean error, we rewrite (1) to minimize the Riemannian reconstruction error and make use of the fact that exp and log are inverse mappings. Therefore, we have

$$\varepsilon_{Riem}(W) = \sum_{i=1}^{n} \big\| \log_{\boldsymbol{x}_i}(\widehat{\boldsymbol{x}}_{Riem,i}) \big\|_{\boldsymbol{x}_i}^2 = \sum_{i=1}^{n} \big\| \sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \big\|_{\boldsymbol{x}_i}^2 \tag{10}$$

subject to $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a kNN of $\boldsymbol{x}_i$ and $\sum_j W_{ij} = 1$. Using similar manipulations as in the Euclidean case, the optimal weights are obtained as in (2), with the local Gram matrix $C_i \in \mathbb{R}^{k\times k}$ defined as

$$C_i(j,l) = \langle \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j), \log_{\boldsymbol{x}_i}(\boldsymbol{x}_l) \rangle_{\boldsymbol{x}_i}. \tag{11}$$

M is then $(I-W)^\top(I-W)$.

Riemannian Calculation of M for LE Here, instead of attempting to write each data point as a linear combination of its kNN, we find a matrix of weights $W \in \mathbb{R}^{n\times n}$ whose entries W_{ij} measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as in (4). Therefore, modifying LE for Riemannian manifolds is less involved than in the case of LLE. Instead of using $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2)$ as in (4), we construct the weight matrix W using the Riemannian distance as

$$W_{ij} = \exp\Big(- \frac{\mathrm{dist}_{Riem}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{\sigma^2} \Big) = \exp\Big(- \frac{\| \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \|_{\boldsymbol{x}_i}^2}{\sigma^2} \Big) \tag{12}$$

subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. As before, $M = D - W$ and D is a diagonal matrix, where $D_{ii} = \sum_j W_{ij}$.

Riemannian Calculation of M for HLLE The second step of HLLE involves computing the tangent coordinates for each $\boldsymbol{x}_i$ by applying Euclidean PCA to its neighbors. This implicitly assumes that these local points lie in a subspace. This assumption is no longer valid if $\boldsymbol{x}_i$ and its kNN lie in a Riemannian manifold. From §3.1, we know that in this case, calculating the principal geodesic components and the projection coordinates is not as simple as doing Euclidean PCA. There is a need to incorporate the correct Riemannian metric, mean and covariance matrix.

Again, let $\{\boldsymbol{x}_{i,j}\}_{j=1}^k$ denote the set of k-nearest neighbors of $\boldsymbol{x}_i$. First we calculate the intrinsic mean $\bar{\boldsymbol{x}}_i$ of the kNN (Algorithm 1). Next, we find the tangent vectors $\boldsymbol{v}_j = \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,j})$ about $\bar{\boldsymbol{x}}_i$ and the geodesic principal directions $\{\mathbf{u}_q\}_{q=1}^d$ using PGA (Algorithm 2). Since $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$ is an orthonormal basis for $T_{\bar{\boldsymbol{x}}_i}\mathcal{M}$, we will rewrite the projection operator in (6) using the Riemannian metric. Thus the tangent coordinates of the kNN are given by the $k \times d$ matrix V, where

$$V_{pq} = \langle \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,p}), \mathbf{u}_q \rangle_{\bar{\boldsymbol{x}}_i},\ p = 1,..,k,\ q = 1,..,d. \tag{13}$$

Once the tangent coordinates are found, the estimation of the Hessian matrix M is the same as in the Euclidean case (7).

Calculation of the Embedding Coordinates The last step of NLDR is to find a Euclidean low-dimensional representation of the data points. As this step is independent of the Riemannian structure, one can find the embedding coordinates as described in §2. That is, the embedding coordinates are obtained based on the d (generalized) eigenvectors of the matrix M associated with its second to $(d+1)$-th smallest (generalized) eigenvalues. Finally, notice that if the Riemannian operations are available in closed-form, then extending NLDR to Riemannian manifolds do not require significant additional computational complexity.

3.3 Local Riemannian Manifold Clustering

In this section, we review the extension of NLDR algorithms for the purpose of clustering data lying in m submanifolds of a Riemannian space proposed [12]. We assume that the data is distributed in a k-disconnected union of m k-connected submanifolds of $\mathcal{M}$. Under this assumption, [12] shows that each of the m submanifolds will be mapped to a different point in $\mathbb{R}^m$. Proposition 1 states the main result of [12]. This proposition shows that in the case of a disconnected union of m k-connected submanifolds, the matrix M has at least m zero eigenvalues, whose eigenvectors give the clustering of the data. This is a generalized result that is applicable to Riemannian LLE, Riemannian LE and Riemannian HLLE. The interested reader is referred to [12] for the proof of Proposition 1.

Proposition 1 *Let $\{\boldsymbol{x}_i\}_{i=1}^n$ be a set of points drawn from a disconnected union of m k-connected d-dimensional submanifolds of a Riemannian manifold. Then, there exist m eigenvectors $\{\mathbf{u}_j\}_{j=1}^m$ in the null space of M such that $\mathbf{u}_j$ corresponds to the j-th group of points, i.e., $\mathbf{u}_{ij} = 1$ if the i-th data point is in the j-th group, and $\mathbf{u}_{ij} = 0$ otherwise.*

With real data, the assumption that the submanifolds are separated will obviously be violated. Therefore, the matrix M will be a perturbed version of the ideal case. However, it is well-known from perturbation theory [18] that if the perturbation is small or the eigengap is big, the eigenvectors $\boldsymbol{v}_j$ might not coincide completely with the indicator vectors $(\mathbf{0}, ., \mathbf{1}, ., \mathbf{0})^\top$ of the clusters, but do so up to a small error term. Hence, it is reasonable to expect that instead of mapping data points on m submanifolds to m points, the mapping will generate a collection of n points distributed around m cluster centers.

We see that there exists a mapping $g : \mathcal{M} \to \mathbb{R}^m$ that gives the membership of each point to the m submanifolds. This mapping is given by the rows of any basis for $\ker(M)$. However, notice that we do not necessarily obtain the set of membership vectors $\{\mathbf{u}_j\}$ when computing a basis for $\ker(M)$, but rather linear combinations of them, including the vector $\mathbf{1}$. In general, linear combinations of segmentation eigenvectors still contain the segmentation of the data. Hence, we can cluster the data into m groups by applying k-means to the columns of a matrix whose rows are the m eigenvectors in the null space of M. Algorithm 3 summarizes our dimensionality reduction and clustering algorithm for m submanifolds of a Riemannian space.

Algorithm 3. (Unsupervised Clustering and Dimensionality Reduction on Riemannian Manifolds)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$,

1. *Nearest neighbors:* Find the kNN of each data point $\boldsymbol{x}_i$ according to the Riemannian distance.
2. *Construction of M:* For each NLDR algorithm, construct the appropriate M described in §3.2.
3. *Clustering:* Compute the m eigenvectors $\{\mathbf{u}_j\}_{j=1}^m$ of M associated with its m smallest eigenvalues and apply k-means to the rows of $[\mathbf{u}_1, \cdots, \mathbf{u}_m]$ to cluster the data into m different groups.
4. *Low-dimensional embedding:* Apply NLDR to each group to obtain a low-dimensional embedding for each submanifold.

4 Riemannian Analysis of Probability Density Functions

In this section, we will show how to impose a Riemannian structure on the space of pdfs. We will adopt the work of [5], which proposes a "spherical" version of the Fisher-Rao metric that allows for closed-form expressions for the various Riemannian operations.

The class of constrained non-negative continuous functions under study here is the set of pdfs defined below. Without loss of generality, we can assume that these functions are defined on the interval $[0, T]$. Therefore, the set $\boldsymbol{\mathcal{P}}$ of pdfs is given by

$$\boldsymbol{\mathcal{P}} = \{\boldsymbol{p} : [0,T] \to \mathbb{R} | \forall s, \boldsymbol{p}(s) \geq 0, \int_0^T \boldsymbol{p}(s) ds = 1\}. \tag{14}$$

The question of how to regard the space of pdfs as a differential manifold endowed with a Riemannian metric and a family of affine connections has a long history behind it. Nevertheless, it remains an active and important research area. Treating statistical structures as geometric structures has the advantage that geometric structures remain invariant under coordinate transforms. [1] first introduces the Riemannian structure formed by the statistical manifold where each point in the manifold denotes a pdf. In addition, [1] also shows that the *Fisher-Rao* metric determines a Riemannian metric. The Fisher-Rao metric is later shown to be the *unique intrinsic metric* on the statistical manifold in [19]. This study of probability and information via differential geometry is known as *information geometry*. The reader is referred to the seminal work of [2] for a complete description.

We will consider the manifold $\boldsymbol{\mathcal{P}}$ of pdfs on the interval $[0, T]$. For any point $\boldsymbol{p}_i \in \boldsymbol{\mathcal{P}}$, the Fisher-Rao metric is defined as

$$\langle \boldsymbol{q}_j, \boldsymbol{q}_k \rangle_{\boldsymbol{p}_i} = \int_0^T \boldsymbol{q}_j(s) \boldsymbol{q}_k(s) \frac{1}{\boldsymbol{p}_i(s)} ds, \tag{15}$$

where $\boldsymbol{q}_j, \boldsymbol{q}_k \in T_{\boldsymbol{p}_i}(\boldsymbol{\mathcal{P}})$ are tangent vectors and $T_{\boldsymbol{p}_i}(\boldsymbol{\mathcal{P}})$ is the set containing the functions tangent to $\boldsymbol{\mathcal{P}}$ at the point $\boldsymbol{p}_i$. This representation turns out to be extremely difficult to work with as ensuring the geodesic between two elements lies on $\boldsymbol{\mathcal{P}}$ is not easy [5].

Even though the space $\boldsymbol{\mathcal{P}}$ turns out to be difficult to work with, we know that it is not the only possible representation for pdfs and in addition, we also know that

the Fisher-Rao metric is the only metric that is invariant to re-parameterizations (essentially coordinate transforms) of the functions [19]. There are many different re-parameterizations of pdfs that are equivalent representations. These include cumulative distribution functions $\int_0^s \boldsymbol{p}(t)dt$, log density functions $\log \boldsymbol{p}(s)$ and square-root density functions $\sqrt{\boldsymbol{p}(s)}$. Each of these parameterizations will lead to a different resulting manifold. Depending on the representation, the resulting Riemannian structure can have varying degrees of complexity and numerical techniques may be required to compute geodesics on the manifold. For example, [3] chooses the log density representation and uses a shooting technique to find geodesics on this space. However, this space has a complicated Riemannian structure and the numerical method used in [3] sometimes leads to large errors. Therefore, the natural question to ask now is, is it possible to use a re-parameterization such that the resulting manifold is simple and the Riemannian operations are easy, preferably closed-form, to compute? Once an efficient representation is found, the corresponding Fisher-Rao metric, which depends on the tangent vector, will then be used as the Riemannian metric.

In a recent work [5], it is proved that by using the square-root representation, the resulting manifold is a unit sphere in a Hilbert space with the Fisher-Rao metric being the usual $\mathbb{L}^2$ metric. Therefore, the various Riemannian operations such as geodesics, exponential maps, logarithmic maps are available in closed form. This is the most efficient representation found to date. The square-root density function is defined as $\boldsymbol{\psi} = \sqrt{\boldsymbol{p}}$, where $\boldsymbol{\psi}$ is assumed to be non-negative to ensure uniqueness. The space of such functions is defined as:

$$\boldsymbol{\Psi} = \{\boldsymbol{\psi} : [0, T] \rightarrow \mathbb{R} | \forall s, \boldsymbol{\psi}(s) \geq 0, \int_0^T \boldsymbol{\psi}^2(s)ds = 1\}. \tag{16}$$

From (16), it is easy to see that the functions $\boldsymbol{\psi}$ lie on a unit sphere. In addition, $\boldsymbol{\Psi}$ forms a convex subset of the unit sphere. The advantage of choosing the square-root density becomes immediately obvious, as many of the Riemannian expressions for the unit sphere are well-known and closed-form. By making use of the representation in (16), we can rewrite (15) and obtain the Fisher-Rao metric as

$$\langle \boldsymbol{v}_j, \boldsymbol{v}_k \rangle_{\boldsymbol{\psi}_i} = \int_0^T \boldsymbol{v}_j(s)\boldsymbol{v}_k(s)ds, \tag{17}$$

where $\boldsymbol{v}_j, \boldsymbol{v}_k \in T_{\boldsymbol{\psi}_i}\boldsymbol{\Psi}$ are tangent vectors. Now, for any two functions $\boldsymbol{\psi}_i, \boldsymbol{\psi}_j \in \boldsymbol{\Psi}$, the geodesic distance between these two points on a unit sphere is simply the angle between them, i.e.,

$$\text{dist}(\boldsymbol{\psi}_i, \boldsymbol{\psi}_j) = \cos^{-1}\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle = \cos^{-1}\big(\int_0^T \boldsymbol{\psi}_i(s)\boldsymbol{\psi}_j(s)ds\big), \tag{18}$$

where $\langle \cdot, \cdot \rangle$ is the normal dot product between points in the sphere under the $\mathbb{L}^2$ metric.

From the differential geometry of the sphere, the exponential map is defined as

$$\exp_{\boldsymbol{\psi}_i}(\boldsymbol{v}) = \cos(\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i})\boldsymbol{\psi}_i + \sin(\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i})\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i}}, \tag{19}$$

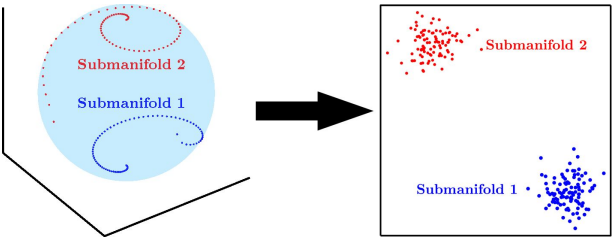


Fig. 2. Illustration of how probability density functions are clustered using our algorithm. Each point in the manifold denotes a pdf and different groups are mapped into different clusters.

where $\boldsymbol{v} \in T_{\boldsymbol{\psi}_i}(\boldsymbol{\Psi})$is a tangent vector at $\boldsymbol{\psi}_i$ and $\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{\psi}_i}} = (\int_0^T \boldsymbol{v}(s)\boldsymbol{v}(s)ds)^{\frac{1}{2}}$. In order to ensure that the exponential map is a bijective function, we restrict $\|\boldsymbol{v}\|_{\boldsymbol{\psi}_i} \in [0, \pi]$. The logarithm map from $\boldsymbol{\psi}_i$ to $\boldsymbol{\psi}_j$ is then given by

$$\overrightarrow{\boldsymbol{\psi}_i \boldsymbol{\psi}_j} = \log_{\boldsymbol{\psi}_i}(\boldsymbol{\psi}_j) = \frac{\mathbf{u}}{(\int_0^T \mathbf{u}(s)\mathbf{u}(s)ds)^{\frac{1}{2}}} \cos^{-1}\langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle, \tag{20}$$

with $\mathbf{u} = \boldsymbol{\psi}_j - \langle \boldsymbol{\psi}_i, \boldsymbol{\psi}_j \rangle \boldsymbol{\psi}_i$.

By substituting the closed-form formulae in this section into the respective operations in Algorithm 3, it is immediately clear that we are able to perform unsupervised clustering of probability density functions. Fig. 2 illustrates the overall idea of our approach.

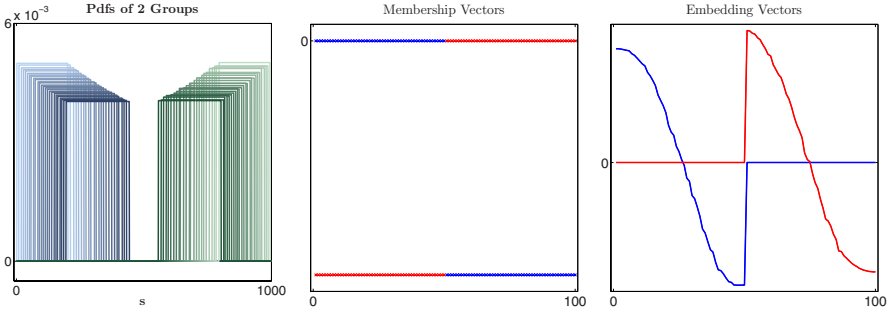
5 Experiments

In this section, we evaluate the performance of the proposed algorithm on both synthetic and real data. Experiments on synthetic data are performed on mixtures of uniform pdfs, while experiments on real data involve the segmentation of images based on texture.

5.1 Synthetic Examples

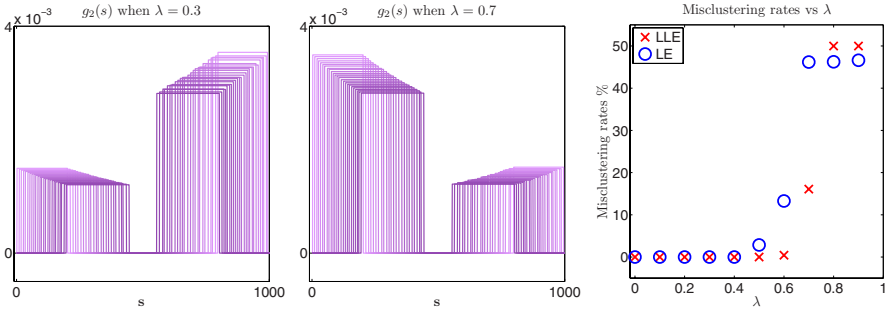
We will first evaluate the performance of Algorithm 3 for clustering two groups of uniform pdfs with 50 pdfs in each group. Fig. 3(a) shows these two groups of pdfs, with the first group f_1 in blue and the second group f_2 in green, defined on the interval $[0, 1000]$. Each different shade of blue or green denotes a different element of its group. Both groups are generated by shifting the intervals in which the probability is not zero and increasing the bandwidths. Let $\mathcal{U}[\alpha, \beta]$ be the uniform distribution on the interval $[\alpha, \beta]$. The pdfs in f_1 are $f_{1,i} = \mathcal{U}[a_i, b_i], i = 1, \ldots, 50$, where $a_i = 4(i-1) + \lambda_1$, $b_i = 195 + 5i + \lambda_2$. The pdfs in f_2 are $f_{2,j} = \mathcal{U}[c_j, d_j], j = 1, \ldots, 50$, where $c_j = 805 - 5j - \lambda_3$, $d_j = 1004 - 4j - \lambda_4$. $\{\lambda_k\}_{k=1}^4$ are drawn from $\mathcal{U}[0, 4]$. Figs. 3(b)-3(c) show that when we apply Riemannian LLE, the two smallest eigenvectors indicate the membership of each group whereas the next two eigenvectors are the embedding vectors.

Next, we validate the performance of our algorithm on two groups of pdfs, one with uniform distributions and the other one with mixtures of uniform distributions. The



(a) 2 groups of uniform pdfs (b) Membership eigenvectors (c) Embedding eigenvectors

Fig. 3. Applying Riemannian LLE to clustering two groups of uniform pdfs shown in (a). Both pdfs are uniform distributions with shifting centers and varying bandwidths.



(a) (b) (c)

Fig. 4. Clustering two groups of pdfs $g_1 = f_1$ and $g_2 = \lambda f_1 + (1 - \lambda) f_2$. Figs. 4(a)-4(b) show g_2 when $\lambda = 0.3, 0.7$. Fig. 4(c) shows the misclustering rates of LLE and LE when λ varies.

Fig. 5. Schmid filter bank that we use to generate the textons and in turn the histograms

groups have 50 pdfs each and are constructed as follows. Let f_1 be the first set of pdfs in blue shown in Fig. 3(a) and f_2 be the second set in green. We set the first group to $g_1 = f_1$ and the second group to $g_2 = \lambda f_1 + (1 - \lambda) f_2$, where $\lambda \in [0, 1]$. Figs. 4(a)-4(b) show g_2 when λ is equal to 0.3 and 0.7. Since noise is introduced in the generation of f_1 and f_2, we repeat this experiment over 500 trials. It is easy to see that when λ approaches 1, the group g_2 merges into g_1. From Fig. 4(c), we see that when λ is small, the misclustering rate is 0%. However, as λ approaches 1, the distance between g_1 and g_2 decreases and the misclustering approaches 50%.

Table 2. Misclustering rates in % for two-class segmentation

Algorithm	Set 1	Set 2	Set 3	Set 4
Riemannian LLE	0	0	1.63	0
Riemannian LE	0	0	19.68	22.9

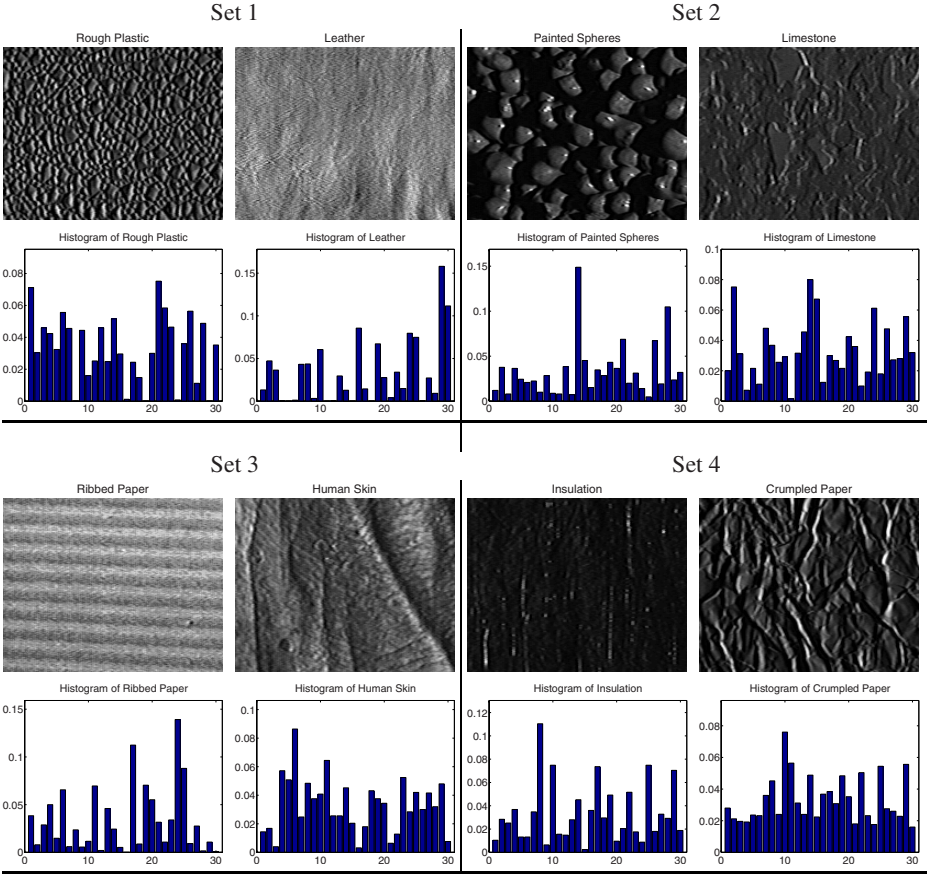


Fig. 6. Textures and corresponding histograms used in the two-class clustering experiments

We test our algorithm on 4 sets of data containing 2 different textures each. There are 92 images in each texture class. In these experiments, the number of nearest neighbors is set to 10. Fig. 6 shows these 4 sets with a typical example of the 2 different textures and the corresponding histograms in each set. Table 2 shows the misclustering percentage of LLE and LE for each set.

5.2 Texture Clustering

We also test our proposed algorithm in the segmentation of different textures. From the Columbia-Utrecht Reflectance and Texture Database (CUReT) found at `http://www1.cs.columbia.edu/CAVE//software/curet/` , we obtain samples of different textures and each grayscale image contains only one texture. In order to construct a histogram that reflects the texture statistics in an image, we will calculate what is commonly known as *textons* [20]. This is done by first applying a filter bank to all images in the training set. We use the Schmid [21] filter banks shown in Fig. 5. This will provide us with a feature vector $f(x, y)$ of dimension 13 at each pixel. Next, we apply k-means to all the vectors in the entire dataset to get 30 cluster centers, also known as the textons. For each image in the dataset, we then compute a histogram that contains the number of pixels corresponding to each one of these 30 bins. This is done by assigning a pixel (x, y) to bin i if the feature vector $f(x, y)$ is closest to cluster center $i = 1, \ldots, 30$, according to the Euclidean distance in $\mathbb{R}^{13}$.

Finally, we test our algorithm on a set of data containing 3 different textures. Fig. 7 shows a typical example of the different textures and the corresponding histograms in each set. The error produced by LLE in clustering is 5.43% whereas LE is significantly higher at 30.07%.

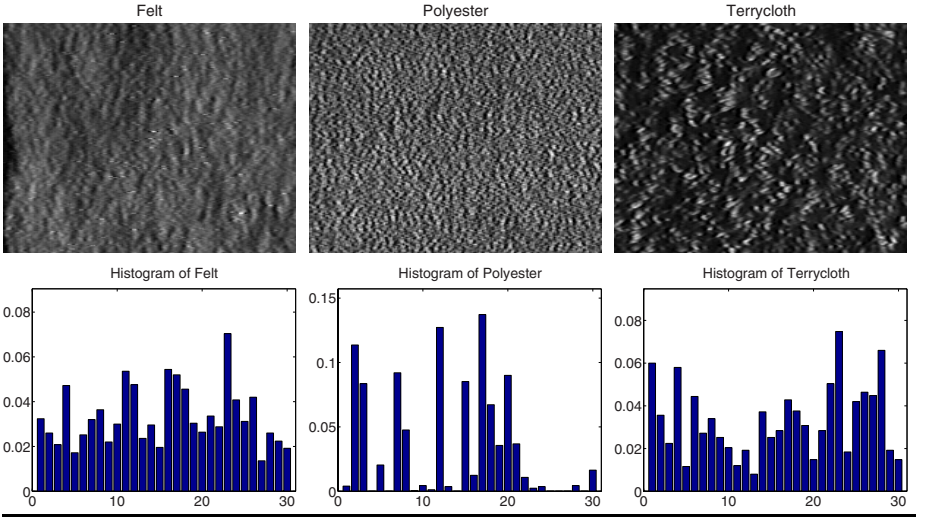


Fig. 7. Textures and corresponding histograms used in the three-class clustering experiment

6 Conclusion

We presented an algorithm to perform clustering of probability density functions. Our method takes into consideration the Riemannian structure of the square-root representation. Results on synthetic and real data are encouraging.

Acknowledgments. The authors thank Rizwan A. Chaudhry for useful discussions. This work has been supported by startup funds from JHU, by grants NSF CAREER IIS-0447739, NSF EHS-0509101, and ONR N00014-05-10836, and by contract JHU APL-934652.

References

1. Rao, C.R.: Information and accuracy attainable in the estimation of statistical parameters. Bull. Calcutta Math. Soc. 37, 81–89 (1945)
2. Amari, S.: Differential-Geometrical Methods in Statistics. Springer, Heidelberg (1985)
3. Mio, W., Badlyans, D., Liu, X.: A computational approach to Fisher information geometry with applications to image analysis. In: Rangarajan, A., Vemuri, B.C., Yuille, A.L. (eds.) EMMCVPR 2005. LNCS, vol. 3757, pp. 18–33. Springer, Heidelberg (2005)
4. Srivastava, A., Joshi, S., Mio, W., Liu, X.: Statistical shape analysis: clustering, learning, and testing. IEEE Transactions on PAMI 27(4), 590–602 (2005)
5. Srivastava, A., Jermyn, I., Joshi, S.: Riemannian analysis of probability density functions with applications in vision. In: IEEE CVPR (2007)
6. Hotelling, H.: Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24, 417–441 (1933)
7. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
8. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
9. Roweis, S., Saul, L.: Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4, 119–155 (2003)
10. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Neural Information Processing Systems, pp. 585–591 (2002)
11. Donoho, D., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. PNAS 100(10), 5591–5596 (2003)
12. Goh, A., Vidal, R.: Clustering and dimensionality reduction on Riemannian manifolds. In: IEEE CVPR (2008)
13. do Carmo, M.P.: Riemannian Geometry. Birkhäuser, Boston (1992)
14. Frechet, M.: Les elements aleatoires de nature quelconque dans un espace distancie. Annales De L'Institut Henri Poincare 10, 235–310 (1948)
15. Fletcher, P.T., Joshi, S.: Riemannian geometry for the statistical analysis of diffusion tensor data. Signal Processing 87(2) (2007)
16. Pennec, X., Fillard, P., Ayache, N.: A Riemannian framework for tensor computing. International Journal of Computer Vision 66(1), 41–46 (2006)
17. Karcher, H.: Riemannian center of mass and mollifier smoothing. Communications on Pure and Applied Mathematics 30(5), 509–541 (1977)
18. Horn, R., Johnson, C.: Matrix Analysis. Cambridge University Press, Cambridge (1985)
19. Cencov, N.N.: Statistical decision rules and optimal inference. In: Translations of Mathematical Monographs, vol. 53. AMS (1982)
20. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. International Journal of Computer Vision 62(1-2), 61–81 (2005)
21. Schmid, C.: Constructing models for content-based image retrieval. In: IEEE Conference on Computer Vision and Pattern Recognition (2001)

Online Manifold Regularization: A New Learning Setting and Empirical Study

Andrew B. Goldberg[1], Ming Li[2], and Xiaojin Zhu[1]

[1] Department of Computer Sciences, University of Wisconsin-Madison
Madison, WI, USA
{goldberg,jerryzhu}@cs.wisc.edu
[2] National Key Laboratory for Novel Software Technology, Nanjing University
Nanjing, China
lim@lamda.nju.edu.cn

Abstract. We consider a novel "online semi-supervised learning" setting where (mostly unlabeled) data arrives sequentially in large volume, and it is impractical to store it all before learning. We propose an online manifold regularization algorithm. It differs from standard online learning in that it learns even when the input point is unlabeled. Our algorithm is based on convex programming in kernel space with stochastic gradient descent, and inherits the theoretical guarantees of standard online algorithms. However, naïve implementation of our algorithm does not scale well. This paper focuses on efficient, practical approximations; we discuss two sparse approximations using buffering and online random projection trees. Experiments show our algorithm achieves risk and generalization accuracy comparable to standard batch manifold regularization, while each step runs quickly. Our online semi-supervised learning setting is an interesting direction for further theoretical development, paving the way for semi-supervised learning to work on real-world lifelong learning tasks.

1 Introduction

Consider a robot with a video camera. The robot continuously takes high frame-rate video of its surroundings, and wants to learn the names of various objects in the video. However, like a child learning in the real world, the robot receives names from humans only very rarely. The robot is thus in a semi-supervised learning situation: most objects are unlabeled, while only a few are labeled by humans.

There are several challenges that distinguish this situation from standard semi-supervised learning. The robot cannot afford to store the massive amount of mostly unlabeled video before learning; it requires an "anytime classifier" that is ready to use at all times, yet is continuously improving; training must be cheap; and since the world is changing, it must adapt to non-stationarity in classification.

These challenges are well-studied in online learning. However, our situation is also different from standard online learning. Online learning (classification)

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 393–407, 2008.

traditionally assumes that every input point is fully labeled; it cannot take advantage of unlabeled data. But in the robot case, the vast majority of the input will be unlabeled. It seems wasteful to throw away the unlabeled input, as it may contain useful information.

We address this situation by combining semi-supervised learning with online learning. The resulting online semi-supervised learning algorithm is based on convex programming with stochastic gradient descent in kernel space. This combination is novel. To the best of our knowledge, the closest prior work is the multiview hidden Markov perceptron ([1], Section 4), which heuristically combines multiview learning with online perceptron. However, that work did not enjoy the theoretical guarantees afforded by the online learning literature, nor did it directly apply to other semi-supervised learning methods. In contrast, our method can lift any batch semi-supervised learning methods with convex regularized risks to the online setting. As a special case, we will discuss online manifold regularization in detail.

The focus of the present work is to introduce a novel learning setting, and to develop practical algorithms with experimental verification. It is important to consider the efficiency issues, as we do in Section 3, for the algorithm to be practically relevant. Our online semi-supervised learning algorithm inherits no-regret bounds from online convex programming but does not provide new bounds. It is our hope that the novel setting where most of the incoming data stream is unlabeled will inspire future work on improved bounds. Some of the future directions are laid out at the end of the paper.

2 Online Semi-supervised Learning

We build online semi-supervised learning with two main ingredients: online convex programming [2] and regularized risk minimization for semi-supervised learning (see the overview in [3,4]). Although kernel-based online convex programming is well-understood [5], we are not aware of prior application in the semi-supervised learning setting.

Consider an input sequence $x_1 \dots x_T$, where $x_t \in \mathbb{R}^d$ is the feature vector of the t-th data point. *Most (possibly even the vast majority) of the points are unlabeled.* Only occasionally is a point x_t accompanied by its label $y_t \in \mathcal{Y}$. This setting differs dramatically from traditional online learning where all points are labeled. Let K be a kernel over x and $\mathcal{H}_K$ the corresponding reproducing kernel Hilbert space (RKHS) [6]. Our goal is to learn a good predictor $f \in \mathcal{H}_K$ from the sequence. Importantly, learning proceeds in an iterative fashion:

1. At time t an adversary picks x_t and y_t, not necessarily from any distribution $P(x, y)$ (although we will later assume *iid* for predicting future data). The adversary presents x_t to the learner.
2. The learner makes prediction $f_t(x_t)$ using its current predictor f_t.
3. With a small probability p_l, the adversary reveals the label y_t. Otherwise, the adversary abstains, and x_t remains unlabeled.

4. The learner updates its predictor to f_{t+1} based on x_t and the adversary's feedback y_t, if any.

We hope the functions $f_1 \dots f_T$ "do well" on the sequence, and on future input if the data is indeed *iid*. The exact performance criteria is defined below.

2.1 Batch Semi-supervised Risks

Before introducing our online learning algorithm, we first review *batch* semi-supervised learning, where the learner has access to the labeled and unlabeled data all at once. A unifying framework for batch semi-supervised learning is risk minimization with specialized "semi-supervised" regularizers. That is, one seeks the solution $f^* = \text{argmin}_{f \in \mathcal{H}_K} J(f)$, where the *batch semi-supervised regularized risk* is

$$J(f) = \frac{1}{l} \sum_{t=1}^{T} \delta(y_t) c(f(x_t), y_t) + \frac{\lambda_1}{2} \|f\|_K^2 + \lambda_2 \Omega(f),$$

where l is the number of labeled points, $\delta(y_t)$ is an indicator function equal to 1 if y_t is present (labeled) and 0 otherwise, c is a convex loss function, λ_1, λ_2 are regularizer weights, $\|f\|_K$ is the RKHS norm of f, and Ω is the semi-supervised regularizer which depends on f and $x_1 \dots x_T$. Specific choices of Ω lead to familiar semi-supervised learning methods:

i) *Manifold regularization* [7,8,9]:

$$\Omega = \frac{1}{2T} \sum_{s,t=1}^{T} (f(x_s) - f(x_t))^2 w_{st}.$$

The edge weights w_{st} define a graph over the T points, e.g., a fully connected graph with Gaussian weights $w_{st} = e^{-\|x_s - x_t\|^2 / 2\sigma^2}$. In this case, Ω is known as the energy of f on the graph. It encourages label smoothness over the graph: similar examples (large w) tend to have similar labels.

ii) *Multiview learning* [10,11,12] optimizes multiple functions $f_1 \dots f_M$ simultaneously. The semi-supervised regularizer

$$\Omega = \sum_{i,j=1}^{M} \sum_{t=1}^{T} (f_i(x_t) - f_j(x_t))^2$$

penalizes differences among the learners' predictions for the same point.

iii) *Semi-supervised support vector machines (S3VMs)* [13,14,15]:

$$\Omega = \frac{1}{T-l} \sum_{t=1}^{T} (1 - \delta(y_t)) \max(1 - |f(x_t)|, 0).$$

This is the average "hat loss" on unlabeled points. The hat loss is zero if $f(x)$ is outside $(-1, 1)$, and is the largest when $f(x) = 0$. It encourages the decision boundary $f(x) = 0$ to be far away from any unlabeled points (outside the margin), thus avoiding cutting through dense unlabeled data regions.

2.2 From Batch to Online

A key observation is that for certain semi-supervised learning methods, the batch risk $J(f)$ is the sum of *convex functions* in f. These methods include manifold regularization and multiview learning, but not S3VMs whose hat loss is non-convex. For these convex semi-supervised learning methods, one can derive a corresponding online semi-supervised learning algorithm using online convex programming. The remainder of the paper will focus on manifold regularization, with the understanding that online versions of multiview learning and other convex semi-supervised learning methods can be derived similarly.

We follow the general approach in [2,5]. Recall the batch risk for our version of manifold regularization in Section 2.1 is

$$J(f) = \frac{1}{l}\sum_{t=1}^{T}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \frac{\lambda_2}{2T}\sum_{s,t=1}^{T}(f(x_s) - f(x_t))^2 w_{st}, \quad (1)$$

and f^* is the batch solution that minimizes $J(f)$. In online learning, the learner only has access to the input sequence up to the current time. We thus define the *instantaneous regularized risk* $J_t(f)$ at time t to be

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2\sum_{i=1}^{t-1}(f(x_i) - f(x_t))^2 w_{it}. \quad (2)$$

The last term in $J_t(f)$ involves the graph edges from x_t to all previous points up to time t. The astute reader might notice that this poses a computational challenge—we will return to this issue in Section 3. While T appears in (2), $J_t(f)$ depends only on the *ratio* T/l. This is the empirical estimate of the inverse label probability $1/p_l$, which we assume is given and easily determined based on the rate at which humans can label the data at hand.

All the J_t's are convex. They are intimately connected to the batch risk J:

Proposition 1. $J(f) = \frac{1}{T}\sum_{t=1}^{T} J_t(f)$.

Our online algorithm constructs a sequence of functions $f_1 \ldots f_T$. Let $f_1 = 0$. The online algorithm simply performs a gradient descent step that aims to reduce the instantaneous risk in each iteration:

$$f_{t+1} = f_t - \eta_t \left.\frac{\partial J_t(f)}{\partial f}\right|_{f_t}. \quad (3)$$

The step size η_t needs to decay at a certain rate, e.g., $\eta_t = 1/\sqrt{t}$. Under mild conditions, this seemingly naïve online algorithm has a remarkable guarantee that on any input sequence, there is asymptotically "no regret" compared to the batch solution f^*. Specifically, let the *average instantaneous risk* incurred by the online algorithm be $J_{air}(T) \equiv \frac{1}{T}\sum_{t=1}^{T} J_t(f_t)$. Note J_{air} involves a varying sequence of functions $f_1 \ldots f_T$. As a standard quality measure in online learning, we compare J_{air} to the risk of the best *fixed* function in hindsight:

$$J_{air}(T) - \min_f \frac{1}{T}\sum_{t=1}^{T} J_t(f)$$
$$= J_{air}(T) - \min_f J(f) = J_{air}(T) - J(f^*),$$

where we used Proposition 1. This difference is known as the average regret. Applying Theorem 1 in [2] results in the no-regret guarantee $\limsup_{T\to\infty} J_{air}(T) - J(f^*) \leq 0$. It is in this sense that the online algorithm performs as well as the batch algorithm on the sequence.

To compute (3) for manifold regularization, we first express the functions $f_1 \dots f_T$ using a common set of representers $x_1 \dots x_T$ [16]

$$f_t = \sum_{i=1}^{t-1} \alpha_i^{(t)} K(x_i, \cdot). \tag{4}$$

The problem of finding f_{t+1} becomes computing the coefficients $\alpha_1^{(t+1)},\dots,\alpha_t^{(t+1)}$. Again, this will be a computational issue when T is large, and will be addressed in Section 3. We extend the kernel online supervised learning approach in [5] to semi-supervised learning by writing the gradient $\partial J_t(f)/\partial f$ in (3) as

$$\frac{T}{l}\delta(y_t)c'(f(x_t), y_t)K(x_t, \cdot) + \lambda_1 f \tag{5}$$
$$+2\lambda_2 \sum_{i=1}^{t-1}(f(x_i) - f(x_t))w_{it}(K(x_i, \cdot) - K(x_t, \cdot)),$$

where we used the reproducing property of RKHS in computing the derivative: $\partial f(x)/\partial f = \partial\langle f, K(x, \cdot)\rangle/\partial f = K(x, \cdot)$. c' is the (sub)gradient of the loss function c. For example, when $c(f(x), y)$ is the hinge loss $\max(1 - f(x)y, 0)$, we may define $c'(f(x), y) = -y$ if $f(x)y \leq 1$, and 0 otherwise. Putting (5) back in (3), and replacing f_t with its kernel expansion (4), it can be shown that f_{t+1} has the following coefficients:

$$\alpha_i^{(t+1)} = (1 - \eta_t\lambda_1)\alpha_i^{(t)} - 2\eta_t\lambda_2(f_t(x_i) - f_t(x_t))w_{it}, \quad i = 1 \dots t-1$$
$$\alpha_t^{(t+1)} = 2\eta_t\lambda_2 \sum_{i=1}^{t-1}(f_t(x_i) - f_t(x_t))w_{it} - \eta_t\frac{T}{l}\delta(y_t)c'(f(x_t), y_t). \tag{6}$$

We now have a basic online manifold regularization algorithm; see Algorithm 1.

When the data is *iid*, the generalization risk of the average function $\bar{f} = 1/T\sum_{t=1}^{T} f_t$ approaches that of f^* [17]. The average function $\bar{f}$ involves all representers $x_1, \dots, x_T$. For basic online manifold regularization, it is possible to incrementally maintain the exact $\bar{f}$ as time increases. However, for the sparse approximations introduced below, the basis changes over time. Therefore, in those cases $\bar{f}$ can be maintained only approximately using matching pursuit [18]. In our experiments, we compare the classification accuracy of $\bar{f}$ vs. f^* on a separate test set, which is of practical interest.

Algorithm 1. Online Manifold Regularization

Parameters: edge weight function w, kernel K, weights λ_1, λ_2, loss function c, label ratio T/l, step sizes η_t
Initialize $t = 1, f_1 = 0$
loop
  receive x_t, predict $f_t(x_t)$ using (4)
  (occasionally) receive y_t
  update f_t to f_{t+1} using (6)
  store x_t, let $t = t + 1$
end loop

3 Sparse Approximations

Unfortunately, Algorithm 1 will not work in practice because it needs to store every input point and soon runs out of memory; it also has time complexity $O(T^2)$. In particular, the instantaneous risk (2) and the kernel representation (4) both involve the sequence up to the current time. To be useful, it is imperative to sparsify both terms. In this section, we present two distinct approaches for this purpose: i) using a small buffer of points, and ii) constructing a random projection tree that represents the manifold structure.

3.1 Buffering

Buffering (e.g., [19] and the references therein) keeps a limited number of points. Let the buffer size be τ. The simplest buffering strategy replaces the oldest point $x_{t-\tau}$ in the buffer with the incoming point x_t. With buffering, the approximate instantaneous risk is

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2\frac{t}{\tau}\sum_{i=t-\tau}^{t-1}(f(x_i) - f(x_t))^2 w_{it}, \quad (7)$$

where the scaling factor t/τ keeps the magnitude of the graph regularizer comparable to the unbuffered version. In terms of manifold regularization, buffering corresponds to a dynamic graph on the points in the buffer. Similarly, the kernel expansion now has τ terms:

$$f_t = \sum_{i=t-\tau}^{t-1} \alpha_i^{(t)} K(x_i, \cdot).$$

With buffering, the function update involves two steps. In the first step, we update f_t to an intermediate function f' represented by a basis of $\tau+1$ elements, consisting of the old buffer and the new point x_t:

$$f' = \sum_{i=t-\tau}^{t-1} \alpha_i' K(x_i, \cdot) + \alpha_t' K(x_t, \cdot)$$

$$\alpha_i' = (1-\eta_t\lambda_1)\alpha_i^{(t)} - 2\eta_t\lambda_2(f_t(x_i) - f_t(x_t))w_{it}, \quad i = t-\tau \ldots t-1$$
$$\alpha_t' = 2\eta_t\lambda_2\frac{t}{\tau}\sum_{i=t-\tau}^{t-1}(f_t(x_i) - f_t(x_t))w_{it} - \eta_t\frac{T}{l}\delta(y_t)c'(f(x_t), y_t). \quad (8)$$

Second, we evict $x_{t-\tau}$ from the buffer, add x_t to the buffer, and approximate f' (which uses $\tau+1$ basis functions) with f_{t+1} (which uses τ basis functions):

$$\min_{\alpha^{(t+1)}} \|f' - f_{t+1}\|^2 \quad \text{s.t. } f_{t+1} = \sum_{i=t-\tau+1}^{t} \alpha_i^{(t+1)} K(x_i, \cdot). \quad (9)$$

Intuitively, we "spread" $\alpha'_{t-\tau}K(x_{t-\tau}, \cdot)$ to the remaining points in the buffer, in an attempt to minimize the change caused by truncation. We use kernel matching pursuit [18] to efficiently find the approximate coefficients $\alpha^{(t+1)}$ in (9). Matching pursuit is a greedy function approximation scheme. It iteratively selects a basis function on which to spread the residual in $\alpha'_{t-\tau}K(x_{t-\tau}, \cdot)$. The number of steps (i.e., basis functions selected) can be controlled to trade-off approximation error and speed. We run matching pursuit until the norm of the residue vector has been sufficiently reduced. We call the above buffering strategy "**buffer**." The overall time complexity for buffering is $O(T)$.

An alternative buffering strategy, "**buffer-U**," evicts the oldest *unlabeled* points in the buffer while keeping labeled points. This is motivated by the fact that the labeled points tend to have larger α coefficients and exert more influence on our learned function. The oldest labeled point is evicted from the buffer only when it is filled with labeled points. Note this is distinct from batch learning: the labeled points only form a better basis, but learning is still done via gradient descent.

3.2 Random Projection Tree

Another way to improve Algorithm 1 is to construct a sparse representation of the manifold. While many embedding techniques exist, we require one that is fast and can be incrementally modified. Recently random projection has been proposed as an efficient means to preserve the manifold structure (see e.g., [20,21]). We build our algorithm upon the online version of the Random Projection Tree (RPtree [22], Appendix I). An RPtree is a tree data structure with desirable theoretical properties that asymptotically traces the manifold. The basic idea is simple: as points arrive sequentially, they are spatially sorted into the RPtree leaves. When enough points fall into a leaf, the RPtree grows by splitting the leaf along a hyperplane with random orientation. An RPtree can be regarded as an efficient online clustering algorithm whose clusters grow over time and cover the manifold, as shown in Figure 1. We refer the reader to [22] for details, while presenting our extensions for semi-supervised learning below.

Let $\{L_i\}_{i=1}^s, s \ll t$ denote the leaves in the RPtree at time t. To model the data points that have fallen into each leaf, we maintain a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ at each leaf L_i, where μ_i and Σ_i are estimated *incrementally* as the

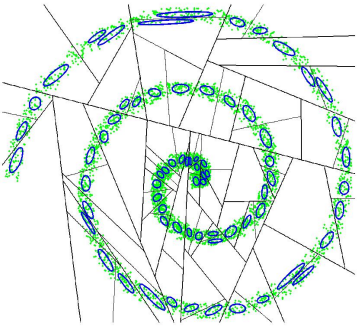


Fig. 1. A random projection tree on the Swiss roll data. Small dots represent data points, line segments represent the random splits in the internal nodes of the RPtree, polygons represent the regions governed by the leaves, and ellipses represent the Gaussian distributions on the data points within each leaf. We exploit the fact that these distributions follow the manifold structure of the data.

data points arrive. We also keep track of n_i, the number of points in leaf L_i. With an RPtree, we approximate the kernel representation of f_t (4) by the means of the Gaussian distributions associated with the tree leaves: $f_t = \sum_{i=1}^{s} \beta_i^{(t)} K(\mu_i, \cdot)$. We approximate the instantaneous risk (2) by

$$J_t(f) = \frac{T}{l}\delta(y_t)c(f(x_t), y_t) + \frac{\lambda_1}{2}\|f\|_K^2 + \lambda_2 \sum_{i=1}^{s} n_i (f(\mu_i) - f(x_t))^2 w_{\mu_i t}. \quad (10)$$

From a graph regularization point of view, this can be understood as having a coarser graph over the RPtree leaves. We define the edge weight $w_{\mu_i t}$ between incoming point x_t and each leaf L_i to be

$$\begin{aligned} w_{\mu_i t} &= \mathbb{E}_{x \sim \mathcal{N}(\mu_i, \Sigma_i)} \left[\exp\left(-\frac{||x - x_t||^2}{2\sigma^2} \right) \right] \quad (11) \\ &= (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} |\Sigma_0|^{-\frac{1}{2}} |\tilde{\Sigma}_i|^{\frac{1}{2}} \\ &\quad \exp\left(-\frac{1}{2} \left(\mu_i^\top \Sigma_i^{-1} \mu_i + x_t^\top \Sigma_0^{-1} x_t - \tilde{\mu}_i^\top \tilde{\Sigma}_i \tilde{\mu}_i \right) \right), \end{aligned}$$

where $\Sigma_0 = \sigma^2 I$, $\tilde{\Sigma}_i = (\Sigma_i^{-1} + \Sigma_0^{-1})^{-1}$, $\tilde{\mu}_i = \Sigma_i^{-1}\mu_i + \Sigma_0^{-1} x_t$, and σ is the bandwidth of the (original point to point) weight. We call this weight scheme "**RPtree PPK**" for its similarity to the probability product kernel [23]. An even simpler approximation is to ignore the covariance structure by defining $w_{\mu_i t} = e^{-\|\mu_i - x_t\|^2 / 2\sigma^2}$. It has computational advantages at the price of precision. We call this weight scheme "**RPtree**."

With an RPtree, the function update occurs in three steps. As space precludes a detailed discussion, we present an outline here. In the first step, upon receiving x_t, we update f_t to an intermediate function f' using a basis of $s + 1$ elements:

$\mu_1, \ldots, \mu_s$ and x_t. This is similar to (8) in the buffering case. In the second step, the RPtree itself is adjusted to account for the addition of x_t. The adjustments include updating the Gaussian parameters for the leaf x_t falls into, and potentially splitting the leaf. In the latter case, the number of leaves s will increase to s', and each new leaf's mean and covariance statistics are established. In the third step, we approximate f' by f_{t+1} using the s' new basis elements $\mu_1, \ldots, \mu_{s'}$ ($s' = s$ if no split happened), similar to (9). The point x_t is then discarded.

4 Experiments

We present a series of experimental results as empirical evidence that online manifold regularization (MR) is a viable option for performing fast MR on large data sets. We summarize our findings as follows:

1. Online MR scales better than batch MR in time and space. Although recent advances in manifold regularization greatly improve the feasible problem size (e.g., [24]), we believe that it takes online learning to handle unlimited input sequences and achieve life-long learning.
2. Online MR achieves comparable performance to batch MR. This is measured by two criteria:
   (a) $J_{air}(T)$ approaches $J(f^*)$, both for the basic online MR algorithm, as well as for the buffering and RPtree approximations.
   (b) Generalization error of $\bar{f}$ approaches that of f^* on test sets.
3. Online MR can handle concept drift (changes in $P(x)$ and $P(y|x)$). The online method (using a limited size buffer) can track a non-stationary distribution and maintain good generalization accuracy, while the batch method trained on all previous data fails to do so.

Our focus is on comparing online MR to batch MR, not semi-supervised learning to supervised learning. It is known that semi-supervised learning does not necessarily outperform supervised learning, depending on the correctness of model assumptions. Thus, our experiments use tasks where batch MR has proven beneficial in prior work, and we demonstrate that online MR provides a useful alternative to batch MR on these tasks.

4.1 Data Sets and Protocol

We report results on three data sets. The first is a toy two-spirals data set. The training sequences and test sets (of size 2000) are generated *iid*. The second is the MNIST digit classification data set [25], and we focus on two binary tasks: 0 vs. 1 and 1 vs. 2. We scaled down the images to 16 x 16 pixels (256 features). The training sequences are randomly shuffled subsets of the official training sets, and we use the official test sets (of size 2115 for 0 vs. 1, and 2167 for 1 vs. 2). The third is the 361-dimensional Extended MIT face vs. non-face image classification data set ("Face") [26]. We sampled a balanced subset of the data, and split this into a training set and a test set. The same test set of size 2000 is used in all

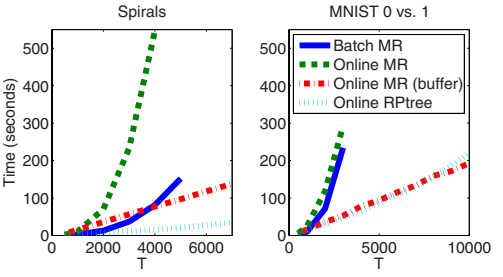


Fig. 2. Runtime growth curves. Batch MR and basic online MR scale quadratically, while the sparse approximations of buffering and RPtree scale only linearly.

experiments, while different training runs use different randomly shuffled subsets of the training set. The labeled rate p_l is 0.02 in all experiments, with points assigned to each class with equal probability.

Our experimental protocol is the following:

1. Generate randomly ordered training sequences and test sets (for MNIST and Face, the test sets are already given).
2. For batch MR, train separate versions on increasing subsequences (i.e., $T = 500, 1000, 2000, \ldots$).
3. For online MR, train once on the entire sequence.
4. For each T, compare the corresponding batch MR f^* with the online classifier trained up to T.

All results are the average of five such trials. The error bars are ± 1 standard deviation.

All experiments use hinge loss c and RBF kernel K. The kernel bandwidth parameter σ_K, λ_1, λ_2, and the edge weight parameter σ were all tuned for batch MR using $T = 500$. When using a limited size buffer, we set $\tau = 300$, and only require that matching pursuit reduce the residue norm by 50%. We use a step size of $\eta_t = \gamma/\sqrt{t}$, where $\gamma = 0.03$ for the RPtree approximation, and 0.1 for all other methods. We implemented all methods using MATLAB and CPLEX.

4.2 Online MR Scales Better than Batch MR

We illustrate this point by comparing runtime growth curves on the spirals and MNIST 0 vs. 1 data sets. Figure 2(left) shows that, for the spirals data set, the growth rates of batch MR and basic online MR are quadratic as expected (in fact, online MR has more overhead in our MATLAB implementation). Batch MR runs out of memory after $T = 5000$, and we stop basic online MR at $T = 4000$ because the runtime becomes excessive. On the other hand, online MR (buffered) and online RPtree are linear. Though not included in the plot, online RPtree PPK has a curve nearly identical to online MR (buffered). Figure 2(right) demonstrates similar trends for the higher dimensional MNIST 0 vs. 1 data set.

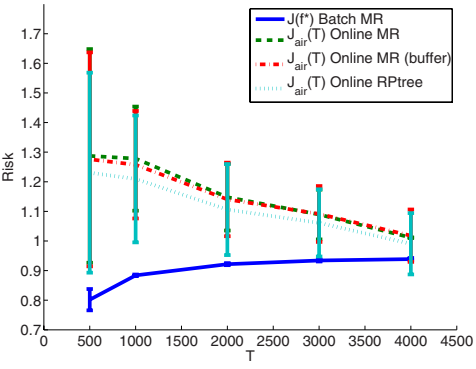


Fig. 3. Online MR's average instantaneous risk $J_{air}(T)$ approaches batch MR's risk $J(f^*)$ as T increases

4.3 Online MR Achieves Comparable Risks

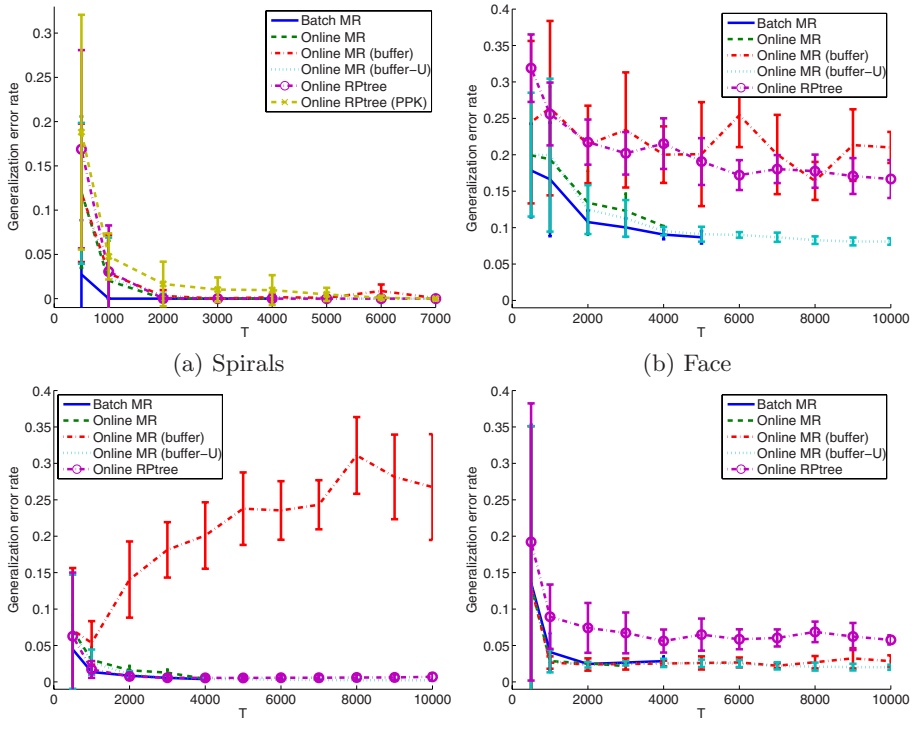
We compare online MR's average instantaneous risk $J_{air}(T)$ vs. batch MR's risk $J(f^*)$ on the training sequence. Our experiments support the theory that $J_{air}(T)$ converges to $J(f^*)$ as T increases. [1] Figure 3 compares these measures for basic online MR and batch MR on the spirals data set. The two curves approach each other. $J_{air}(T)$ continues to decrease beyond $T = 4000$ (not pictured). Figure 3 also shows that online MR (buffer) and online RPtree are good approximations to basic online MR in terms of J_{air}.

4.4 Generalization Error of Online MR

The experiments in this section compare the averaged function $\bar{f}$ of online MR and the batch solution f^* in terms of generalization error on test sets. Figure 4 presents results for all the data sets. We observe that online MR buffer-U is the best and consistently achieves test accuracy that is comparable to batch MR.

From Figure 4(a), we observe that, for the spirals data set, all the online methods perform nearly as well as batch MR. As is to be expected, batch MR makes the most efficient use of the data and reaches 0 test error first, while the online methods require only a little additional data to reach this level (after all, standard incremental learning usually needs multiple passes over the training set). Buffering and RPtree perform as well as basic online MR, showing little sign of approximation error. Panels (b), (c), and (d) in Figure 4 show that buffer-U can be much better than buffer. This is understandable, since matching pursuit may

[1] While the average regret approaches zero asymptotically, the step size of $\eta_t = 1/\sqrt{t}$ decays rapidly, potentially leading to slow convergence. Thus, it is possible that long sequences (i.e., large T values) would be required for the online algorithm to compete with the best batch algorithm. Nevertheless, our experiments show this is not actually a problem in practice.



(a) Spirals (b) Face

(c) MNIST 0 vs. 1 (d) MNIST 1 vs. 2

Fig. 4. Generalization error of batch MR's f^* and online MR's $\bar{f}$ as T increases. Online MR buffer-U consistently achieves test accuracy comparable to batch MR.

provide a poor approximation to the contributions of the discarded data point. In high dimensional space, there may be few similar data points remaining in the small buffer, so much of the weight assigned to discarded points is lost. Under the buffer-U strategy, we alleviate this issue by preserving the larger weights on labeled points, which approximate the function better. RPtree PPK on these high dimensional data sets involves expensive inversion of (often singular) covariance matrices and is not included in the comparison. The performance of RPtree is no better than buffer-U.

4.5 Online MR Handles Concept Drift

Lastly, we demonstrate that online MR can handle concept drift. When the underlying distributions, both $P(x)$ and $P(y|x)$, change during the course of learning, using buffered online MR is extremely advantageous. For this experiment, we "spin" the two spirals data set so that the spirals smoothly rotate 360° in every 4000 points (Figure 5). All points in the space will thus change their true labels during the sequence. We still provide only 2% of the labels to the

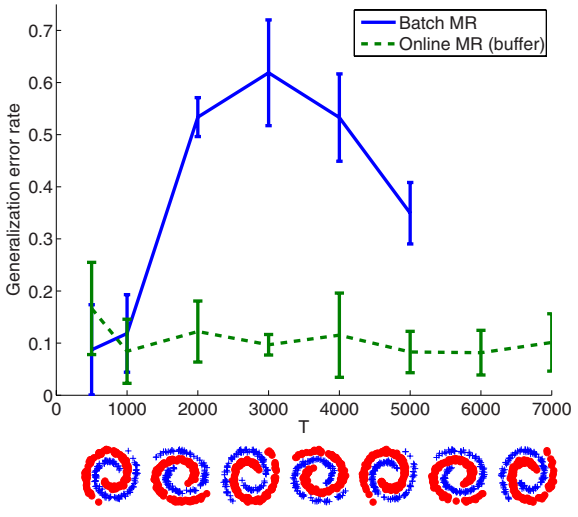


Fig. 5. Online MR (buffer) has much better generalization error than batch MR when faced with concept drift in the rotating spirals data set

algorithms. The test set for a given T consists of 2000 points drawn from the current underlying distribution.

For this experiment, we show the generalization error of batch MR's f^* vs. online MR (buffer)'s f_T, since the latest function is expected to track the changes in the data. Figure 5 illustrates that online MR (buffer) is able to adapt to the changing sequence and maintain a small error rate. In contrast, batch MR uses all data points, which now tend to conflict heavily (i.e., newer data from one class overlaps with older data from the other class). As expected, the single batch classifier f^* is inadequate for predicting such changing data.

5 Conclusions

We presented an online semi-supervised learning algorithm that parallels manifold regularization. Our algorithm is based on online convex programming in RKHS. We proposed two sparse approximations using buffering and online random projection trees to make online MR practical. The original batch manifold regularization algorithm has time complexity at least $O(T^2)$; so does the online version without sparse approximation. In contrast, the RPtree approximation has complexity $O(T \log T)$, where each iteration requires $O(\log T)$ leaf lookups (the tree's height is $O(\log T)$ because each leaf contains a constant maximum number of points). Buffering has complexity $O(T)$. Experiments show that our online MR algorithm has risk and generalization error comparable to batch MR, but scales much better. In particular, online MR (buffer-U) tends to have the best performance.

There are many interesting questions remaining in this online semi-supervised learning setting. Future work will proceed along two directions. On the empirical side, we will further speed up online MR, for example by using fast neighbor search to reduce the number of candidate basis elements in matching pursuit. We also plan to study practical online algorithms for other semi-supervised learning methods, in particular those with non-convex risks like S3VMs. On the theoretical side, we plan to investigate different regret notions that might be appropriate for this setting, performance guarantees with concept drift, and models that do not require all previous points.

Acknowledgements

A. Goldberg and X. Zhu were supported in part by the Wisconsin Alumni Research Foundation. This work was completed while M. Li was a visiting researcher at University of Wisconsin-Madison under a State Scholarship from the Chinese Scholarship Council. The authors also thank Shuchi Chawla for helpful discussions on online learning.

References

1. Brefeld, U., Büscher, C., Scheffer, T.: Multiview discriminative sequential learning. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 60–71. Springer, Heidelberg (2005)
2. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: ICML 2003 (2003)
3. Chapelle, O., Zien, A., Schölkopf, B. (eds.): Semi-supervised learning. MIT Press, Cambridge (2006)
4. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison (2005)
5. Kivinen, J., Smola, A.J., Williamson, R.C.: Online learning with kernels. IEEE Transactions on Signal Processing 52(8), 2165–2176 (2004)
6. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
7. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of Machine Learning Research 7, 2399–2434 (2006)
8. Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: ICML 2005 (2005)
9. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML 2003 (2003)
10. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: COLT 1998 (1998)
11. Sindhwani, V., Niyogi, P., Belkin, M.: A co-regularized approach to semi-supervised learning with multiple views. In: ICML 2005 (2005)
12. Brefeld, U., Gaertner, T., Scheffer, T., Wrobel, S.: Efficient co-regularized least squares regression. In: ICML 2006 (2006)
13. Joachims, T.: Transductive inference for text classification using support vector machines. In: ICML 1999 (1999)

14. Chapelle, O., Sindhwani, V., Keerthi, S.S.: Branch and bound for semi-supervised support vector machines. In: NIPS 2006 (2006)
15. Collobert, R., Sinz, F., Weston, J., Bottou, L.: Large scale transductive SVMs. The Journal of Machine Learning Research 7, 1687–1712 (2006)
16. Kimeldorf, G., Wahba, G.: Some results on Tchebychean spline functions. Journal of Mathematics Analysis and Applications 33, 82–95 (1971)
17. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Transactions on Information Theory 50(9), 2050–2057 (2004)
18. Vincent, P., Bengio, Y.: Kernel matching pursuit. Machine Learning 48(1-3), 165–187 (2002)
19. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: A kernel-based perceptron on a fixed budget. In: NIPS 2005 (2005)
20. Hegde, C., Wakin, M., Baraniuk, R.: Random projections for manifold learning. In: NIPS 2007 (2007)
21. Freund, Y., Dasgupta, S., Kabra, M., Verma, N.: Learning the structure of manifolds using random projections. In: NIPS 2007 (2007)
22. Dasgupta, S., Freund, Y.: Random projection trees and low dimensional manifolds. Technical Report CS2007-0890, University of California, San Diego (2007)
23. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. Journal of Machine Learning Research, Special Topic on Learning Theory 5, 819–844 (2004)
24. Tsang, I., Kwok, J.: Large-scale sparsified manifold regularization. In: NIPS 2006 (2006)
25. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, vol. 86(11), pp. 2278–2324 (1998)
26. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research 6, 363–392 (2005)

A Fast Algorithm to Find Overlapping Communities in Networks

Steve Gregory

Department of Computer Science
University of Bristol, BS8 1UB, England
steve@cs.bris.ac.uk

Abstract. Many networks possess a community structure, such that vertices form densely connected groups which are more sparsely linked to other groups. In some cases these groups overlap, with some vertices shared between two or more communities. Discovering communities in networks is a computationally challenging task, especially if they overlap. In previous work we proposed an algorithm, CONGA, that could detect overlapping communities using the new concept of *split betweenness*. Here we present an improved algorithm based on a *local* form of betweenness, which yields good results but is much faster. It is especially effective in discovering small-diameter communities in large networks, and has a time complexity of only O(n log n) for sparse networks.

1 Introduction and Related Work

In recent years, networks (graphs) have increasingly been used to represent various kinds of complex system in the real world. Many networks exhibit *community structure*: the tendency of vertices to form *communities* (or *modules*) such that intracommunity edges are denser than the edges between communities. Communities often reflect important relationships between individuals (vertices), so the automatic discovery of communities has become one of the key tasks in network analysis.

Even if we restrict our attention to unipartite networks with undirected, unweighted edges, as we do in this paper, there is already a wide choice of community detection algorithms. Many of these are described in the survey papers of [6, 14], and there are also many recent algorithms, including [3, 17, 21, 24, 27].

Unfortunately, there is no standard definition of *community* and no consensus about how a network should be divided into communities. The vast majority of existing algorithms partition a network into a *flat* set of *disjoint* sets (clusters) of vertices, though it is often possible, or necessary, to choose the number of clusters. However, in some networks the community structure is not flat: for example, a collaboration network may contain a community for each research area, each comprising a number of subcommunities corresponding to research groups. A few algorithms [5, 11] can detect such a hierarchical community structure. Moreover, in many networks, communities are not disjoint: for example, some researchers work on more than one topic and therefore belong simultaneously to multiple research groups. Some algorithms [2, 9, 20, 28] are able to detect these overlapping communities.

In this paper we focus on the detection of overlapping communities. In previous work we designed an algorithm, CONGA (Cluster-Overlap Newman Girvan Algorithm)

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 408–423, 2008.

[9], for this purpose. It extends Girvan and Newman's [8, 18] algorithm (the "GN algorithm") with the ability to split vertices between clusters, based on the new concept of *split betweenness*. CONGA yields good results but is extremely slow, with approximately cubic time complexity, so it can only cope with networks containing at most a few thousand vertices and edges. Many real-world networks are far larger than this, so CONGA cannot be used.

CONGA inherits its low speed from the GN algorithm. Both algorithms rely on *betweenness*, which is a *global* centrality measure: at each step, it counts the number of shortest paths between *all* pairs of vertices in the network. For a fast, scalable, algorithm we need a measure that can be computed *locally*. In this paper we show how CONGA can be made much faster using *local betweenness* [10, 23].

In the next section we outline CONGA, introduce the concept of local betweenness, and then describe our new algorithm: the CONGO (CONGA Optimized) algorithm. Section 3 presents the results of experiments with the new algorithm on both synthetic and real-world networks. We compare its performance and execution time with both CONGA and CFinder [20], another state-of-the-art algorithm for finding overlapping communities. Conclusions appear in Section 4.

2 The CONGO Algorithm

2.1 The CONGA Algorithm

The CONGA algorithm [9] comprises a sequence of steps, each of which removes an edge from the network *or* splits a vertex into two vertices:

1. Calculate edge betweenness of edges and split betweenness of vertices.
2. Remove edge with maximum edge betweenness or split vertex with maximum split betweenness, if greater.
3. Recalculate edge betweenness and split betweenness.
4. Repeat from step 2 until no edges remain.

Initially, the n-vertex network is treated as a single cluster, assuming it is connected. Eventually, step 2 causes the cluster to split into two components (clusters). Clusters continue to be split into two until only singleton clusters remain. The binary splits can be represented as a dendrogram, which is used to reconstruct a partition of the network into any desired number of clusters.

CONGA is the same as the GN algorithm [8, 18] except for the vertex splitting step, which allows overlapping clusters. Because of this, a vertex v may be split into i vertices (copies of v) distributed between j clusters ($1 \leq j \leq i$). When reconstructing the partition, these copies of v are replaced by v itself in each of these j clusters.

The edge betweenness of an edge e is the number of shortest paths, between all pairs of vertices, that pass along e. The split betweenness of a vertex v is the number of shortest paths that *would* pass between the two parts of v if it were split. Since there are many ($2^{d(v)-1}$-1, where $d(v)$ is the degree of v) ways to split v into two, we choose the *best split*: the one that maximizes the split betweenness.

In [9] we give an approximate algorithm for computing the split betweenness of a vertex from its pair betweennesses. The *pair betweenness* of v for $\{u,w\}$, where u and w are neighbours of v, is the number of shortest paths traversing both edges $\{u,v\}$ and $\{v,w\}$. It is straightforward to compute this while computing edge betweenness.

The GN algorithm has a worst-case time complexity of $O(m^2n)$, where m is the number of edges and n is the number of vertices. In CONGA, each vertex v can split into at most m/n vertices on average (i.e., $d(v)/2$), so the number of vertices after splitting is $O(m)$ instead of n. This makes the time complexity $O(m^3)$ in the worst case: there are $O(m)$ iterations, and both step 1 and step 3 are $O(m^2)$.

In practice, the speed depends heavily on the number of vertices that are split (which increases the network size) and on how easily the network breaks into separate components. This is because, in step 3, betweenness need be calculated only for the component containing the removed edge or split vertex, or for both components if step 2 caused the component to split.

2.2 Local Betweenness

Betweenness is expensive to compute because it counts *all* shortest paths in the network. One way to avoid this is to count only *short* shortest paths. We redefine the edge betweenness of edge e to be the number of shortest paths running along e whose length is less than or equal to h (a parameter of the algorithm). The *pair betweenness* of vertex v for $\{u,w\}$ is the number of shortest paths traversing $\{u,v\}$ and $\{v,w\}$ whose length is less than or equal to h. *Split betweenness* is derived from pair betweennesses in the same way as in CONGA.

Step 1 of the CONGA algorithm is performed by a breadth-first search from every vertex. Using local betweenness, the depth of this search (from each vertex) is limited to h, which is faster than traversing every edge in the network.

Local betweenness has an even greater effect on the speed of step 3: betweenness need not be recalculated for the whole network, but only locally: in a small subgraph around the edge that was removed, or the vertex that was split, in step 2. In Figs. 1 and 2 we illustrate how step 3 of CONGA can be optimized in this way, but first we need to define this small subgraph, which we call an h-region.

The *h-region* of edge $\{u,v\}$ — the region affected by the removal of $\{u,v\}$ — is the smallest subgraph containing all shortest paths no longer than h that pass along $\{u,v\}$. This is an induced subgraph with vertex set

$$V_{u,v,h} = \{w : d(u,w) < h \vee d(v,w) < h\} \tag{1}$$

where $d(u,w)$ denotes the shortest-path distance between u and v. The *h-region* of vertex v — the region affected by splitting v — is the smallest subgraph containing all shortest paths no longer than h that pass through v or start/end at v. This is an induced subgraph with vertex set

$$V_{v,h} = \{w : d(v,w) \leq h\} \tag{2}$$

For example, Fig. 1(a) shows a small network with the 2-region of edge {h,i} shown shaded. Fig. 2(a) shows another network, highlighting the 2-region of vertex u.

(a) Edge {h,i} selected for removal. 2-region of {h,i} is shaded.

(b) Shortest paths within region are found and subtracted from betweenness.

(c) {h,i} is removed. Shortest paths within region are found and added to betweenness.

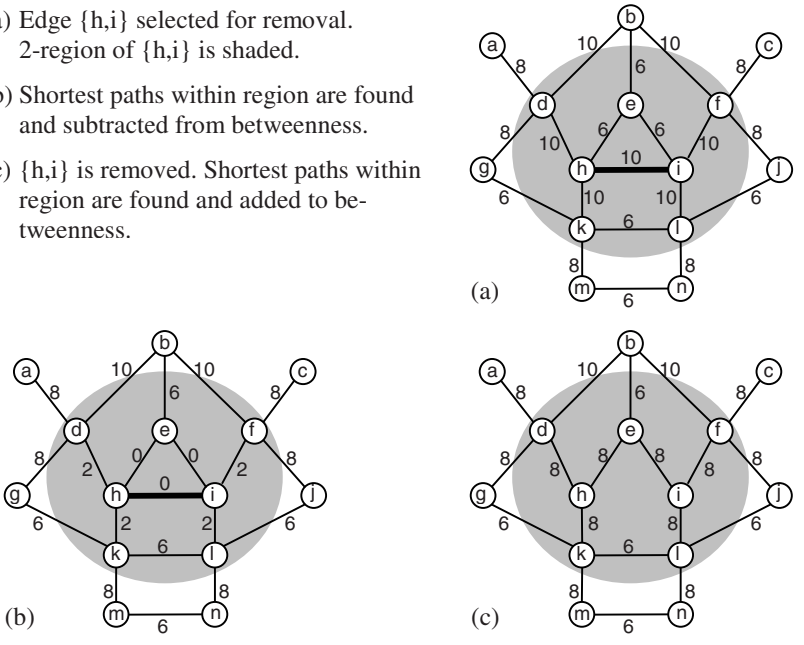


Fig. 1. Local recalculation of betweenness after removing an edge

(a) Vertex u selected for splitting. 2-region of u is shaded.

(b) Shortest paths within region are found and subtracted from betweenness.

(c) u is split. Shortest paths within region are found and added to betweenness.

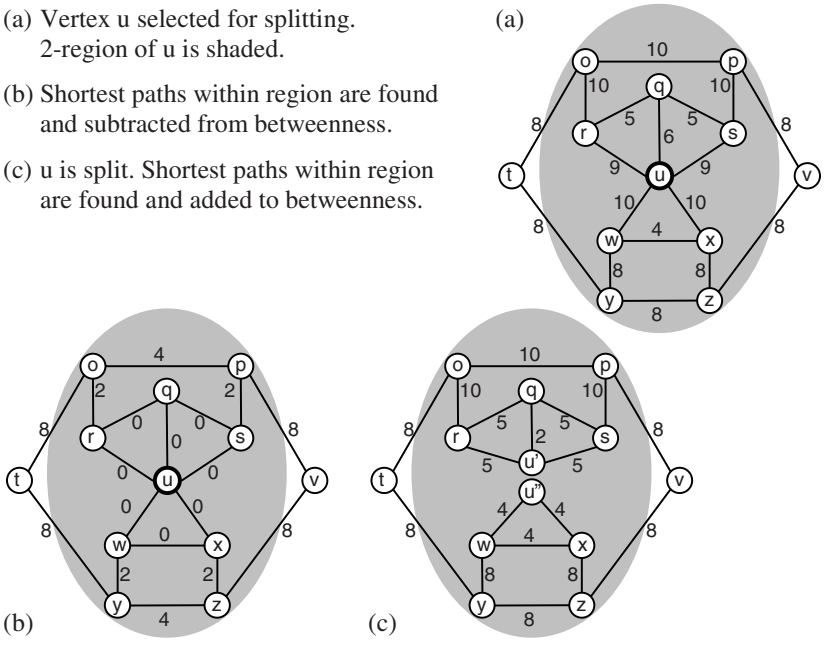


Fig. 2. Local recalculation of betweenness after splitting a vertex

In step 3, our new algorithm recalculates betweenness by a local method. It first "undoes" the betweenness of the *h*-region, by finding all shortest paths no longer than *h* that lie entirely within the region and subtracting their number from the (previously computed) edge betweenness of the edges they traverse and the pair betweennesses of the vertices they pass through. This has the effect of reducing the betweenness of the chosen edge (Fig. 1(b)) or vertex (Fig. 2(b)) to zero. After removing the edge or splitting the vertex, it again finds all shortest paths no longer than *h* within the region and adds their number to the edge betweenness of the edges they traverse and the pair betweennesses of the vertices they pass through; see Figs. 1(c) and 2(c).

2.3 The CONGO Algorithm

The CONGO algorithm is the same as CONGA (Section 2.1) but using local betweenness, explained in Section 2.2. The complete CONGO algorithm is as follows:

1. Calculate edge betweenness of edges and split betweenness of vertices.
2. Find edge with maximum edge betweenness or vertex with maximum split betweenness, if greater.
3. Recalculate edge betweenness and split betweenness:
   a) Subtract betweenness of *h*-region centred on the removed edge or split vertex.
   b) Remove the edge or split the vertex.
   c) Add betweenness for the same region.
4. Repeat from step 2 until no edges remain.

In practice, CONGO's execution time depends strongly on the structure of the network, but we can estimate its time complexity as follows.

For step 1, the time complexity for $h=\infty$ is $O(mn)$; this would reduce to $O(m)$ for $h=1$, which is of no practical use because the 1-betweenness of every edge is the same. For other small values of *h*, we make the simplifying assumption that all vertices have about the same degree, $2m/n$. Then, for each of the *n* vertices, the tree searched contains $O((m/n)^h)$ vertices. This makes the time complexity of step 1 approximately $O(m^h/n^{h-1})$, or $O(n)$ for a sparse network.

Making the same assumption for step 3, an *h*-region of an edge contains $O((m/n)^{h-1})$ vertices and $O((m/n)^h)$ edges; an *h*-region of a vertex contains $O((m/n)^h)$ vertices and $O((m/n)^{h+1})$ edges. Therefore, the time complexity of step 3 is approximately $O((m/n)^{2h+1})$, or $O(1)$ for a sparse network.

Step 2 takes $O(\log m)$ time, and the loop containing steps 2 and 3 is repeated $O(m)$ times. Therefore, the time complexity of the whole algorithm is $O(m \log m + m^{2h+2}/n^{2h+1})$, or $O(n \log n)$ for a sparse network.

3 Experiments

3.1 Experiments on Synthetic Networks

A common way to evaluate network clustering algorithm is by generating artificial networks based on a known community structure and comparing the known *communities*

with the *clusters* found by the algorithm. The comparison can be done in various ways, including the Mutual Information measure [7] and Rand Index [22]. We use the *F-measure*, defined as the harmonic mean of recall and precision, where:

- *recall*: the fraction of vertex pairs belonging to the same community that are also in the same cluster.
- *precision*: the fraction of vertex pairs in the same cluster that also belong to the same community.

(A pair of vertices are considered to "belong to the same community/cluster" if there exists *at least one* community/cluster that they *both* belong to. Because of overlap, there may be more than one of these.)

We randomly generated a set of networks containing n vertices divided into c equally-sized communities, each containing nr/c vertices. Vertices are randomly and evenly distributed between communities such that each vertex is a member of r communities on average. r is a measure of overlap, ranging from 1 (communities are disjoint) to c (communities each contain all vertices). The network is then constructed by placing edges between pairs of vertices randomly, with probability ip_{in} if there are i ($\geq$1) communities to which both vertices belong, and p_{out} otherwise. All networks used in the experiments are connected, and results shown are the average of 10 runs.

Below we compare our CONGO algorithm, for h=2 and h=3, with CONGA (which is equivalent to CONGO with h=∞). We also compare it with CFinder [1], based on the clique percolation algorithm of Palla *et al*. [20], one of the most efficient and best-known algorithms for finding overlapping communities. For CONGO and CONGA, the number of clusters is a parameter of the algorithm, so we set this to c, the known number of communities. This is impossible with CFinder, whose only parameter is k (cluster density). For fairness, we show the results from CFinder for *all* values of k.

Fig. 3 shows results for 256 vertices in 32 communities. The overlap is 2, meaning that each community contains 16 vertices. As p_{out} increases, the community structure becomes less evident and so CONGO's F-measure decreases, especially for h=2. In contrast, CFinder is relatively resilient to these intercommunity edges.

Fig. 4 shows the effect of increasing the density of intracommunity edges. The CONGO results are good and improve as p_{in} increases. Again, h=2 is worse than h=3, but only for small values of p_{in}. CFinder, for each k, peaks at a different value of p_{in}.

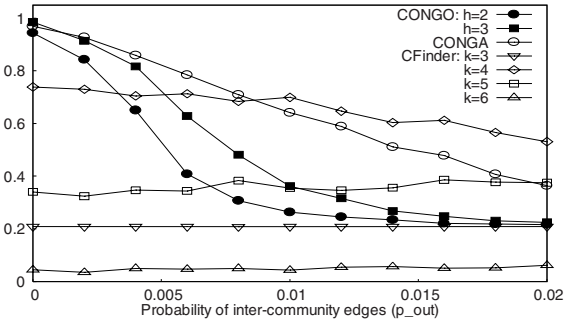


Fig. 3. F-measure for random networks with n=256, c=32, r=2, p_{in}=0.5, various p_{out}

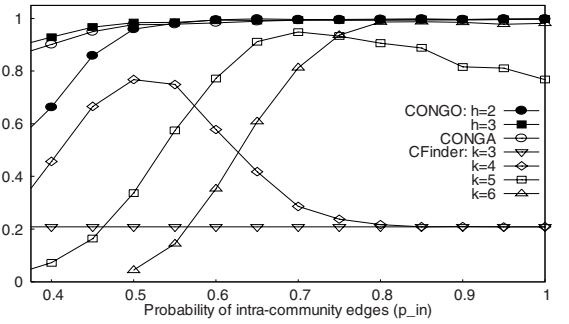


Fig. 4. F-measure for random networks with n=256, c=32, r=2, p_{out}=0, various p_{in}

In Fig. 5 we fix p_{in} and p_{out} and vary the overlap, r. CONGO's results decline as r gets larger, especially for h=2, while CFinder's results again peak at a different value of r for each k.

Fig. 6 shows the effect of varying the network size while keeping the community size constant. As expected, the F-measure results are quite stable, except for very small networks, because these contain very few communities.

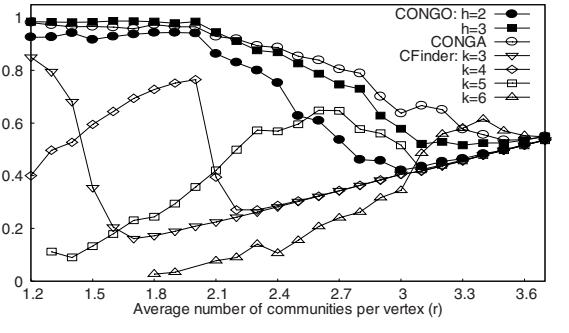


Fig. 5. F-measure for random networks with n=256, c=32, p_{in}=0.5, p_{out}=0, various r

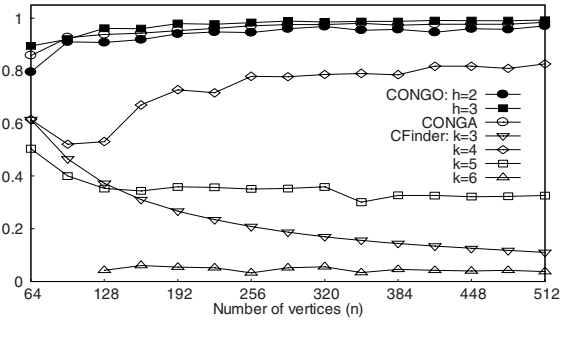


Fig. 6. F-measure for random networks with c=n/8, r=2, p_{in}=0.5, p_{out}=0, various n

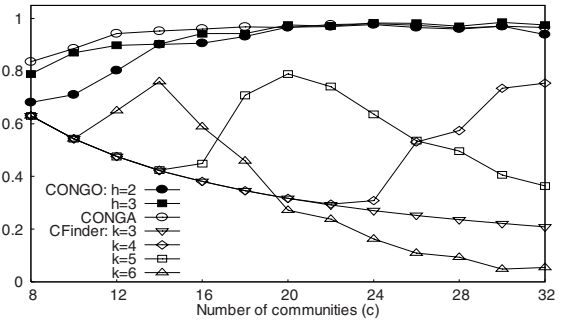


Fig. 7. F-measure for random networks with n=256, r=2, p_{in}=0.5, p_{out}=0, various c

In Fig. 7 we fix the size of the network but vary the number (and therefore size) of the communities. CONGO again performs well, but slightly less well for h=2.

In summary, on synthetic networks, CONGO's results are similar to those of CONGA: generally better than CFinder except where there are intercommunity edges. CONGO, like CONGA, treats an intercommunity edge as evidence that the two communities overlap. The difference between CONGO and CONGA is that a smaller value of h leads to slightly lower accuracy (F-measure).

Figs. 8 and 9 show the effect of local betweenness on execution time. For CONGO and CONGA, the plots show the time taken to compute the entire dendrogram and extract the clustering from it, using the author's implementation of the algorithms in Java, running on an AMD Opteron 250 CPU at 2.4GHz. For CFinder (v1.21), the times include the generation of solutions for all values of k, on the same machine.

Fig. 8 shows the time to cluster networks of varying size containing fixed-size (16) overlapping communities. The figure shows the approximately cubic time complexity of CONGA, which can only handle 2000 vertices in 20 minutes. CFinder is faster, taking only 15 minutes to cluster a 30000-vertex network. However, CONGO can cope with 500000 vertices in about 10 (h=2) or 20 (h=3) minutes. The CONGO results seem to confirm the O(n log n) time complexity that we predicted.

In Fig. 8 the community size is fixed and so the average degree is constant: about 5 (shown by the dashed line). This is not always the case in real networks. Fig. 9 shows how the algorithms scale in the extreme case: where there is an increasing number of vertices divided into a fixed number (12) of communities. Now, CONGO's execution time increases with the number of edges, but much more slowly than CFinder's.

3.2 Modularity

Evaluating an algorithm on real-world networks is challenging, because there is usually no known "correct" solution. The quality of clusterings must be assessed in a different way: for example, by modularity [17, 18], which measures the relative number of intracluster and intercluster edges. A high modularity indicates that there are more intracluster edges than would be expected by chance.

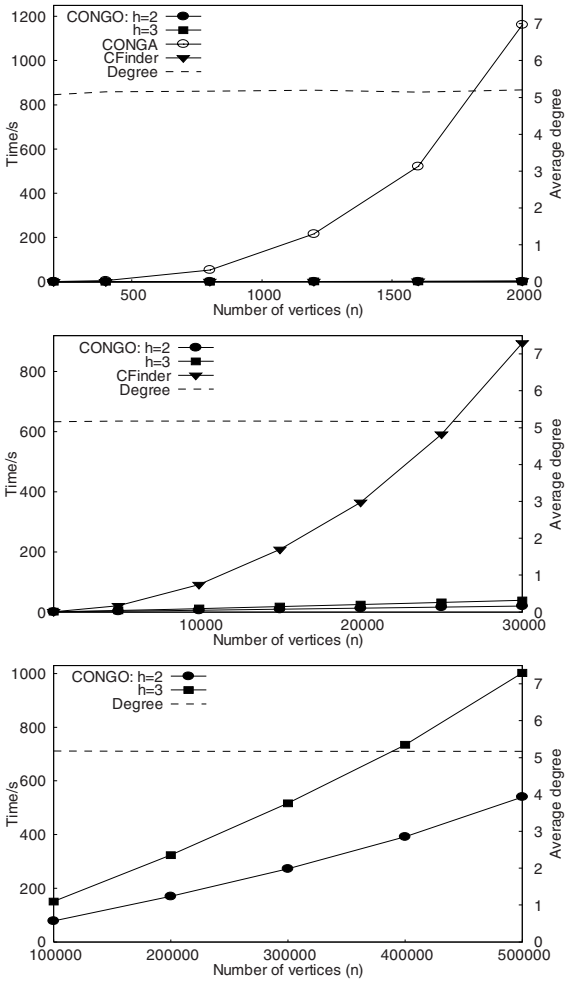


Fig. 8. Execution time for random networks $c=n/8$, $r=1.2$, $p_{in}=0.5$, $p_{out}=0$, various n

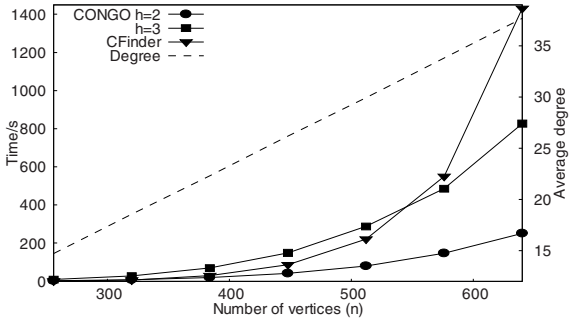


Fig. 9. Execution time for random networks $c=12$, $r=1.2$, $p_{in}=0.5$, $p_{out}=0$, various n

The original modularity measure, Q, is defined only for disjoint communities, but Nicosia *et al*. [19] have recently proposed a new modularity measure, Q_{ov}, which *is* defined also for overlapping communities. The definition of Q_{ov} is too long to reproduce here, but its main features are:

1. $Q_{ov} = 0$ when all vertices belong to the same community or all belong to singleton communities.
2. Higher values of Q_{ov} show stronger community structure.
3. Each vertex may belong to any number of communities with any *belonging coefficient*. For each vertex, the belonging coefficients for all communities sum to 1.

Although both old and new measures are named "modularity", they generally have different values even when applied to the same clustering and network. In the rest of this paper we use the term "modularity" to refer to Q_{ov}. We use it in this section to evaluate solutions on real-world networks. For our experiments we set the belonging coefficients of each vertex to $1/c$, where c is the number of communities it is in; i.e., equal membership of all communities.

It is sometimes assumed that the best clustering is the one that maximizes the value of modularity. The maximum value of modularity has even been used to compare clustering algorithms. However, as pointed out in [10], the peak value of modularity does *not* in general coincide with the correct, or best, clustering. This is true of both Q and Q_{ov} modularity measures.

To illustrate this, Fig. 10 shows the modularities of the solutions found by CONGO and CFinder for a synthetic network with 250 overlapping communities. For CONGO, modularity peaks at 0.822 between 45 and 53 clusters, where the F-measure is below 0.1. At 250 clusters, the correct solution, with F-measure 0.977 (h=3) or 0.891 (h=2), the modularity is only 0.701 (h=3) or 0.705 (h=2). CFinder finds a solution with 291 clusters with modularity 0.635 and F-measure 0.877.

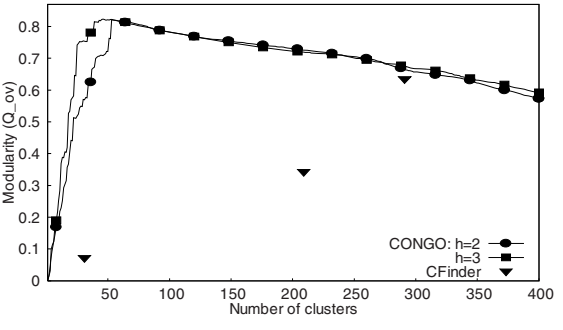


Fig. 10. Modularity of clusterings of random network: n=2000, c=250, r=1.2, p_{in}=0.5, p_{out}=0

Fig. 11 shows the results of a similar network with a larger overlap. CONGO's Q_{ov} at 250 clusters is 0.25 for both h=3 and h=2, and the F-measure is 0.998 (h=3) or 0.992 (h=2). Although Q_{ov} has a local maximum at 250 clusters, it is well below the global maximum between 63 and 79 clusters, where the F-measure is below 0.001. The closest CFinder solution is at 290 clusters, with Q_{ov} 0.142 and F-measure 0.842.

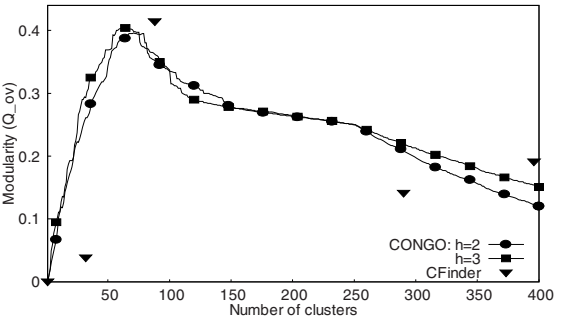


Fig. 11. Modularity of clusterings of random network: n=2000, c=250, r=2, p_{in}=0.5, p_{out}=0

We conclude that there is at best a tenuous relationship between modularity and correctness. Nevertheless, because modularity is widely used to assess clustering algorithms, we use it in the next section to evaluate the performance of CONGO (and CFinder) on some real-world networks. Because the peak value of modularity is meaningless, we plot the modularity of *all* solutions containing up to $n/5$ clusters.

3.3 Experiments on Real-World Networks

We have run the CONGO algorithm, and CFinder, on several real-world networks, listed in Table 1. The table shows the source of each network, its size, and the execution times for CONGO (to compute the entire dendrogram) and CFinder (v1.21) (to generate solutions for all values of k), running on an AMD Opteron 250 at 2.4GHz.

Table 1. Results on real-world networks

Name	Ref.	Fig.	Vertices	Edges	Runtime / s		
					CONGO		CF
					h=3	h=2	
netscience	[16]	12	379	914	1.4	1.3	0.3
cond-mat-2003	[13]	13	27519	116181	45110	1111	1140
blogs	[26]	14	3982	6803	33.5	6.1	3.2
blogs2	[26]	15	30557	82301	11702	286	405
PGP	[4]	16	10680	24316	636	82	35022
word_association	[12]	17	7205	31784	12026	172	97
protein-protein	[20]	18	2445	6265	94.5	8.2	2.9

"netscience" (Fig. 12) and "cond-mat-2003" (Fig. 13) are collaboration networks of coauthorships, of different sizes. The first of these is small enough for CONGA to handle, so its modularities are plotted along with those of CONGO. CONGA finds high-modularity solutions for small numbers of clusters, but is otherwise similar to CONGO for h=2 and h=3. For both networks, the modularities of the h=2 and h=3 solutions are

quite similar. CFinder finds several solutions, one of which, for a relatively large number of clusters, has a higher modularity than CONGO's.

"blogs" (Fig. 14) and "blogs2" (Fig. 15) are networks of communication relationships between owners of blogs on the MSN (Windows Live™) Spaces website. "blogs2" is much larger than "blogs" and has a higher average degree. "PGP" (Fig. 16) is yet another type of social network, representing PGP key signing. For all three networks, the modularities are quite similar to those of Figs. 12 and 13.

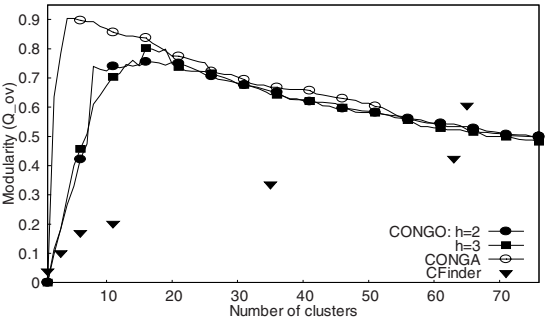


Fig. 12. Modularity of netscience network

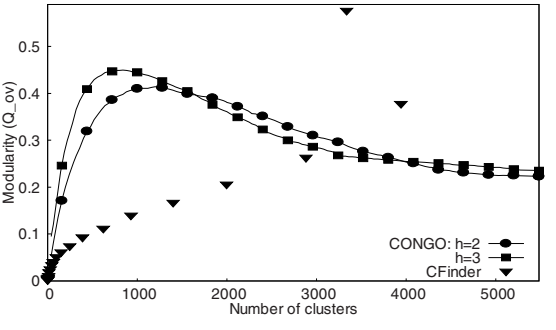


Fig. 13. Modularity of cond-mat-2003 network

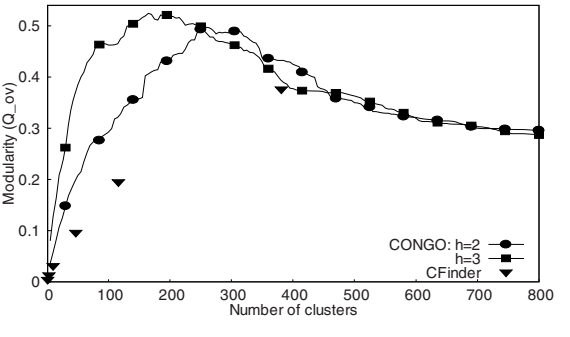


Fig. 14. Modularity of blogs network

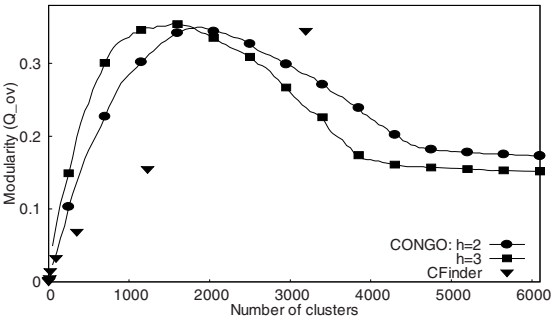


Fig. 15. Modularity of blogs2 network

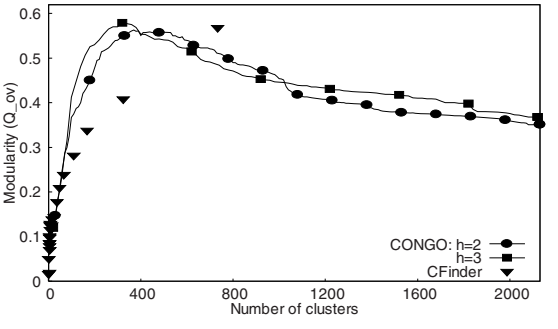


Fig. 16. Modularity of PGP network

Figs. 17 and 18 show two non-social networks, from psychology and biology, respectively. "word_association" is a word association network from [20], converted from an original directed, weighted version [12]. "protein-protein" is a yeast core protein-protein interaction graph provided by [20]. For the first of these, CFinder finds a higher modularity solution than CONGO for a small number of clusters.

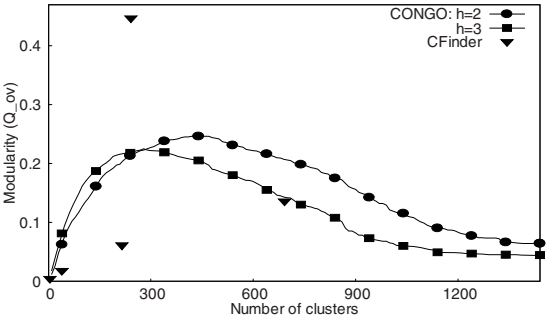


Fig. 17. Modularity of word_association network

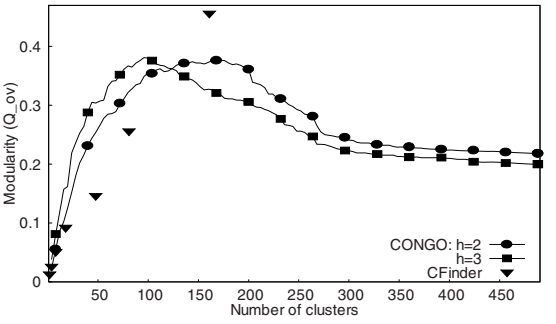


Fig. 18. Modularity of protein-protein network

Finally, we present an experiment on a real-world example with a known solution. Wirz [25] has constructed an ego-graph of his "friends" on Facebook, the social network website. An *ego-graph* of v is a network whose vertices are the "friends" of v and whose edges are the "friend" links between these vertices; v itself is excluded from the network in this experiment. The network has 84 vertices, in five components, and 351 edges. Wirz has manually classified the 84 vertices into ten communities and three isolated vertices, based on personal judgement and without knowledge of the network structure. CONGA, CONGO, and CFinder were run on this network.

Table 2 shows the results for ten clusters (the correct number), seven clusters (the clustering that maximizes modularity), and six clusters (at which CFinder found its highest-modularity solution, for k=3). CONGO, for h=3 or more, has higher F-measure and modularity than CFinder; for h=2, CONGO performs slightly worse than CFinder, but has higher modularity for 11 or more clusters (not shown in the table).

Table 2. Results on ego-graph network

F-measure

	c=10	**c=7**	**c=6**
CONGA	0.353	0.310	0.225
h=3	0.345	0.310	0.225
h=2	0.281	0.209	0.225
CFinder			0.303

Modularity

	c=10	**c=7**	**c=6**
CONGA	0.704	0.912	0.886
h=3	0.722	0.912	0.886
h=2	0.607	0.834	0.886
CFinder			0.858

4 Conclusions

The results presented in Section 3 show that CONGO can be an effective and fast algorithm for detecting overlapping communities in networks. Compared with CONGA, it is substantially faster, especially for h=2. Indeed, almost all of the real-world networks used in Section 3 are too large for CONGA to process in a reasonable time. CONGO is slightly less accurate than CONGA in most, but not all, cases. As discussed in [10], we believe this is because local betweenness is unable to correctly identify communities whose diameter is much larger than h. When communities have a small diameter, CONGO can be at least as accurate as CONGA.

Compared with CFinder, CONGO has similar execution time. CFinder is faster for small examples but CONGO, with h=2, is faster for larger networks. For synthetic networks, CONGO appears generally more accurate than CFinder. For real-world networks, modularity is usually better for CONGO, but CFinder often finds one solution with high modularity: this is for k=3 in all cases except the "jazz" network. Looking at these high-modularity solutions in more detail reveals a few large clusters and many small (mostly 3-vertex) clusters; in addition, there are many vertices that appear in no clusters. The sizes of CONGO's clusters are generally more balanced. It is hard to say which type of solution is best, except perhaps by comparing the computed clusterings with a "ground truth" solution, as in the "ego-graph" network of Section 3.

In conclusion, we believe that CONGO with h=2 is ideally suited to finding overlapping communities in very large networks. For smaller networks, where a solution can be found quickly, h can be increased for more accurate results.

Future work includes improving the CONGO algorithm. One issue is the value of the parameter. h=2 is fast and usually effective, but sometimes a larger value is required; h=3 is sometimes more effective, but too slow in general. The best value for h seems to depend on the diameter of communities, which might vary widely in real-world networks. It may be better to allow the length of shortest paths to vary dynamically in different parts of a network to suit the diameter of each community.

Another possible improvement is to introduce "belonging coefficients", showing how strongly each vertex belongs to each cluster. This should make solutions more informative than in the current algorithm, in which vertices belong equally to all their clusters. For example, partitioning a network into {1,2,3} and {1,2,3,4,5,6} seems meaningless, but if vertices 1, 2, and 3 belong more strongly to the first cluster than the second, the solution is more like two clusters {1,2,3} and {4,5,6} that overlap.

Further information related to this paper, including the implementation, networks analysed, and results, can be found at http://www.cs.bris.ac.uk/~steve/networks/ .

Acknowledgements. I am very grateful to Peter Flach for comments on a draft of this paper. Thanks are also due to Vincenzo Nicosia for explaining the details of his extended modularity measure, and to Martin Wirz for providing his ego-graph data.

References

1. Adamcsek, B., Palla, G., Farkas, I., Derényi, I., Vicsek, T.: CFinder: Locating Cliques and Overlapping Modules in Biological Networks. Bioinformatics 22, 1021–1023 (2006)
2. Baumes, J., Goldberg, M., Magdon-Ismail, M.: Efficient Identification of Overlapping Communities. In: Kantor, P., Muresan, G., Roberts, F., Zeng, D.D., Wang, F.-Y., Chen, H., Merkle, R.C. (eds.) ISI 2005. LNCS, vol. 3495, pp. 27–36. Springer, Heidelberg (2005)
3. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast Unfolding of Community Hierarchies in Large Networks. Eprint arXiv:0803.0476v1 at arxiv.org (2008)
4. Boguña, M., Pastor-Satorras, R., Diaz-Guilera, A., Arenas, A.: Phys. Phys. Rev. E 70, 056122 (2004)
5. Clauset, A., Moore, C., Newman, M.E.J.: Structural Inference of Hierarchies in Networks. In: Statistical Network Analysis: Models, Issues, and New Directions, pp. 1–13 (2007)

6. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing Community Structure Identification. J. Stat. Mech., P09008 (2005)
7. Fred, A.L.N., Jain, A.K.: Robust Data Clustering. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 128–133. IEEE Press, New York (2003)
8. Girvan, M., Newman, M.E.J.: Community Structure in Social and Biological Networks. Proc. Natl. Acad. Sci. USA 99, 7821–7826 (2002)
9. Gregory, S.: An Algorithm to Find Overlapping Community Structure in Networks. In: Kok, J.N., Koronacki, J., López de Mántaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 91–102. Springer, Heidelberg (2007)
10. Gregory, S.: Local Betweenness for Finding Communities in Networks. Technical report, University of Bristol (2008)
11. Lancichinetti, A., Fortunato, S., Kertesz, J.: Detecting the Overlapping and Hierarchical Community Structure of Complex Networks. Eprint arXiv:0802.1218v1 at arxiv.org (2008)
12. Nelson, D.L., McEvoy, C.L., Schreiber, T.A.: The University of South Florida Word Association, Rhyme and Word Fragment Norms (1998), `http://w3.usf.edu/FreeAssociation/`
13. Newman, M.E.J.: The Structure of Scientific Collaboration Networks. Proc. Natl. Acad. Sci. USA 98, 404–409 (2001)
14. Newman, M.E.J.: Detecting Community Structure in Networks. Eur. Phys. J. B 38, 321–330 (2004)
15. Newman, M.E.J.: Fast Algorithm for Detecting Community Structure in Networks. Phys. Rev. E 69, 066133 (2004)
16. Newman, M.E.J.: Finding Community Structure in Networks Using the Eigenvectors of Matrices. Phys. Rev. E 74, 036104 (2006)
17. Newman, M.E.J.: Modularity and Community Structure in Networks. Proc. Natl. Acad. Sci. USA 103, 8577–8582 (2006)
18. Newman, M.E.J., Girvan, M.: Finding and Evaluating Community Structure in Networks. Phys. Rev. E 69, 026113 (2004)
19. Nicosia, V., Mangioni, G., Carchiolo, V., Malgeri, M.: Extending Modularity Definition for Directed Graphs with Overlapping Communities. Eprint arXiv:0801.1647v3 at arxiv.org (2008)
20. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. Nature 435, 814–818 (2005)
21. Pons, P., Latapy, M.: Computing Communities in Large Networks Using Random Walks. J. Graph Algorithms and Applications 10(2), 191–218 (2006)
22. Rand, W.M.: Objective Criteria for the Evaluation of Clustering Methods. J. Am. Stat. Assoc. 66, 846–850 (1971)
23. Salvetti, F., Srinivasan, S.: Local Flow Betweenness Centrality for Clustering Community Graphs. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 531–544. Springer, Heidelberg (2005)
24. Wakita, K., Tsurumi, T.: Finding Community Structure in Mega-scale Social Networks. In: 16th International World Wide Web Conference, WWW 2007, pp. 1275–1276 (2007)
25. Wirz, M.: Personal communication
26. Xie, N.: Social Network Analysis of Blogs. M.Sc Dissertation. University of Bristol (2006)
27. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: SCAN: a Structural Clustering Algorithm for Networks. In: 13th International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 824–833. ACM, New York (2007)
28. Zhang, S., Wang, R., Zhang, X.: Identification of Overlapping Community Structure in Complex Networks Using Fuzzy C-means Clustering. Physica A: Statistical Mechanics and its Applications 374(1), 483–490 (2007)

A Case Study in Sequential Pattern Mining for IT-Operational Risk

Valerio Grossi, Andrea Romei, and Salvatore Ruggieri

Dipartimento di Informatica, Università di Pisa
Largo Bruno Pontecorvo 3, 56127 Pisa Italy
{vgrossi,romei,ruggieri}@di.unipi.it

Abstract. IT-operational risk management consists of identifying, assessing, monitoring and mitigating the adverse risks of loss resulting from hardware and software system failures. We present a case study in IT-operational risk measurement in the context of a network of Private Branch eXchanges (PBXs). The approach relies on preprocessing and data mining tasks for the extraction of sequential patterns and their exploitation in the definition of a measure called *expected risk*.

Keywords: sequential pattern, pre and post-processing, operational risk.

1 Introduction

According to the International Convergence of Capital Measurement and Capital Standards, known as Basel II [5], *operational risk* is "the risk of loss resulting from inadequate or failed internal processes, people and systems, or from external events". In the specific event of business disruption and system failures (e.g., hardware and software failures), the term Information-Technology (IT) operational risk is adopted.

Operational risk management consists of identifying, assessing, monitoring and mitigating the most potentially adverse risks [2,4,7]. On the basis of the risk management evaluation of an organization, the board of directors, regulations, shareholders, or the highly competitive market may require the organization to revise its internal processes, to set aside capital, to subscribe insurance policies, or to make investments in order to cover the residual risk.

Risk identification and assessment is conducted at the level of business units or processes by self-assessment against a checklist of potential vulnerabilities, or by collecting a set of statistics or metrics called risk indicators, or – by increasing the sophistication level – by formal risk quantification against measures of the distributions of frequency and impact of losses. In this sense, the risk of an event is formally defined as the "probability of the event" $\times$ "loss due to the event".

Risk monitoring and mitigation consists of regularly monitoring operational loss events, providing early warning indicators of an increased risk of future losses, and promptly mitigating the risk by reducing the exposure to, or the frequency and/or the impact of loss events.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 424–439, 2008.

The literature on operational, financial and market risk assessment accounts for contributions from statistics and simulation. Statistical models are based on the characterization of loss distributions or, at least, of certain parameters such as the expected loss and the tail loss. High frequency low impact risks (such as transaction processing errors) are modelled by the expected loss and the standard deviation of loss. Risk mitigation consists of acting on the organization processes, infrastructure and personnel. Low frequency high impact risks (such as frauds or earthquakes) are modelled by the tail of the loss distribution. Risk mitigation consists of setting aside capital or to subscribe an insurance. Existing statistical approaches consider mainly the low frequency high impact risks, such as in the approaches of Value at Risk [8], coherent measures [3], and extreme value theory [9]. Bayesian Networks have been adopted [2, Chapter 14] [6] as a powerful tool to cope with shortage of data (as in rare events), to integrate qualitative prior knowledge (such as expert opinions), and to make what-if scenario analyses.

In this paper, we concentrate on the high frequency low impact class of risk by reporting a case study in IT-operational risk in the context of a network of Private Branch eXchanges (PBX). We adopt sequential pattern mining on weighted sequences for the purpose of defining and validating the notion of *expected risk* as a predictive measure of the risk in managing the network of PBX's. We report the problems found and the solutions adopted both in the data preprocessing task and in the sequential pattern extraction and deployment task. To the best of our knowledge, this is the first paper reporting the implementation of a KDD process in the IT-operational risk context.

2 Case Study Specification

2.1 Monitoring a Network of PBX's

We introduce a case study concerning the management of a network of PBX's by a Communication Service Provider (CSP). The customers of the CSP are small-medium enterprisers requiring both voice and data lines at their premises at different contractually agreed quality of services. The customers externalize to the CSP the maintenance of the PBX's and the actual management of the communication services. When a malfunctioning occurs, customers refer to the CSP call center, which can act remotely on the PBX, e.g., to reboot the system. If the problem is not recoverable remotely, as in the case of hardware failure, a technician is sent on-site. Both call center contacts and technician reports are logged in the CSP customer relationship management database.

A PBX is doubly redundant, i.e., it actually consists of two independent communication apparatuses and a self-monitoring software. Automatic alarms produced by the equipment are recorded in the PBX system log. Call center operators can access the system log to control the status of the PBX. Also, a centralized monitoring software collects on a regular basis system logs from all the installed PBX's.

Among the operational risk events, PBX malfunctioning may have different impact on the CSP. At one extreme, the call center operator can immediately

Table 1. [TECH-DB] Log of technician on-site interventions

Attribute	Description
`Date`	Problem opening date and time
`PBX-ID`	Unique ID of the Private Branch eXchange
`CType`	Customer line of business
`Tech-ID`	Unique ID of technician's intervention
`Severity`	Problem severity recorded after problem solution
`Prob-ID`	Problem type recorded after problem solution

solve the problem. At the other extreme, a technician intervention may be required, with the customer's offices inoperative in the meantime, and the risk that the agreed quality of service could not be guaranteed. To record the impact of a malfunctioning, the severity level of the problem occurred is evaluated and documented by the technician. In this context, our case study aims at:

- the Extraction, Transformation and Loading (ETL) into a merged database of the available sources of data, including customer type information, call center logs, technicians reports, and PBX system logs;
- the characterization of early warnings of problems in terms of typical sequences of alarms that lead to them;
- the exploitation of the sequential patterns extracted for automatic on-line malfunctioning prediction and risk quantification.

2.2 Data Sources

Data has been provided by a leading regional CSP. The CSP's customer relationship management system records in the [TECH-DB] table the history of technician interventions at the customer premises. For each problem, at least the following attributes of information are available.

The `Date` attribute consists of the problem opening date and time, defined as the time the call center receives a customer call reporting the notice of a malfunctioning. The `PBX-ID` is a unique identifier of the involved PBX. If a customer has installed more than one PBX, this is determined by the call center operator based on the customer description of the problem and the available configuration of PBX's installed at the customer premise. `CType` is the customer line of business, accordingly to a CRM categorization including: banks, health care, industry, insurance and telecommunication businesses. The `Tech-ID` attribute is a unique identifier of the technician intervention: during a same intervention one or more problems may be tackled. `Severity` is a measure of the impact of the problem. It is defined on a scale from 1 to 3 as follows:

3 critical, service unavailable;
2 medium, intermittent service interruptions;
1 low level problem.

Finally, the `Prob-ID` attribute is a coding of the malfunctioning solved by the technician. Two hundred problem descriptions are codified.

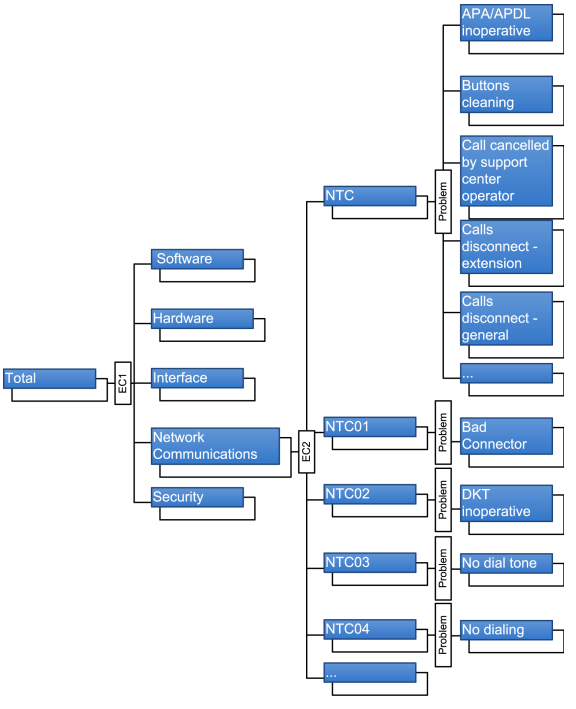


Fig. 1. A hierarchy of problem types for PBX malfunctioning

Table 2. [ALARM-DB] Log of PBX alarms

Attribute	Description
`PBX-ID`	Unique ID of the Private Branch eXchange
`TestDate`	Date and time log was downloaded
`Alarms`	Sequence of alarms raised since last log download

The second data source is the collection of logs generated by PBX's. Logs are downloaded into a centralized repository, called [ALARM-DB], on a regular round-robin basis.

For a given PBX identifier `PBX-ID` and date-time of log download `TestDate`, [ALARM-DB] stores the set of alarms raised by the PBX since the previous log download. Sixteen distinct alarms are available in the data. Unfortunately, the precise time an alarm is raised is not stored in the PBX log.

3 Preprocessing IT-Operational Logs

PBX logs stored in the [ALARM-DB] table and technician's reports stored in the [TECH-DB] table are not directly suitable as an input for sequential pattern mining. In this section, we report two main issues with pre-processing those data

in order to yield a database of sequences. The first issue is concerned with the granularity level in the analysis of PBX problem types. The second one with the problem of building the sequences of alarms related to a PBX problem. Preprocessing has been automated as a collection of ETL data flows developed using the data integration module of the Pentaho suite [11]. A GUI written in Java puts together the various data flows in a stand-alone application.

3.1 A Hierarchy of Problem Types

Since two hundred problem types it is too fine-grained detail, problem types are organized in a three level hierarchy, which is partly shown in Figure 1. The lowest level is the problem type. The highest level (EC1) consists of the Basel II event categories for IT-operational risk: software, hardware, interface, security and network communications. The middle level (EC2) is an intermediate categorization left at the choice of the user during the preprocessing phase.

Every problem type readily falls in one of the five EC1 level members. However, the mapping cannot be automated for a new problem type, and then the preprocessing GUI asks the user to provide the EC1 and EC2 levels for a previously unseen problem type. In the rest of this paper, we will concentrate our attention at the level of EC1, but we point out that, if large volume of data is available, the overall approach can be followed at finer levels of detail.

3.2 An Heuristics for Joining Alarm and Problem Logs

For a technician intervention, the sequence of alarms generated by the involved PBX has to be reconstructed in order to relate a malfunctioning of the PBX with the alarms raised by it. Unfortunately, the problem cannot be directly solved. While a technician intervention records the timestamp it occurred, the alarm log contains the timestamp an alarm is downloaded to the central repository, not the timestamp the alarm is raised. Since alarm log collection may occur once every a few days, this is an issue.

We adopt the following heuristics. Let *Opening* be the timestamp a problem for a PBX is opened. This is available in the [TECH-DB] table. We leave the user the choice to join the problem with the sequence of alarms collected in the time interval [*Opening-Diff1*, *Opening+Diff2*], where *Diff1* and *Diff2* are parameters of the preprocessing procedure.

3.3 A Database of Sequences

The preprocessing of available input produced a database of 1899 sequences, spanning over a period of four months, of the form:

```
Date: 22/02/2007 8.36
CType: Bank
PBX-ID: 90333
Prob-ID: Hardware
Severity: 2
```

```
AlarmSequence:
   {CARD}:19/02/2007 18:34:01;
   {}:20/02/2007 18:12:53;
   {PCM TIME SLOT}:21/02/2007 17:15:21
--
Date: 21/02/2007 16.54
CType: Health
PBX-ID: 91993
Prob-ID: Security
Severity: 2
AlarmSequence:
   {DIGITAL TRUNK CARD,DKT SUBUNIT}:19/02/2007 17:56:47;
   {POWER_SUPPLY}:21/02/2007 09:10:07
```

The attributes `Date`, `CType`, `PBX-ID` and `Severity` are directly taken from [TECH-DB]. `Prob-ID` is obtained by lifting the problem description to its EC1 category in the hierarchy of Figure 1. Finally, the sequence of sets of alarms, labelled by the date each set is collected, is obtained by joining alarm and problem logs. In the first sequence above, a `CARD` alarm is raised on 19/02/2007, then no alarm on the next day, a `PCM TIME SLOT` alarm on the 21/02/2007, and finally a malfunctioning is reported to the call center on 22/02/2007. As parameters for the joining heuristics, we have set *Diff1* to three months and *Diff2* to one day. Setting *Diff1* as large as possible is desirable, but it is important not to overlap with the time interval of a past problem for the same PBX. Concerning *Diff2*, it should be set to the average period that alarms are collected into the centralized repository. Assuming the period is one day, an alarm raised the same day of a customer call could or could not be already processed by ETL flows at the time of the call. Without any further information, and considering that such alarms are the most valuable for prediction, we assume they have been processed.

4 Sequential Pattern Mining for Risk Assessment

4.1 Sequences and Sequential Patterns

Let us recall a few standard definitions [1,10]. Given a finite set $\mathcal{I}$ of items, an itemset T is a subset of $\mathcal{I}$. A sequence is an ordered list of itemsets $\langle T_1, \ldots, T_n \rangle$. For brevity, we write $\langle t_1, \ldots, t_n \rangle$ if for $i = 1 \ldots n$, $T_i = \{t_i\}$, i.e., all T_i's are singletons. A sequence $s_1 = \langle T_1, \ldots, T_n \rangle$ is a sub-sequence of another sequence $s_2 = \langle S_1, \ldots, S_m \rangle$ (or s_2 is a super-sequence of s_1), denoted as $s_1 \sqsubseteq s_2$, if there exists $1 \leq p_1 < \ldots < p_n \leq m$ such that for $i = 1 \ldots n$, $T_i \subseteq S_{p_i}$. The concatenation of two sequences s_1 and s_2 is denoted by $s_1 \cdot s_2$. A sequence database $\mathcal{D}$ is a collection of sequences. The (relative) support (w.r.t. $\mathcal{D}$) of a sequence s is defined as the fraction of sequences in $\mathcal{D}$ which are sub-sequences of s, i.e., $supp(s) = |\{s' \in \mathcal{D} \mid s' \sqsubseteq s\}|/|\mathcal{D}|$. A sequential pattern is a sequence $SP = \langle T_1, \ldots, T_n \rangle$ such that $supp(SP) \geq \xi$, where ξ is a fixed minimum support

threshold. For notational purposes, it is convenient to label sequential patterns with their support, and to differentiate them from sequences. Therefore, we write the sequential pattern SP in the form $T_1 \rightarrow T_2 \rightarrow \ldots T_n[s]$, where $s = supp(SP)$. A sequence $s \in \mathcal{D}$ supports SP if $SP \sqsubseteq s$. A sequential pattern is maximal if there is no other sequential pattern that is a super-sequence of it. We denote by $Max\ \mathcal{S}$ the set of maximal sequential patterns in $\mathcal{S}$. Restricting to maximal sequential patterns alleviates the problem of dealing with an exponential number of extracted patterns. Several algorithms have been proposed in the literature to extract (maximal) sequential patterns [10].

4.2 Adapting Sequences and Sequential Patterns

For the problem under consideration, we will make use of a variant of sequential pattern mining, where sequences and sequential patterns are weighted.

In our context, we fix the set of items $\mathcal{I}$ to include alarms identifiers, problem type items $\texttt{Prob-ID}= \beta$ and business line items $\texttt{CType}= \alpha$, where β is an EC1-level problem type and α is the customer line of business. A sequence is now of the form:

$$\texttt{CType} = \alpha\ ,\ \texttt{AlarmSet}_1\ ,\ \ldots \texttt{AlarmSet}_k\ ,\ \texttt{Prob-ID} = \beta\ [sev]$$

For an occurrence of a PBX problem recorded in the technician's database, a sequence models the temporal succession of alarms $\texttt{AlarmSet}_i$ raised by the PBX. The heuristics described in Sect. 3.2 is adopted in order to join a problem occurrence with the succession of alarms related to it.

A sequence starts with $\texttt{CType} = \alpha$, ends with $\texttt{Prob-ID} = \beta$, and it is labelled with the problem severity sev:

- Starting with $\texttt{CType} = \alpha$ is a work-around to differentiate the sequences on the basis of the type of business of customers. This is motivated by the requirement that risk management has to differentiate risk for line of business or processes. Therefore, we are interested in modeling specific patterns of problems due to the different usages of the PBX network by different businesses.
- Similarly, ending the sequence with $\texttt{Prob-ID} = \beta$ allows for isolating patterns that lead to a specific problem from patterns that hold in general.
- Labelling the sequence with the problem severity is a weighting strategy, based on the impact of the problem occurred. For a sequence s, we write $SEV(s)$ to denote its severity label.

As a result of the above definition of sequences, we are interested in extracting from a database of sequences $\mathcal{TS}$, which we call the training set, sequential patterns SP of the form:

$$\texttt{CType} = \alpha \rightarrow \texttt{AlarmSet}_1 \rightarrow \ldots \texttt{AlarmSet}_n \rightarrow \texttt{Prob-ID} = \beta\ [supp, sev] \quad (\star)$$

where:

- the severity label *sev* is $(\Sigma_{s\in\mathcal{TS},SP\sqsubseteq s}SEV(s))/|\{s \in \mathcal{TS} \mid SP \sqsubseteq s\}|$, i.e., the average severity of sequences in the training set supporting the sequential pattern;
- the support label *supp* is $supp(SP)/supp(\langle \texttt{CType} = \alpha, \texttt{Prob-ID}= \beta\rangle)$, i.e., the relative support of the sequential pattern w.r.t. the number of sequences starting with $\texttt{CType} = \alpha$ and ending with $\texttt{Prob-ID} = \beta$.

Sequential patterns of the form $(\star)$ can be extracted by the following procedure. First, split the sequence database into one database for each distinct pair $(\texttt{CType} = \alpha, \texttt{Prob-ID} = \beta)$; then run any sequential pattern extraction algorithm from the literature on each sequence database and for a specified minimum support threshold; then remove extracted sequential patterns not including an item Prob-ID as the last item; finally, calculate severity of a sequential pattern by averaging severities of the sequences supporting it. Alternatively, multidimensional [12] or context-based [13] approaches to sequential pattern mining could be adapted.

4.3 Mean Risk

Consider an initial fragment $s_1 = \langle \texttt{CType}= \alpha_1, \texttt{AlarmSet}_1 , \ldots, \texttt{AlarmSet}_k\rangle$ of a sequence. Assume for the moment that we do not or cannot exploit any sequential pattern. How can we then define the risk that s is followed by $\texttt{Prob-ID} = \beta$? Since the risk of an event is "probability of the event" × "loss due to the event", we approximate:

- the "probability of the event" as the ratio:

$$p_{\alpha,\beta} = \frac{supp(\langle \texttt{CType} = \alpha, \texttt{Prob-ID} = \beta\rangle)}{supp(\langle \texttt{CType} = \alpha\rangle)},$$

  i.e., the confidence that a sequence in the training set starting with $\texttt{CType} = \alpha$ will be followed by $\texttt{Prob-ID} = \beta$;
- the "loss due to the event" as the average severity of sequences in the training set starting with $\texttt{CType} = \alpha$ and ending with $\texttt{Prob-ID} = \beta$:

$$l_{\alpha,\beta} = \frac{\Sigma_{s\in\mathcal{TS},\langle \texttt{CType}=\alpha,\texttt{Prob-ID}=\beta\rangle\sqsubseteq s}SEV(s)}{|\{s \in \mathcal{TS} \mid \langle \texttt{CType} = \alpha, \texttt{Prob-ID} = \beta\rangle \sqsubseteq s\}|}.$$

We define the *mean risk* that the initial fragment s_1 starting with $\texttt{CType} = \alpha$ will end with $\texttt{Prob-ID} = \beta$ as $p_{\alpha,\beta} \times l_{\alpha,\beta}$. If s_1 does not start with $\texttt{CType} = \alpha$, i.e. $\alpha_1 \neq \alpha$, then the mean risk is zero, as one could expect. Formally:

$$\text{MRISK}(\alpha, \beta, s_1) = \begin{cases} p_{\alpha,\beta} \times l_{\alpha,\beta} & \text{if } \langle \texttt{CType} = \alpha\rangle \sqsubseteq s_1 \\ 0 & \text{otherwise.} \end{cases}$$

Summary mean risk for a line of business w.r.t. all problem types is defined as the sum of the risk for individual problem types:

$$\text{SMRISK}(\alpha, s_1) = \Sigma_{\beta\in dom(\texttt{Prob-ID})}\text{MRISK}(\alpha, \beta, s_1).$$

When $\alpha = \alpha_1$, it can be rewritten as: $1 \times \frac{\Sigma_{s \in \mathcal{TS}, \langle \texttt{CType=}\alpha \rangle \sqsubseteq s} SEV(s)}{|\{s \in \mathcal{TS} \mid \langle \texttt{CType=}\alpha \rangle \sqsubseteq s\}|}$, namely "probability of any problem" $\times$ "average severity" for the line of business. Similarly, we define the summary mean risk for a problem type w.r.t. all lines of business:

$$\text{SMRISK}(\beta, s_1) = \text{MRISK}(\alpha_1, \beta, s_1) = \Sigma_{\alpha \in dom(\texttt{CType})} \text{MRISK}(\alpha, \beta, s_1).$$

The three definitions above extend to a set S of initial fragments by summing up the individual risk of each element in S.

4.4 From Sequential Patterns to Expected Risk

Consider now a specific initial fragment $s = \langle$`CType=Bank`, `CARD` , `CARD SUBUNIT`, `DIGITAL TRUNK CARD`$\rangle$ of a sequence. We would like to adopt the sequential patterns extracted from the training set to the purpose of enhancing the notion of mean risk to a notion called *expected risk*. Intuitively, we start by looking at the sequential patterns supported by s (with the exclusion of the last item in the sequential pattern – i.e., of the `Prob-ID` item). Assume that the set $\mathcal{SP}$ of such sequential patterns consists of:

SP$_0$ `CType=Bank` $\rightarrow$ `CARD SUBUNIT` $\rightarrow$ `Prob-ID=Software` [0.4, 3]
SP$_1$ `CType=Bank` $\rightarrow$ `CARD` $\rightarrow$ `Prob-ID=Hardware` [0.2, 2]
SP$_2$ `CType=Bank` $\rightarrow$ `CARD SUBUNIT` $\rightarrow$ `Prob-ID=Hardware` [0.3, 3]
SP$_3$ `CType=Bank` $\rightarrow$ `CARD` $\rightarrow$ `CARD SUBUNIT` $\rightarrow$ `Prob-ID=Hardware` [0.1, 3]
SP$_4$ `CType=Bank` $\rightarrow$ `DIGITAL TRUNK CARD` $\rightarrow$ `Prob-ID=Hardware` [0.2, 2]

For `Prob-ID=Software`, there is only one supported sequential pattern, namely SP_0. By recalling that the risk of an event is "probability of the event" $\times$ "loss due to the event", we set the expected risk (to have a problem with software) to $0.4 \times 3 = 1.2$.

Consider now `Prob-ID=Hardware`. First, we observe that SP_1 is a sub-sequence of SP_3, hence it is superseded by SP_3, and similarly for SP_2. Therefore, we restrict to maximal sequential patterns in $\mathcal{SP}$, namely to SP_3 and SP_4. We now define the expected risk to have a hardware problem by averaging the severities of SP_3 and SP_4 based on their support, i.e., as $(0.1 \times 3 + 0.2 \times 2)/(0.1 + 0.2) = 0.7/0.3 = 2.33$. Notice that we scale the average severity by dividing by the sum of the supports of the maximal sequential patterns. The motivation is twofold. On the one side, two (maximal) sequential patterns may have some common supporting sequence, hence the sum of their supports can be strictly greater than one. On the other side, the set of maximal sequential patterns may not cover all possible sequences (rare ones cannot be modeled by sequential patterns by definition), i.e., the sum of their supports can be strictly lower than one.

Finally, let us consider `Prob-ID=Interface`. There is no sequential pattern supported by s, at least for the fixed minimum support threshold. Therefore, the reasoning followed so far cannot be applied. Intuitively, we fall in the case that no sequential information is available, i.e., on the notion of mean risk, and then we set the expected risk to the mean risk.

Let us introduce some notation. Let $\mathcal{SP}$ be the set of sequential patterns extracted from the training set $\mathcal{TS}$. For a given initial fragment s_1 and problem

type β, we define the set of sequential patterns supported by s_1 and that are maximal as:

$$\mathcal{M}(s_1, \beta) = Max\{SP \in \mathcal{SP} \mid SP \sqsubseteq s_1 \cdot \langle \texttt{Prob-ID} = \beta \rangle\}.$$

The *expected risk* is then defined as:

$$\text{ERISK}(\alpha, \beta, s_1) = \begin{cases} \dfrac{\Sigma_{SP \in \mathcal{M}(s_1,\beta)} supp(SP) \times sev(SP)}{\Sigma_{SP \in \mathcal{M}(s_1,\beta)} supp(SP)} & \text{if } \mathcal{M}(s_1, \beta) \neq \emptyset \\ \text{MRISK}(\alpha, \beta, s_1) & \text{otherwise.} \end{cases}$$

The definition readily extends to a set S of initial fragments, to summaries for line of business and problem type as follows:

$$\text{ERISK}(\alpha, \beta, S) = \Sigma_{s_1 \in S} \text{ERISK}(\alpha, \beta, s_1)$$
$$\text{SERISK}(\alpha, S) = \Sigma_{\beta \in dom(\texttt{Prob-ID})} \text{ERISK}(\alpha, \beta, S)$$
$$\text{SERISK}(\beta, S) = \Sigma_{\alpha \in dom(\texttt{CType})} \text{ERISK}(\alpha, \beta, S).$$

Notice that in the special case that there is no sequential pattern, i.e. $\mathcal{SP} = \emptyset$, these definitions collapse to the ones for the mean risk.

One could consider removing non-maximal sequential patterns off-line when sequential patterns are extracted, whilst now they are removed on-line when the expected risk is calculated. Unfortunately, this approach does not lead to the same results. In fact, consider an initial fragment $s = \langle \texttt{CType=Bank}, \texttt{ALARM}_1 \rangle$. If non-maximal sequential patterns are removed off-line, then SP_1 above could not be taken into consideration. Moreover, no maximal sequential pattern SP_3-SP_4 is supported by s. Summarizing, we have no sequential pattern to exploit in defining the expected risk of s.

4.5 Actual Risk

Once an initial fragment completes to a full sequence s, i.e., it is terminated by a problem type item $\texttt{Prob-ID} = \beta$ and labelled by severity $SEV(s)$, it is easy to calculate the involved risk. We define the *actual risk* of the sequence as $1 \times SEV(s)$, and this value contributes only to problem type β:

$$\text{ARISK}(\alpha, \beta, s) = \begin{cases} SEV(s) & \text{if } \langle \texttt{CType} = \alpha, \texttt{Prob-ID} = \beta \rangle \sqsubseteq s \\ 0 & \text{otherwise.} \end{cases}$$

The measure readily extends to sets of sequences, and to summaries for lines of business and problem types as done for mean risk and expected risk.

5 Deploying Sequential Patterns

5.1 Application Scenario

Consider a set of sequential patterns extracted from a training set of past sequences. How can the notion of expected risk be turned into practice for risk

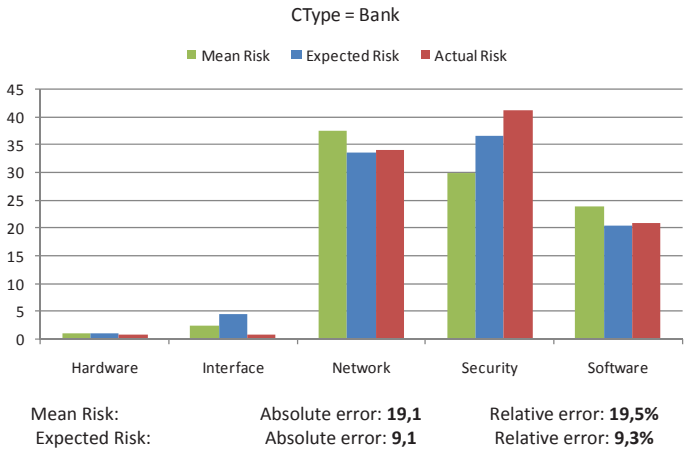


Fig. 2. Expected risk vs actual risk: detail for the `Bank` customer line of business

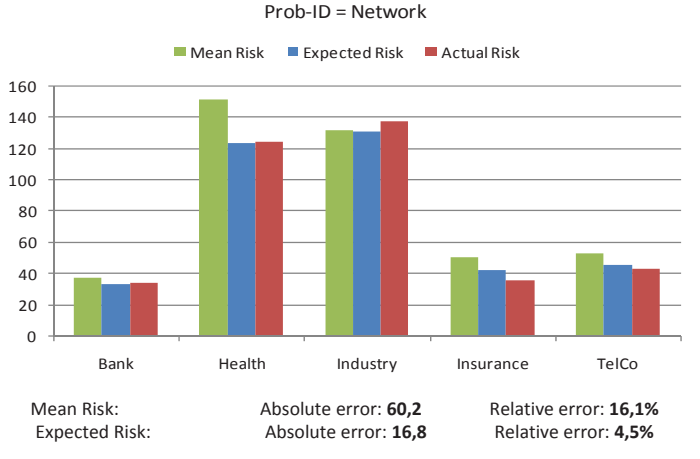


Fig. 3. Expected risk vs. actual risk: detail for `Network` problem type

assessment and mitigation? Of course, the analysis of the sequential patterns by a domain expert might highlight previously unknown patterns of alarms leading to malfunctioning with high average severity.

Besides this descriptive usage, we concentrate here on the deployment of extracted patterns in on-line risk assessment. Let us assume an application scenario where a call center operator receives a call from a customer reporting a malfunctioning. A ticket is open for dealing with the malfunctioning. At the time of the call, the following information is known: the customer (line of business), the involved PBX, the sequence of alarms of that PBX collected up to that time. In other words, an initial fragment s_1 is available, as assumed in Section 4.4.

A decision support system can then exploit the notion of expected risk to estimate the overall risk of the currently open tickets. This information is useful

to assess the current level of risk for the various customer lines of business and problem types. Operatively, it can help in determining the needed effort in terms of required expertise for technician's interventions, e.g., how many software and hardware technicians should be alerted.

When the tickets are closed and the technician reports are available, the actual risk can be calculated. The comparison between the expected risk and the actual risk provides a measure of the accuracy of the notion of expected risk. Also, since expected risk is a refinement of mean risk, it is worth evaluating the improvement of accuracy of expected risk over mean risk. The next two subsections report experimental results on those two accuracy issues.

In particular, the experimental results are obtained by partitioning the available database of sequences (see Section 3.3) into temporally separated training set $\mathcal{TS}$ and test set $\mathcal{TE}$. The split date-time was set to obtain about a 75%-25% partitioning[1]. Moreover, we set the minimum support threshold in the sequential pattern extraction (see Section 4.2) to 5%.

5.2 Expected Risk vs Mean Risk vs Actual Risk

The overall expected risk for a business line α is represented as an histogram chart with the distinct values β for problem type `Prob-ID` on the X-axis and the value of $\text{ERISK}(\alpha, \beta, \mathcal{TE})$ on the Y-axis. The distributions of the mean risk and of the actual risk measures are represented in the same manner. The difference area between the expected and the actual risk histograms measures the error of adopting expected risk for estimating actual risk. More formally, the absolute error is defined as:

$$\Sigma_{\beta \in dom(\texttt{Prob-ID})} abs(\text{ERISK}(\alpha, \beta, \mathcal{TE}) - \text{ARISK}(\alpha, \beta, \mathcal{TE}))$$

and the relative error is its ratio over $\Sigma_{\beta \in dom(\texttt{Prob-ID})} \text{ARISK}(\alpha, \beta, \mathcal{TE})$. Similar definitions can be stated for the mean risk.

Figure 2 shows the histogram charts of mean, expected and actual risks for `CType=Bank`, namely for the bank subnetwork of PBX's. Apart from the `Interface` problem type, the expected risk measure provides a much better estimation of actual risk than the mean risk. The relative errors for the various business lines are summarized in the following table:

			Relative Error	
CType	**No. Seq.**	**Actual Risk**	**Mean Risk**	**Expected Risk**
Bank	51	98	19.5%	9.3%
Health	185	306	17.0%	9.4%
Industry	159	289	6.8%	4.9%
Insurance	68	127	47.5%	25.0%
TelCo	47	91	31.6%	16.8%

[1] We observe that, in a more realistic scenario, the number of open tickets at a certain time is typically low, especially if compared to closed ones, since problems are usually solved within 48 hours time.

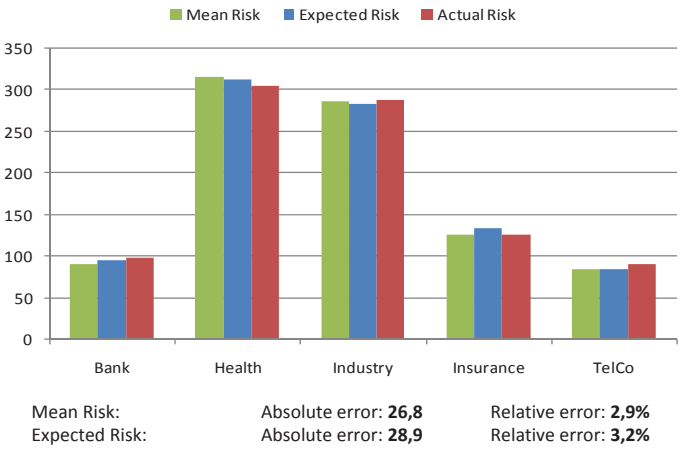


Fig. 4. Expected risk vs. actual risk: summary for lines of business

As a general observation, the relative error of expected risk is near the half of the error of the mean risk. Figure 3 shows the histogram charts of mean, expected and actual risks for `Prob-ID=Network`, namely for the problem types related to the communication network of PBX's, and for the various customer business lines. Expected risk turns out to improve over mean risk for all business lines. Now the difference area between expected and actual risk charts is obtained as:

$$\Sigma_{\alpha \in dom(\texttt{CType})} abs(\text{ERisk}(\alpha, \beta, \mathcal{TE}) - \text{ARisk}(\alpha, \beta, \mathcal{TE})).$$

The relative errors for the various problem types are summarized next:

		Relative Error	
Prob-ID	**Actual Risk**	**Mean Risk**	**Expected Risk**
Hardware	35	20.6%	27.3%
Interface	24	42.6%	69.0%
Network	374	16.1%	4.5%
Security	241	28.2%	11.0%
Software	237	14.5%	12.5%

For the problem types `Hardware` and `Interface` there is a degradation of performances of expected risk over mean risk, whilst for the other problem types there is a gain. Notice, however, that there is a very low actual risk for `Hardware` and `Interface`, and hence a low number of sequences.

Consider now the summary measure $\text{SERisk}(\alpha, \mathcal{TE})$, which provides for a business line α the expected total risk w.r.t. all problem types. Figure 4 shows the histogram charts of mean, expected and actual risks. By averaging over a whole line of business, the predictive power of mean risk improves, and its refinement to expected risk yields no additional benefit. As stated in the introduction, this

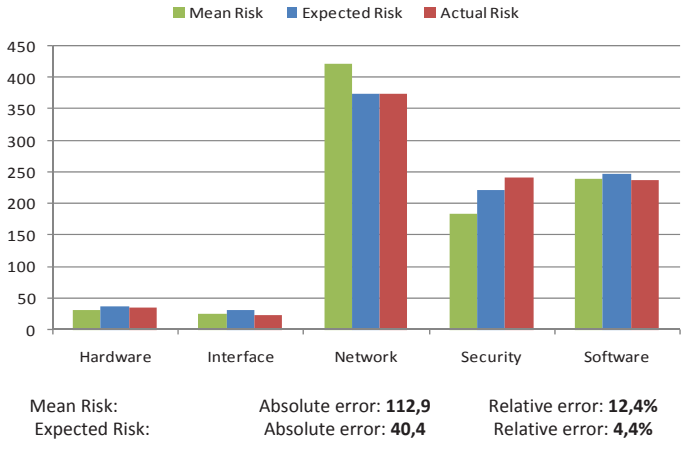


Fig. 5. Expected risk vs. actual risk: summary for problem types

confirms that simple statistics, such as mean and standard deviation, are enough to deal with high-frequency (and necessarily low impact) loss events. As soon as we go into the details of a specific line of business, frequency decreases, and then sequential information leads to better predictive power.

Finally, Figure 5 shows the charts for the summary measure SERisk $(\beta, \mathcal{TE})$, which provides for a problem type β the expected total risk w.r.t. all business lines. By averaging on the problem type, expected risk improves over mean risk considerably.

5.3 Tuning the Parameters

Let us consider here a few issues concerning the choice of parameters in pattern extraction and deployment. Figure 6 shows how the expected risk error is affected by the minimum support threshold. The figure reports the relative error:

$$\frac{abs(\text{ERisk}(\texttt{Health}, \beta, \mathcal{TE}) - \text{ARisk}(\texttt{Health}, \beta, \mathcal{TE}))}{\text{ARisk}(\texttt{Health}, \beta, \mathcal{TE})}$$

for two sample β, and the total relative error:

$$\frac{\Sigma_{\beta \in dom(\texttt{Prob-ID})} abs(\text{ERisk}(\texttt{Health}, \beta, \mathcal{TE}) - \text{ARisk}(\texttt{Health}, \beta, \mathcal{TE}))}{\Sigma_{\beta \in dom(\texttt{Prob-ID})} \text{ARisk}(\texttt{Health}, \beta, \mathcal{TE})}$$

for various minimum support thresholds. It is immediate to observe that lower minimum supports lead to lower errors. However, after reaching some minimum, the error does not improve and it eventually starts increasing.

Another choice we made was to partition the training set based on pairs $(\texttt{CType} = \alpha, \texttt{Prob-ID} = \beta)$, and to extract sequential patterns from each partition. An alternative choice is to partition with respect to $\texttt{CType} = \alpha$ only, i.e.,

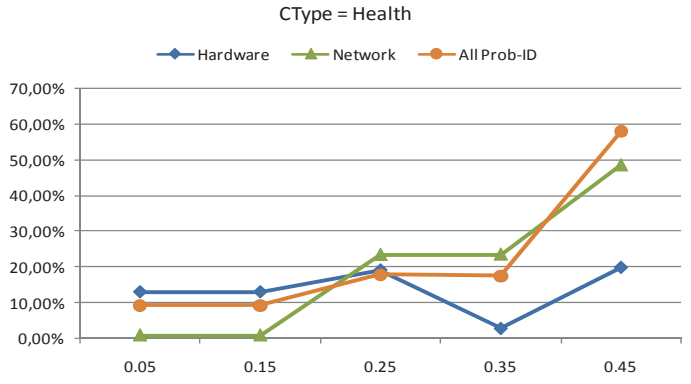


Fig. 6. Expected risk relative error by varying the minimum support

not forcing the extraction of sequential patterns for every problem type. This alternative leads to extract sequential patterns where only frequent problem types appear. We are more accurate for them, but less accurate for minority problem types. As an example, the relative error of expected risk for the `Bank` line of business in Figure 2 degrades to 23.2%.

Moreover, we have also conducted experiments on the maximum number of sequential patterns to be considered for a pair ($\texttt{CType} = \alpha$, $\texttt{Prob-ID} = \beta$) among those having a minimum support. In the experiments reported so far, the number was set to the top 5 (w.r.t. the support) for all pairs. Extensive experimentation shows that, for a same minimum support threshold, a large number of sequential patterns (> 20) leads to poorer performances, and that the optimal number might vary for each pair ($\texttt{CType} = \alpha$, $\texttt{Prob-ID} = \beta$). Therefore, a form of self-tuning of parameters might improve the reported results.

6 Conclusions

Is data mining suitable for IT-operational risk management? We believe that we provided an affirmative answer to the question – yet, preliminary and for a specific case study. For high frequency - low impact loss events, the extraction of frequent (sequential) patterns from past log of data can improve basic measures of risk, which rely only on simple statistics, such as in the case of the mean risk. The improvement consists of more accurate predictions for lower frequency events, as in the case of a specific line of business or problem type. Nevertheless, a frequent pattern approach, like the one proposed here, cannot deal with very low frequency events, which have very few occurrences in databases or none at all. Hence, our approach is complementary to the ones proposed in the statistical and simulation literature for low frequency - high impact loss events.

A further work we intend to pursue is to enhance the basic statistics with a classification based approach: all in all, the mean risk measure is a "decision stump" classifier on the `CType` attribute. A better classifier could be trained

by using additional predictive attributes, such as the PBX hardware/software version, or, in order to evaluate the gain due to sequential information, the presence/absence of alarms.

Acknowledgments. This work has been partly supported by *MUlti-industry, Semantic-based next generation business INtelliGence (MUSING)*, a European RTD project (2006-2010) founded by the 6^{th} Framework Programme - Information Society Technologies, Contract No. FP6-027097, `http://www.musing.eu`.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns: Generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
2. Alexander, C. (ed.): Operational Risk: Regulation, Analysis and Management. Prentice Hall, Englewood Cliffs (2003)
3. Artzner, P., Delbaen, F., Eber, J.M., Heath, D.: Coherent measures of risk. Mathematical Finance 9, 203–228 (1999)
4. Basel Committee on Banking Supervision: Sound Practices for the Management and Supervision of Operational Risk. BIS (2003), `http://www.bis.org/publ/bcbs96.htm`
5. Basel Committee on Banking Supervision: International Convergence of Capital Measurement and Capital Standards: A Revised Framework. BIS (2006), `http://www.bis.org/publ/bcbs128.htm`
6. Cowell, R.G., Verrall, R.J., Yoon, Y.K.: Modeling operational risk with bayesian networks. Journal of Risk & Insurance 74, 795–827 (2007)
7. Davis, E. (ed.): Operational Risk: practical approaches to implementation. Incisive Media Investments (2005)
8. Jorion, P.: Value at Risk: the new benchmark for managing financial risk. McGraw-Hill, New York (2007)
9. Klüppelberg, C.: Risk management with extreme value theory. In: Finkenstädt, B., Rootzén, H. (eds.) Extreme Values in Finance, Telecommunications and Environment, pp. 101–168. Chapman & Hall / CRC, Boca Raton (2003)
10. Li, T.-R., Xu, Y., Ruan, D., Pan, W.-M.: Sequential pattern mining. In: Intelligent Data Mining. Studies in Computational Intelligence, vol. 5, pp. 103–122. Springer, Heidelberg (2005)
11. Pentaho Corporation. Pentaho: Open Source Business Intelligence, Version 1.6 (2008), `http://www.pentaho.com`
12. Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q., Dayal, U.: Multi-dimensional sequential pattern mining. In: Proc. of CIKM 2001, pp. 81–88. ACM, New York (2001)
13. Ziembinski, R.: Algorithms for context based sequential pattern mining. Fundamenta Informaticae 76(4), 495–510 (2007)

Tight Optimistic Estimates for Fast Subgroup Discovery

Henrik Grosskreutz, Stefan Rüping, and Stefan Wrobel

Fraunhofer IAIS, Schloss Birlinghoven, 53754 St. Augustin, Germany
{henrik.grosskreutz,stefan.rueping,stefan.wrobel}@iais.fraunhofer.de

Abstract. Subgroup discovery is the task of finding subgroups of a population which exhibit both distributional unusualness and high generality. Due to the non monotonicity of the corresponding evaluation functions, standard pruning techniques cannot be used for subgroup discovery, requiring the use of optimistic estimate techniques instead. So far, however, optimistic estimate pruning has only been considered for the extremely simple case of a binary target attribute and up to now no attempt was made to move beyond suboptimal heuristic optimistic estimates. In this paper, we show that optimistic estimate pruning can be developed into a sound and highly effective pruning approach for subgroup discovery. Based on a precise definition of optimality we show that previous estimates have been tight only in special cases. Thereafter, we present tight optimistic estimates for the most popular binary and multi-class quality functions, and present a family of increasingly efficient approximations to these optimal functions. As we show in empirical experiments, the use of our newly proposed optimistic estimates can lead to a speed up of an order of magnitude compared to previous approaches.

1 Introduction

Subgroup discovery [Klö96, Wro97] is the task of finding subgroups of a population with high generality and distributional unusualness. It is a general approach that has shown to be useful in a variety of application scenarios, like medical consultation systems [ABP06], spatial analysis [KM02], marketing campaign planning [LCGF04], and also in contrast set mining tasks [KLGK07].

Unfortunately, if applied to real-world problems, subgroup discovery quickly results in excessive computation, due to its exponential dependency on the number of attributes. Different approaches have proposed to cope with that problem: While sampling based approaches [SW00] relax the task by allowing a certain degree of departure from the optimal solution and a (controllable) error probability, other approaches make use of sophisticated data structures [AP06] or heuristics [Klö02, LKFT04].

Another approach, proposed by Wrobel in [Wro97], is to prune the search space using so-called *optimistic estimates*. An optimistic estimate is a function that, given a subgroup s, provides a bound for the quality of every subgroup s' that is a refinement of s. Surprisingly, the use of optimistic estimates for fast

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 440–456, 2008.

subgroup discovery has not yet been developed into a mature technology: until recently optimistic estimates have only been considered for the extremely simple case of a binary target attribute, and even in this case no attempt was made to move beyond suboptimal optimistic estimates.

In this paper, we investigate the question whether the optimistic estimates considered so far provide bounds that are, in some sense, optimal. To this end we provide a formal definition of *tight* optimistic estimates, that is optimistic estimates that are as conservative as possible wrt. the information at hand, namely the size of a subgroup and its distribution over different classes. Using this definition, we show that the optimistic estimate proposed in [Wro97] is not tight. Thereafter, we present new tight optimistic estimates for some of the most common quality functions. We also present a family of increasingly efficient approximations to these optimal functions. While these optimistic estimates are not tight, they have the advantage that they are simpler to calculate.

Summarizing, the main contributions of this paper are thus (i) the formal definition of tightness, (ii) new, tight optimistic estimates for some of the most common quality functions, and (iii) a family of increasingly efficient approximations to these optimal functions. In an experimental section, we show that our results are not only interesting from a theoretical point of view, but also have a significant impact on the performance of subgroup discovery algorithms.

The paper is organized as follows: In section 2, we define the basics of the subgroup discovery task. In section 3, we provide our definition of tight optimistic estimates; thereafter, we present and prove new (tight) optimistic estimates. Section 4 contains the experiments, while section 5 concludes.

2 Preliminaries

In this section, we will introduce our terminology, formally define the problem of subgroup discovery, and motivate the concept of optimistic estimates.[1]

2.1 The Task of Subgroup Discovery

Let $DB = \{R_1, ..., R_N\}$ be a *database* or *dataset*, consisting of N rows, each built up from of $l+1$ values. We distinguish one attribute c, called the *class attribute* with domain $D(c) = \{c_1, ...c_m\}$, from the l ordinary attributes $\{a_1, a_2, ...a_l\}$ with domains $D(a_i) = \{v_{i,1}, ..., v_{i,m_i}\}$. Every database row R_j is an n-tuple $(v_{j,1}, ..., v_{j,l}, c_j)$, and we call c_j its class.

A *subgroup description* sd is a set of *terms* $\{t_1, ..., t_k\}$ where every term t_i is a constraint on an attribute, i.e. t_i has the form $(a_i = v_i)$, $v_i \in D(a_i)$. The *length* of the subgroup description is the number of terms it is built of. We call a subgroup description $sd' = \{t'_1, ..., t'_{k'}\}$ a *refinement* of a subgroup description $sd = \{t_1, ..., t_k\}$, denoted by $sd' \succ sd$, if $\{t_1, ..., t_k\}$ is a subset of $\{t'_1, ..., t'_{k'}\}$.

[1] In the following presentation, we assume that a dataset is provided as a single table. However, the concept of (tight) optimistic estimates also applies to the multi-relational setting involving joins over relations as considered in [Wro97].

Given a database DB and a subgroup description sd, the *subgroup extension* of sd on DB is the set of rows $R_j \in DB$ that satisfy all terms $t_i \in sd$. Please note that if sd' is a refinement of sd, i.e. $sd' \succ sd$, then for every database DB the subgroup extension for DB and sd' is a subset of the subgroup extension for DB and sd.

Given a set of rows $R = \{R_1, ...R_n\}$ (a database or subgroup extension), we call n its *size* and $\boldsymbol{p} = (p_1, ..., p_m)$, where p_i is the fraction of the rows of R of class i, its *class distribution*. Formally, $\boldsymbol{p}$ is defined as follows:

$$p_i := 1/n \times |\{r | r \in R \wedge \mathit{class}(r) = i\}|.$$

Figure 1 shows an example with hypothetical data, inspired from a medical domain. The rows represent medical prescriptions made by doctors. As class attribute, we consider the cost of the prescription. Beside this special attribute, the prescriptions contain the doctor's specialty and the information whether the doctor's practice is in an urban or a rural environment. In this example, $\{Specialty = Surgery\}$ and $\{Specialty = Surgery, Region = Urban\}$ are two subgroup descriptions, and the corresponding subgroups consist of the rows that fulfill these conditions. The size of the subgroup extension of $\{Specialty = Surgery\}$ is 3 and its probability distribution is $p_{high} = 0.33$, $p_{medium} = 0$, $p_{low} = 0.66$.

Cost	Specialty	Region
High	Surgery	Urban
Medium	Internal Med	Urban
Medium	Psychiatry	Urban
Medium	Internal Med	Rural
Low	Surgery	Rural
Low	Surgery	Rural

Fig. 1. Prescription example

A *quality function* q is a mapping from $DB \times sd$ to the reals. Intuitively, a quality function expresses how "interesting" a subgroup is. Almost all quality functions considered in the literature only depend on some parameters of the subgroup and the database, in particular on the size n of the subgroup, the size N of the database, the class distribution $\boldsymbol{p}$ of the subgroup, and the class distribution $\boldsymbol{p_0}$ of the database. Table 1 summarizes some of the most prominent quality functions [Klö96, SW00]: the Piatetsky-Shapiro quality function dealing with the two-class case, and the Split, Gini and Person χ^2 quality functions for n-ary class attributes.[2]

The problem of *subgroup discovery* is defined as follows: Given a database DB, a quality function q, and a number k, determine the k subgroup descriptions with maximum quality. Or, put more formally: return a set of k subgroup descriptions G such that

$$\forall \text{ subgroup descriptions } sd: \; sd \notin G \Rightarrow q(DB, sd) \leq q*,$$

where $q* = \min_{sd \in G} q(DB, sd)$.

[2] The notation and definitions used in other papers like [KLJ03, Wro97] sometimes slightly differ from ours, but are (factor-) equivalent. For example, the Gini-Quality is often expressed using the *generality* (i.e. n/N) of the subgroup. Other authors use a notation like $p(\mathit{class}|\, s)$ to denote the probability of a class in a subgroup.

Table 1. Common quality functions for subgroups

Name	Type	Definition
Piatetsky-Shapiro	2	$n(p - p_0)$
Split	N	$n \sum_i (p_i - p_{0i})^2$
Gini	N	$\frac{n}{N-n} \sum_i (p_i - p_{0i})^2$
Pearson's χ^2	N	$n \sum_i \frac{(p_i - p_{0i})^2}{p_{0i}}$

2.2 Optimistic Estimates and Their Use in Subgroup Discovery

Before we present the definition of optimistic estimates, we would like to motivate this concept by taking a look at possible algorithmic approaches to subgroup discovery. Given that the space of candidate subgroup descriptions can be considered as a tree with subgroup descriptions of length 1 at the first level, subgroup descriptions of length 2 at the next level and so on, one obvious approach to subgroup discovery is to perform some kind of search.

Of course, in this approach the size of the search space is exponential in the number of attributes and hence it is desirable to use some kind of pruning strategy. Unfortunately, unlike related tasks like frequent item mining where state-of-the art algorithms like FpGrowth [HPYM04] exploit the property of *monotonicity*, in subgroup discovery this property does not hold: Even if the subgroup description $a_1 = x$ does not have a sufficient quality, it is still necessary to consider its refinements. In fact, even if neither $a_1 = x$ nor $a_2 = y$ are interesting subgroups, $(a_1 = x, a_2 = y)$ might very well be interesting.

However, if we have already found k subgroups and we knew that all refinements s' of a subgroup s had a quality that is worse than that of all k subgroups found so far, we could safely prune that branch. What is needed to do so is an *optimistic estimate* for the refinements s' of s [Wro97]:

Definition 1. *An optimistic estimate $oe(s)$ for a given quality function q is a function that satisfies the following: $\forall$ subgroups $s, s'.\ s' \succ s \implies oe(s) \geq q(s')$.*

3 Tight Optimistic Estimates

In this section, we will present our definition of *tight* optimistic estimates. Thereafter, we will present new optimistic estimates for all quality functions in Table 1.

3.1 A Definition of Tightness with Respect to Probability and Size

The quality functions from Table 1 are all defined in terms of a few characteristics of the subgroup and the dataset, namely

- the distributions over the classes in the subgroup, denoted by $\boldsymbol{p}$;
- the size of the subgroup, denoted by n;
- the distributions over the classes in the dataset, denoted by $\boldsymbol{p_0}$; and
- the size of the dataset, denoted by N.

We will call such quality functions "probability/size quality functions" or "p/n quality functions". Formally, a p/n quality function is a function $q(\boldsymbol{p}, n, \boldsymbol{p_0}, N)$ from $[0,1]^c \times N \times [0,1]^c \times N$ to the reals (here, c is the number of classes). We call $\boldsymbol{p}$ and n the *parameters* of the subgroup and $\boldsymbol{p_0}$ and N the parameters of the overall population. Similarly, we will call optimistic estimates that only make use of these parameters "p/n optimistic estimates". Formally, a p/n optimistic estimate is a function from $[0,1]^c \times N \times [0,1]^c \times N$ to the reals such that

$$\forall \text{ subgroups } s, s'.\, s' \succ s \implies oe(\boldsymbol{p}(s), n(s), \boldsymbol{p_0}, N) \geq q(\boldsymbol{p}(s'), n(s'), \boldsymbol{p_0}, N).$$

Here, $\boldsymbol{p}(s)$ and $n(s)$ denote the class distribution and the size for subgroup s. In general, there are infinitely many optimistic estimates. We are interested in optimistic estimates that are as *conservative* as possible in the following sense:

Definition 2. *Given a quality function q and two optimistic estimates oe_1 and oe_2, we call oe_1 is more* conservative *than oe_2 if* $\forall N, \boldsymbol{p_0}, n, \boldsymbol{p}.\, oe_1(\boldsymbol{p}, n, \boldsymbol{p_0}, N) \leq oe_2(\boldsymbol{p}, n, \boldsymbol{p_0}, N)$.

The more conservative an optimistic estimate, the larger part of the search space can potentially be pruned: if we have already found k subgroups with a minimum quality $minQ$, then we can prune the branch of subgroups below s if and only if $oe(s) < minQ$. We will now formally define the notion of *tight* optimistic estimates, i.e. optimistic estimates that are as conservative as possible:

Definition 3. *An optimistic estimate oe for a quality function q is* tight *if for any population DB and any subgroup description sd there is a subset s' of the extension of sd on DB such that the quality of s' is equal to the optimistic estimate for sd on DB. Formally: the optimistic estimate oe is* tight *iff*

$$\forall\, DB, sd\,.\, \exists n', \boldsymbol{p'}.\, [n' \leq n \wedge n'\boldsymbol{p'} \preceq n\boldsymbol{p} \wedge oe(\boldsymbol{p}, n, \boldsymbol{p_0}, N) = q(\boldsymbol{p'}, n', \boldsymbol{p_0}, N)].$$

Here, $\boldsymbol{p_0}$ and $\boldsymbol{p}$ denote the probability distribution in DB, respectively in the extension of sd on DB, while N and n denote the size of DB respectively of the subgroup extension (actually, $\boldsymbol{p}, n, \boldsymbol{p_0}$ and N are functions of DB and s). $n'\boldsymbol{p'} \preceq n\boldsymbol{p}$ means that for all i, $n'p'_i \leq np_i$, i.e. the number of rows of class i in the subset of sd must be no larger than the number of rows of class i in sd.

Please note that the above definition does only require that there is a *subset* of the extension of sd on DB with quality $oe(\boldsymbol{p}, n, \boldsymbol{p_0}, N)$ – it does not require that there actually is a *subgroup description* with that quality. That is, the definition considers every subset of rows that is consistent with the restrictions provided by the parameters $\boldsymbol{p}$ and n. It is obvious that if an optimistic estimate for q is *tight*, then there is no optimistic estimate for q that is more conservative.

3.2 A Tight Estimate for Piatetsky-Shapiro

We will now apply our definition of tightness to the optimistic estimate published in [Wro97] for the Piatetsky-Shapiro function, and show that it is not tight. Thereafter, we will present a tight optimistic estimate for that quality function. First, we remark that in this two-class case we use p resp. p_0 to refer to the first component of $\boldsymbol{p}$ resp. $\boldsymbol{p_0}$.

Lemma 1. *The optimistic estimate* $n(1-p_0)$ *presented in [Wro97] for the Piatetsky-Shapiro function* $n(p-p_0)$ *is not tight. The optimistic estimate* $oe^*_{ps} := np(1-p_0)$ *is tight.*

Proof. We first show that oe^*_{ps} is tight. Suppose we are given a database DB and an arbitrary subgroup extension s with probability p and size n. The case $p = 0$ is trivial, so let $p > 0$; The subgroup extension s contains np rows of the first class. These rows are a subset of s with size np and a class distribution of $p = 1$; thus, this subset has quality $np(1-p_0)$. This construction did not make any assumptions on DB or s, thus there is always a subset s' with $q(s') = oe^*_{ps}(s)$, and oe^*_{ps} is tight.

Finally, to see that $n(1-p_0)$ is not tight it is sufficient to note that $np(1-p_0) < n(1-p_0)$ for some $n > 0$, $p_0 < 1$ and $p < 1$. □

3.3 Tight Estimates for the Multi-class Quality Functions Split, Gini and Pearson's χ^2

Next, we turn to the multi-class quality functions. First, please note that all multi-class quality functions $q(\boldsymbol{p}, n, \boldsymbol{p_0}, N)$ considered can be reformulated as functions $q(\boldsymbol{m}, \boldsymbol{p_0}, N)$, where $\boldsymbol{m} = (m_1, ..., m_c)$ is a vector whose components are the numbers of rows of the different classes $1, ..., c$. The m_i's can be obtained from $\boldsymbol{p}, n$ by taking the scalar product $n \cdot (p_1, ..., p_c)^T$, while $\boldsymbol{p}$ and n can be obtained from the m_i's by calculating $n = \sum_j m_j$ and $\boldsymbol{p} = \frac{1}{n} \cdot \boldsymbol{m}$. Using this new representation, we can now present a scheme of tight optimistic estimates for the multi-class quality functions in Table 1:

Lemma 2. *The following is a tight optimistic estimate for every multi-class quality function* q *in Table 1 (in fact, for any quality function that is convex in* $\boldsymbol{p}$ *resp.* $\boldsymbol{m}$*):*

$$oe^*_q(p_1, ..., p_c, n, \boldsymbol{p_0}, N) := \max_{m'_1,...,m'_c | m'_i \in \{0, np_i\}} \{q((m'_1, ..., m'_c)^T, \boldsymbol{p_0}, N)\} \quad (1)$$

The above is thus the maximum over the 2^c possible combinations of values for $m'_1, ..., m'_c$, resulting from the constraint that every m'_i can either take the value 0 or np_i. Please note that although not specified in Equation 1, the case $m'_i = 0$ for every i needs not be considered.

Proof. The proof that oe^*_q is an optimistic estimate is based on the fact that all the multi-class quality functions considered in Table 1 are convex in $\boldsymbol{p}$ resp. $\boldsymbol{m}$.

First, we use the fact that by definition a tight optimistic estimate for the refinements of a subgroup s is the maximum over the quality of every possible subset of subgroup s. The subgroup s has np_1 rows of class 1, np_2 rows of class 2, and so on. Thus, every refinement s' of s consists of at most np_i rows of class i. Hence, a tight optimistic estimate for a quality function q can be calculated as follows:

$$\max_{m'_1,...,m'_c | \forall i. m'_i \in N_+ \wedge 0 \leq m'_i \leq np_i} \{q((m'_1, ..., m'_c)^T, \boldsymbol{p_0}, N)\} \quad (2)$$

Please note that unlike in Equation 1, the above considers the maximum over *every* subset of rows of the subgroup s. The above expression is not only an optimistic estimate, but it is tight, which follows directly from our definition of tightness from Section 3.1.

It remains to show that Equations 1 and 2 are equivalent. This can be shown using the fact that every quality function in Table 1 is convex in its parameters $\boldsymbol{m}$. In fact, for every c-dimensional convex function f the maximum over a polyhedron $P = [0, m_1] \times [0, m_2] \times ... \times [0, m_c]$ is an extreme point of P, also called a vertex [Bre96, BV04]. Thus, the maximum over P can be calculated by taking the maximum of the values at every extreme point. The proof is completed by the fact that every quality function in Table 1 is convex, as shown in Appendix A. Please note that although in the Appendix we consider the extension of the quality functions to the real numbers, the extreme points are nevertheless tuples of positive natural numbers. □

Let us now consider the computational complexity for the calculation of the tight optimistic estimate. oe_q^* involves taking the maximum of 2^c values of q, where c is the number of classes. All multi-class quality functions we considered can be reformulated to the form $q = \phi_1 + ... + \phi_c$ (for the Split quality function, the expressions ϕ_i summed up are $n(p_i - p_{0i})^2$, for Gini they are $\frac{n}{N-n}(p_i - p_{0i})^2$ and for Pearson's they are $\frac{n}{p_{0i}}(p_i - p_{0i})^2$). The calculation of such an expression ϕ_i involves only a constant number of subtractions and multiplications, thus the tight optimistic estimates oe_q^* have a computational complexity of $O(c\,2^c)$ primitive (add/multiply) operations. Please note that the computational complexity of oe_q^* does not depend on the size of dataset, but only on the parameters $\boldsymbol{p}, n, \boldsymbol{p_0}$ and N, which have to be calculated anyway to compute the quality of the subgroup.

3.4 A Family of Increasingly Conservative Optimistic Estimates

For large numbers of classes, c, the computational complexity, $O(c\,2^c)$, of the tight optimistic estimate oe_q^* can become problematic, as will be confirmed by the experiments in Section 4.

In this section, we present a scheme of optimistic estimates that are not tight, but faster to calculate. The estimates are increasingly conservative and at the same time increasingly complex to calculate. The idea is not to consider all 2^c combinations as done in oe_q^*, but instead to consider only combinations for d classes at a time. That is, we consider 2^d different combinations for the d selected classes; for the other classes, we only consider the two extreme cases where either *every* $m_i' = m_i$ or every $m_i' = 0$ (for classes i not within the d classes). Finally, the sum over all these maximums is calculated and used as an estimate.[3]

Here is the definition of the scheme oe_q^d of optimistic estimates, where d determines the number of classes considered at a time:

[3] This scheme is a generalization of the optimistic estimate proposed in [GRSW08], which considers just one class at a time (instead of d).

$$oe_q^d(\boldsymbol{p}, n, \boldsymbol{p_0}, N) := \sum_{j=1,d+1,2d+1,\ldots} \left[\max_{m'_j,\ldots,m'_{j+d-1} | m'_j \in \{0, np_j\}} \left(\max \Big\{ \sum_{i=j}^{j+d-1} \phi_i(\boldsymbol{m'_-}, \boldsymbol{p_0}, N), \sum_{i=j}^{j+d-1} \phi_i(\boldsymbol{m'_+}, \boldsymbol{p_0}, N) \Big\} \right) \right] \quad (3)$$

Here, the ϕ_i are summands of the quality functions as discussed in the previous section. The $\boldsymbol{m'_-}$ stands for the vector $(0, 0, ..., m'_j, ..., m'_{j+d-1}, 0, ..., 0)^T$ and $\boldsymbol{m'_+}$ for the vector $(np_0, np_1, ..., np_{j-1}, m'_j, ..., m'_{j+d-1}, np_{j+d}, ..., np_c)^T$; intuitively $\boldsymbol{m'_-}$ stands for the case where the subset of s includes none of the rows of the classes $1, ..., j-1, j+d, ..., c$ of the subgroup s, while $\boldsymbol{m'_+}$ stands for the case where every row of these classes is present. Of course, if c is not a multiple of d, the last summands might involve less than d classes.

Proof. Similar to the proof for oe_q^*, the proof is based on the fact that every quality function considered is convex, as shown in the Appendix. Additionally, we use the fact all quality functions considered can be brought to the form $\phi_1(\boldsymbol{m}, \boldsymbol{p_0}, N) + ... + \phi_c(\boldsymbol{m}, \boldsymbol{p_0}, N)$, with ϕ_i being $(\sum_j m_j)(\frac{m_i}{\sum_j m_j} - p_{0i})^2$ for Split, $\frac{(\sum_j m_j)}{N-(\sum_j m_j)}(\frac{m_i}{\sum_j m_j} - p_{0i})^2$ for Gini and $\frac{(\sum_j m_j)}{p_{0i}}(\frac{m_i}{\sum_j m_j} - p_{0i})^2$ for Pearson's. Now the following holds:

$$\max_{m'_1,\ldots,m'_c | \forall i. m'_i \in N_+ \wedge 0 \leq m'_i \leq np_i} \{\sum_{i=1}^{c} \phi_i(\boldsymbol{m'}, \boldsymbol{p_0}, N)\} =$$

$$\max_{m'_1,\ldots,m'_c | \forall i. m'_i \in \{0, np_i\}} \{\sum_{i=1}^{c} \phi_i(\boldsymbol{m'}, \boldsymbol{p_0}, N)\} \leq$$

$$\sum_{j=1,d+1,2d+1,\ldots,c} \max_{m'_1,\ldots,m'_c | \forall i. m'_i \in \{0, np_i\}} \{ \sum_{i=j}^{j+d-1} \phi_i(\boldsymbol{m'}, \boldsymbol{p_0}, N)\} \leq$$

$$\sum_{j=1,d+1,\ldots,c} \max_{m'_j,\ldots,m'_{j+d-1} | m'_j \in \{0, np_j\}} \left[\max_{m'_1,\ldots m'_{j-1}, m_{j+d},\ldots m'_c | m'_i \in \{0, np_i\}} \{ \sum_{i=j}^{j+d-1} ...\} \right]$$

where ... stands for $\phi_i(\boldsymbol{m'}, \boldsymbol{p_0}, N)$. Now we make use of the fact that ϕ_i is not only convex in m_i and $m_j, j \neq i$, but also in the sum of m_j's with $j \neq i$. That is, for every set of indexes $J = \{j_1, ...j_n\}$ not including i, ϕ_i is convex in $\sum_{k \in J} m_k$. This follows from the fact that in all definitions of ϕ_i (i.e. for the definition for Split, Gini and χ^2) the m_j with $j \neq i$ only occur in the sum $\sum_j m_j$. Thus, for any set of indexes J that does not include i, ϕ_i could be considered as a function of the new parameter $(\sum_{k \in J} m_k)$ and the remaining $m_{k'}$, i.e. those with index $k' \notin J$. The resulting function is of the same form as ϕ_i (except that it takes less parameters) and thus is convex in $\sum_{k \in J} m_k$.

In particular, any ϕ_i with $j \leq i \leq j+d-1$ is convex in $\sum_{k\in\{0,\ldots,j-1,j+d,\ldots c\}} m_k$. Therefore, it is sufficient to consider the case where the sum is minimal or maximal, that is when either all these m'_k's are zero or all have value np_k. So the above is bounded by

$$\sum_{j=1,d+1,\ldots,c} \left[\max_{m'_j,\ldots,m'_{j+d-1}|m'_j\in\{0,np_j\}} \left(\max_{(\forall k.\ m'_k=0),(\forall k.\ m'_k=np_k)} \sum_{i=j}^{j+d-1} \phi_i(\boldsymbol{m}', \boldsymbol{p_0}, N) \right) \right]$$

This is equivalent to oe_q^d and the proof is completed. □

Some Considerations on the new Optimistic Estimates. We will now consider some properties of the optimistic estimates oe_q^d. As before, we use c to denote the number of classes. The computation of oe_q^d involves the evaluation of $O(\frac{c}{d}2^d d) = O(c2^d)$ different expressions ϕ_i, meaning that the higher the number d of classes considered at a time, the higher the computational cost. We remark that in an implementation of the function oe_q^d, only those classes where $m_i > 0$ have to be considered, meaning that $O(c2^d)$ is only the worst case.

Lemma 3. *oe_q^{2d} is at least as conservative as oe_q^d. oe_q^d is tight if either $d \geq c$, or $c = 2$ and $d \geq 1$.*

Proof. It is easy to see that if $c \leq d$, then $oe_q^d = oe_q^*$, because the number of classes d in oe_q^d for which every combination is considered is exactly c, as in oe_q^*. oe_q^{2d} is at least as conservative as oe_q^d because oe_q^{2d} considers every combination of classes considered by oe_q^d (and some more combinations).

In the special case $c = 2$, oe_q^1 is tight because the set of indexes $\{0, ..., j-1, j+d, ...c\}$ considered in the second sum of Equation 3 actually involves only one class index, and hence effectively every combination of the two classes is considered, just as in oe_q^*. To show that oe_q^d is not tight otherwise, it is sufficient to have one example. The experiments in Section 4 have plenty of them, and the next paragraph also presents one. Here is another one: consider $\boldsymbol{p_0} = (0.1, 0.45, 0.45)$ (for $c = 3$) respectively $\boldsymbol{p_0} = (0.1, 0.3, \frac{0.3}{c-3}, ..., \frac{0.3}{c-3}, 0.3)$ (for $c > 3$). Furthermore, consider a subgroup s with $\boldsymbol{m} = (10, 10, 0, 0, ...0, 10)$. It is easy to verify that of all subsets of $\boldsymbol{m}$, $\boldsymbol{m'} = (10, 0, 0, ...0, 0)$ has the highest quality. However, the last summand of oe_q^d would consider the values 10 and 0 for m'_c only in combination with *either* both $m'_1 = 10$ and $m'_2 = 10$ *or* both $m'_1 = 0$ and $m'_2 = 0$. That is, it would not consider the actual maximizing combination $(10, 0, 0, ...0, 0)$, but instead will provide an overoptimistic estimate. □

An Example. To illustrate the effect of different optimistic estimates, let us reconsider the example from Figure 1. In particular, let us consider the subgroup description $\{Region = Urban\}$. The corresponding subgroup consists of the first three rows of the dataset and has a probability distribution of $p_{High} = 1/3$, $p_{Med} = 2/3$, and $p_{Low} = 0$.

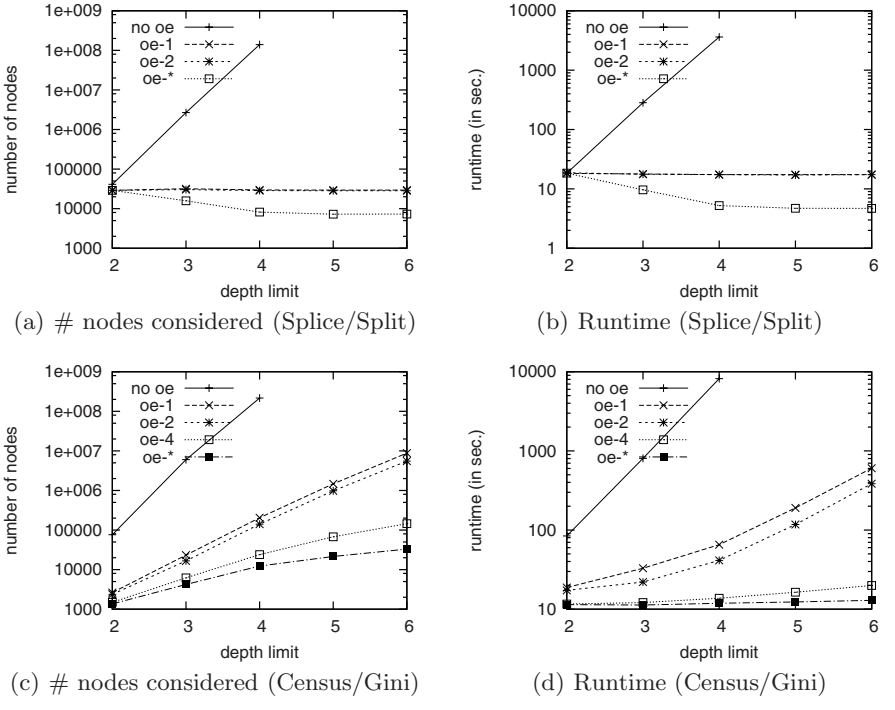
For the Split quality function, we get a tight optimistic estimate of 0.176 for the quality of the refinements of $\{Region = Urban\}$. Using the suboptimal

estimate oe_q^1, we only get the estimate 0.255. While both estimates can be used by a subgroup discovery algorithm, if the minimal required quality is 0.195 only the tight estimate of 0.176 would allow to prune all subgroups description below $\{Region = Urban\}$ (In fact, 0.195 is exactly the minimal required quality at that point, if the algorithm sketched in Section 4 is used).

4 Experimental Evaluation

Sketch of a Subgroup Discovery Algorithm. To evaluate the impact of different optimistic estimates, we used a branch and bound depth-first-search (DFS) algorithm similar to OPUS^O [Web95] (without optimistic reordering). The optimistic estimates are used to prune as large a part of the search space as possible, and furthermore determine the order in which the nodes are expanded during DFS. Our implementation makes use of FP-Trees [HPY00] to speedup the calculation of the parameters $\boldsymbol{p}$ and n of a subgroup, as first proposed in [AP06]. The overall algorithm, called DpSubgroup, is described in more detail in [GRSW08].

Datasets and Results. We evaluated the impact of the different optimistic estimates on five datasets: four datasets from the UCI Machine Learning Repository



(a) # nodes considered (Splice/Split) (b) Runtime (Splice/Split)

(c) # nodes considered (Census/Gini) (d) Runtime (Census/Gini)

Fig. 2. Results for the Splice and Census dataset (with 3 resp. 5 classes) using the Split resp. the Gini quality function. The curves show the number of nodes considered (left images) resp. the runtime (right images) for different optimistic estimates.

[AN07] and one real-world dataset. In particular, we used the "Mushroom" dataset with 8124 rows, 22 attributes and 2 classes; the "Soybean" dataset with 683 rows, 35 attributes and 19 classes; the "splice" dataset with 3190 rows, 62 attributes and 3 classes; a sample of 30.000 rows of the UsCensus1990 database with 68 attributes and 5 classes (we used "dIncome1" as class attribute); and finally a prescription dataset with 29 attributes, 60488 rows and 11 classes from the *iWebCare* project (`http://iwebcare.iisa-innov.com/`). In all experiments, we searched for the top 100 groups on an Intel Core 2 Duo T7500 with 2 GB of RAM under Windows XP.

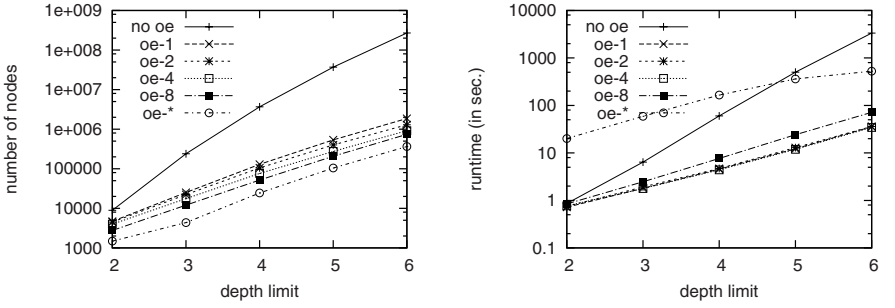
Figure 2 shows the performance results on the datasets Census and Splice, using different quality functions. The horizontal axis shows the depth limits for the subgroup discovery, that is a maximum length of the subgroup descriptions considered. The different curves show the results using different optimistic estimates: "oe-d" stands for subgroup discovery with oe_q^d, "oe-*" shows the results using the tight optimistic estimate oe_q^*, and finally "no oe" shows the performance without any optimistic estimate pruning. It is worthwhile to note that "no oe" essentially corresponds to the algorithm SD-Map (which also makes use of FP-Trees but does not use optimistic estimate pruning), because this algorithms has been shown to outperform all other exhaustive algorithms like Apriori-SD [KLJ03] by an order of magnitude [AP06].

The figure shows both the number of nodes explored during the subgroup discovery and the overall runtime, using a logarithmic scale. As expected, the number of nodes considered depends on the optimistic estimate used: The higher the degree d in oe_q^d, the less nodes are considered. The use of the tight optimistic estimate oe_q^* results in a minimal number of considered nodes.[4] Similar to the number of nodes, the runtime is affected by the optimistic estimate used. Although the performance ratio does not exactly correspond to the node ratio, the ordering of the optimistic estimates is the same. The performance gain using pruning can become as large as an order of magnitude and more.

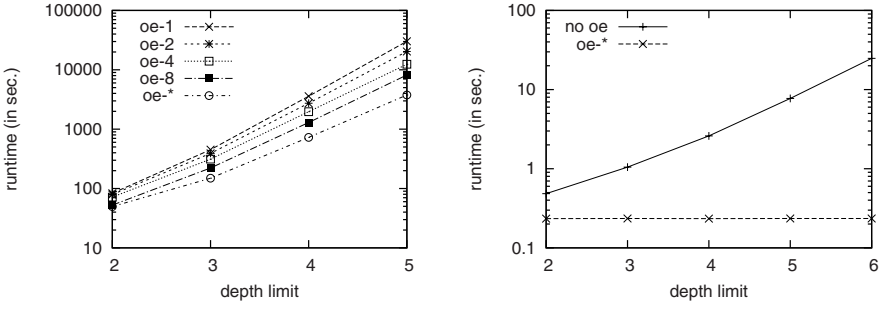
Figure 3 shows the results for the Soybean and the Mushroom dataset. We first consider the results for the Soybean dataset (subfigures (a)-(c)): Again, the higher the degree d of the optimistic estimate oe_q^d, the less nodes are considered, with oe_q^* being optimal (Figure 3(a)). Regarding the runtime, however, the situation is different: Figure 3(b) shows that the runtime is minimal when a pruning level d is between 1 and 4. The reason is that although the more conservative optimistic estimates reduce the number of nodes considered, their calculation is more expensive. This effect becomes apparent in the Soybean dataset because it has much more classes than the earlier datasets.

The above experiment shows that for datasets with a large number of classes, it can be appropriate to use a non-tight optimistic estimate. However, the best pruning level depends on the *ratio* of the costs for the calculation of the optimistic

[4] It is interesting to note that the number of nodes and the runtime sometimes *decreases* when the depth limit increases. The reason is that the algorithm quickly finds subgroups with a very high quality at a higher level, which allows to prune a larger part of the search space than possible if only shorter subgroups were considered.



(a) # nodes considered (Soybean/Gini) (b) Runtime using FpTree (Soybean/Gini)



(c) Runtime using *persistent* FpTree (d) Runtime (Mushroom/P.-S.)

Fig. 3. Number of nodes (a) and runtime (b, c) on the Soybean dataset (using the Gini quality function), and runtime on the mushroom dataset (d) (using the Piatetstky-Shapiro quality function)

estimate, and the cost for the calculation of the parameters of a subgroup. So far we used FP-Trees, which allowed a very fast calculation of n and $\boldsymbol{p}$. However, if the dataset is very large it might not be possible to keep the FP-Tree in main memory. To see how the situation changes if the calculation of the parameters n and p becomes more expensive, we have run some experiments using a *persistent* FP-Tree, i.e. an implementation where the FP-Trees are stored on disk [HPYM04]. Our prototypical implementation is based on the object database db4o [PEHH06]. Figure 3(c) shows the resulting runtime: In this setting, the use of conservative optimistic estimates pays off again, with oe_q^* resulting in the fastest calculation (the number of nodes is, of course, unaffected).

Finally, Figure 3(d) shows the runtime for the Mushroom dataset, using the Piatetstky-Shapiro quality. As in the multi-class case, the use of optimistic estimates results in a significant speedup in this two-class example. The results from the prescriptions dataset do not significantly differ, so we merely used them in the summarizing table in the next paragraph.

Summary. The above results show that the optimistic estimates presented in this paper have a significant impact on the performance of the subgroup discovery. The use of the (fastest to calculate) optimistic estimates oe_q^1 and oe_{ps}^* never

slows down the execution but instead results in a tremendous speedup. The performance gain reaches an order of magnitude at relative small depth bounds (about 2 to 4) and gets even larger when the depth limit continues to increase.

The use of more conservative optimistic estimates allow to further speedup the subgroup discovery. The following table summarizes the performance gain over oe_q^1 achieved using the optimistic estimates oe_q^d and oe_q^* (with $d > 1$). For different values of d (i.e. 2, 3, 4 and ∞), it compared the performance with that achieved using oe_q^1, aggregated over different quality functions and datasets. In particular, it shows the relative runtime (the runtime using the more conservative optimistic estimate, divided by the runtime using oe_q^1) in the best experiment (Minimal), the worst experiment (Maximal) and on average. The table is based on a total of 248 experiments.

	oe_q^2	oe_q^3	oe_q^4	oe_q^*
Minimal relative runtime compared to oe_q^1	62%	21%	3%	1%
Average relative runtime compared to oe_q^1	93%	69%	63%	847%
Maximal relative runtime compared to oe_q^1	113%	118%	135%	3640%

The table shows that the larger d, the more the runtime can decrease (due to the stronger pruning) - but it can also increase (due to the computation time of the optimistic estimate). Overall, the use of oe_q^4 (which is tight if the number of classes is ≤ 4) is a safe choice in most situations. While in the worst example, it was slower by 35% than oe_q^1, on average it was faster, taking only 63% of oe_q^1's runtime; In the best example, it even reduced the runtime to 3% of that of oe_q^1.

5 Summary and Discussion

In this paper, we have pursued the investigation of optimistic estimates for fast subgroup discovery, started in [Wro97]. In particular, we considered and formalized the concept of *tight* optimistic estimates. We have shown that the optimistic estimate proposed in [Wro97] is not tight, and have presented new tight optimistic estimates, including tight optimistic estimates for several multi-class quality functions.

While the use of tight optimistic estimates minimizes the number of subgroups considered, their calculation sometimes becomes quite time consuming. To cope with this difficulty, we have presented a family of increasingly efficient approximations of the tight optimistic estimates. While these estimates are (in general) not tight, they allow to trade more conservative estimates for faster computation and thereby provide a mean to select an optimistic estimate with the right ratio of conservative-ness and computational cost.

The results are interesting both from a theoretic and a practical point of view. On the theoretical side, the notions of conservative and tight optimistic estimates allow to compare optimistic estimates. On the practical side, our experiments show that the use of the new optimistic estimates oe_{ps}^*, oe_q^d and oe_q^* result in a significant speedup compared to current state-of-the-art algorithms like SD-Map [AP06]. While for problems with a relatively small number of classes oe_q^* is the estimate of choice, for datasets with a larger number of classes (more than 6 or

so) the optimal choice depends on more factors. Overall, the use of oe_q^4 is a safe choice in most situations.

The idea to perform pruning based on an optimistic evaluation of the search space below a node was already investigated before the introduction of optimistic estimate into subgroup discovery. In particular, Webb [Web95] considers optimistic pruning for rule learning tasks in his OPUS search algorithm. The concept is also applied in other pattern-mining tasks involving non-monotonic objective functions, like tiling databases [GGM04]. Of course, the objective function considered in tasks like tiling are different than in subgroup discovery.

In future work, we plan to investigate optimistic estimates for other quality function, and to extend the concept of tight optimistic estimates to numeric target attributes. We would also like to investigate whether the optimistic estimates oe_q^d can be improved by some kind of heuristic grouping of the classes, with the idea not just to build arbitrary groups of d attributes. It would also be interesting to combine the optimistic estimates presented in this paper with other approaches to subgroup discovery, in particular with sampling-based approaches [SW00]. Altogether, we believe that the definition of tight optimistic estimates and the new optimistic estimates presented in this paper can be a valuable instrument in a wide range of subgroup discovery algorithms.

Acknowledgments. The financial support of the European Commission under the Project *iWebCare* (IST-2005-028055) is gratefully acknowledged.

References

[ABP06] Atzmüller, M., Baumeister, J., Puppe, F.: Introspective subgroup analysis for interactive knowledge refinement. In: Sutcliffe, G., Goebel, R. (eds.) FLAIRS Conference, pp. 402–407. AAAI Press, Menlo Park (2006)

[AN07] Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)

[AP06] Atzmüller, M., Puppe, F.: SD-map - a fast algorithm for exhaustive subgroup discovery. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 6–17. Springer, Heidelberg (2006)

[BFOS84] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. (eds.): Classification and Regression Trees. Wadsworth (1984)

[Bre96] Breiman, L.: Technical note: Some properties of splitting criteria. Machine Learning 24(1), 41–47 (1996)

[BV04] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

[GGM04] Geerts, F., Goethals, B., Mielikäinen, T.: Tiling databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 278–289. Springer, Heidelberg (2004)

[GRSW08] Grosskreutz, H., Rüping, S., Shaabani, N., Wrobel, S.: Optimistic estimate pruning strategies for fast exhaustive subgroup discovery. Technical report, Fraunhofer Institute IAIS (2008), `http://publica.fraunhofer.de/eprints/urn:nbn:de:0011-n-723406.pdf`

[HPY00] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) SIGMOD Conference, pp. 1–12. ACM, New York (2000)

[HPYM04] Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8(1), 53–87 (2004)

[KLGK07] Kralj, P., Lavrac, N., Gamberger, D., Krstacic, A.: Contrast set mining through subgroup discovery applied to brain ischaemina data. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 579–586. Springer, Heidelberg (2007)

[KLJ03] Kavsek, B., Lavrac, N., Jovanoski, V.: Apriori-sd: Adapting association rule learning to subgroup discovery. In: R. Berthold, M., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.) IDA 2003. LNCS, vol. 2810, pp. 230–241. Springer, Heidelberg (2003)

[Klö96] Klösgen, W.: Explora: A multipattern and multistrategy discovery assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271 (1996)

[Klö02] Klösgen, W.: Subgroup Discovery. In: Handbook of Data Mining and Knowledge. Oxford University Press, Oxford (2002)

[KM02] Klösgen, W., May, M.: Spatial subgroup mining integrated in an object-relational spatial database. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 275–286. Springer, Heidelberg (2002)

[LCGF04] Lavrac, N., Cestnik, B., Gamberger, D., Flach, P.A.: Decision support through subgroup discovery: Three case studies and the lessons learned. Machine Learning 57(1-2), 115–143 (2004)

[LKFT04] Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup discovery with cn2-sd. Journal of Machine Learning Research 5, 153–188 (2004)

[PEHH06] Paterson, J., Edlich, S., Hörning, H., Hörning, R.: The Definitive Guide to db4o. Apress, Berkely (2006)

[SW00] Scheffer, T., Wrobel, S.: A sequential sampling algorithm for a general class of utility criteria, pp. 330–334. ACM Press, New York (2000)

[Web95] Webb, G.I.: Opus: An efficient admissible algorithm for unordered search. J. Artif. Intell. Res (JAIR) 3, 431–465 (1995)

[Wro97] Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)

A Proof of Convexity of the Multi-class Quality Functions

A.1 Split and Pearson's χ^2

Both the Split and Pearson's χ^2 quality functions are nonnegative weighted sums. The nonnegative weighted sum of convex functions is convex ([BV04]), hence it is sufficient to show that the summands are convex. The summand of both quality functions can be brought to the following form

$$\phi_i = c_i(\sum_j n_j)(\frac{n_i}{\sum_j n_j} - p_{0i})^2$$

where in the case if Split $c_i = 1$ and in the case of Pearson's $c_i = \frac{1}{p_{0i}}$.

We only need to consider the case where $c_i = 1$, because the c_i's are merely positive weights. The ϕ_i's are twice differentiable, thus it is sufficient to show that the *Hessian* or second derivative is positive semidefinite [BV04]. We only consider the first summand ϕ_1, as the other cases are analog. It is somewhat laborious but straightforward to verify that

$$\nabla^2\phi_1 = \frac{2}{(\sum_j n_j)^3}\begin{bmatrix} (\sum_{j\neq 1} n_j)^2 & -n_1(\sum_{j\neq 1} n_j) & ... & -n_1(\sum_{j\neq 1} n_j) \\ -n_1(\sum_{j\neq 1} n_j) & (n_1^2) & ... & (n_1^2) \\ ... & ... & ... & ... \\ -n_1(\sum_{j\neq 1} n_j) & (n_1^2) & ... & (n_1^2) \end{bmatrix}$$

The above matrix can be brought to the following form

$$GG^T$$

with $G = (\sum_{j\neq 1} n_j, -n_1, ..., -n_1)^T$, that is there is a Cholesky decomposition and hence the matrix is positive-definite [BV04].

A.2 Gini

The fact that the Gini quality function is convex can be derived from previous work in the area of decision tree construction. This comes from the fact that the Gini quality functions is based on the *Gini index*, used as a splitting criterion in the construction of decision trees [Bre96, BFOS84]. The Gini index is defined as $G(p) = \sum_j p_j(1 - p_j)$ and measures the "impurity" of a distribution p. The gain in impurity resulting from a split is defined as $\Theta(s) = G(\boldsymbol{p_0}) - P_L G(\boldsymbol{p_l}) - P_R G(\boldsymbol{p_r})$ and was used as a *goodness of a split* measure. Here, $\boldsymbol{p_0}$ denotes the distribution over the classes in the overall population, $\boldsymbol{p_l}$ and $\boldsymbol{p_r}$ denote the distribution in the left and the right subpopulation resulting from a split, P_L denotes the proportion of the population send to the left by the split and $P_R = 1 - P_L$ denotes the proportion send to the right.

This goodness of split was turned into the Gini quality function [Klö96]:

$$G(\boldsymbol{p_0}) - g * G(\boldsymbol{p}) - (1 - g) * G(\boldsymbol{p^*})$$

where $\boldsymbol{p_0}$ is (as before) the class distribution in the overall population, $\boldsymbol{p}$ the class distribution in the subgroup, $\boldsymbol{p^*}$ the probability distribution in the remainder, i.e. in the examples from the overall population not in the subgroup, and finally g the generality of the subgroup. It is easy to verify that the components of p^* are defined by $p_i^* = \frac{Np_{0i} - Ngp_i}{N(1-g)} = \frac{p_{0i} - gp_i}{1-g}$. By inserting this definition of p^* we obtain the following, more familiar-looking definition of the Gini quality,

$$\frac{g}{1-g}\sum_j (p_i - p_{0i})^2 = \frac{n}{N-n}\sum_j (p_i - p_{0i})^2$$

[Bre96] shows that $g * G(\boldsymbol{p}) + (1 - g) * G(\boldsymbol{p^*})$ is concave in the proportion of the probabilities that are sent to the left (which he calls α). We remark that Breiman

uses the term convex in the meaning of "convex downward"; We use the term convex in the opposite sense, as [BV04] (Breimans notation also differs from ours in other respects). Now our vector $\boldsymbol{m}$ can be obtained from α by an affine mapping, namely $m_i = Np_{0i}\,\alpha_i$, which implies that $g * G(\boldsymbol{p}) + (1-g) * G(\boldsymbol{p^*})$ is also concave in $\boldsymbol{m}$, because affine mappings preserves concavity. Hence the Gini quality function is convex in $\boldsymbol{m}$.

Watch, Listen & Learn: Co-training on Captioned Images and Videos

Sonal Gupta, Joohyun Kim, Kristen Grauman, and Raymond Mooney

Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, Texas 78712-0233
U.S.A
{sonaluta,scimitar,grauman,mooney}@cs.utexas.edu

Abstract. Recognizing visual scenes and activities is challenging: often visual cues alone are ambiguous, and it is expensive to obtain manually labeled examples from which to learn. To cope with these constraints, we propose to leverage the text that often accompanies visual data to learn robust models of scenes and actions from partially labeled collections. Our approach uses co-training, a semi-supervised learning method that accommodates multi-modal views of data. To classify images, our method learns from captioned images of natural scenes; and to recognize human actions, it learns from videos of athletic events with commentary. We show that by exploiting both multi-modal representations and unlabeled data our approach learns more accurate image and video classifiers than standard baseline algorithms.

1 Introduction

Systems able to automatically annotate and index visual content will be crucial to managing the world's ever-growing stores of digital images and videos. However, learning to recognize objects and actions based on visual cues alone remains quite difficult, due to factors ranging from unpredictable illumination to the sheer variety in appearance exhibited by instances of the same class. Furthermore, accurate results often depend on access to substantial labeled data, which in practice can be cumbersome to obtain in adequate quantities.

We propose to facilitate the learning process in this domain by integrating both visual and linguistic information, as well as unlabeled multi-modal data. In particular, we consider the tasks of recognizing categories in natural scenes from images with caption text, and recognizing human actions in sports videos that are accompanied by an announcer's commentary. Both are interesting data sources given their ready availability, but are nonetheless challenging due to the loose association between the dual cues as well as the frequent ambiguity of either cue alone. We design an approach using co-training [6] that takes local appearance and spatio-temporal descriptors together with text-based features to learn the categories from a partially labeled collection of examples.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 457–472, 2008.

The majority of state-of-the-art systems for image and video classification use unimodal data – either visual or textual features alone [32,12,5,35,20,14,17]. Given the natural occurrence of both feature types together, researchers have only recently begun to explore ways to learn from multi-modal image and language data. Previous work has focused on learning the association between visual and textual information [2,13,11], using supervised methods to improve text-based video retrieval [15], improving audio-visual human-computer interfaces [8], and designing unsupervised methods to cluster images [3] or strengthen image features [30]. In contrast, we consider learning to classify images and videos from labeled and unlabeled multi-modal examples, demonstrating that our approach can improve the classification of novel instances by exploiting both cues—or even the visual data alone. While co-training has previously been applied to learn from two textual views [6] or two visual views [31,7], we present comprehensive results on using visual and linguistic information as separate views, with the idea that these distinct cues will complement each another well during training.

Our main contribution is a semi-supervised approach to recognizing scenes and human actions from captioned images or commentated videos. We show that by exploiting multi-modal data and unlabeled examples, our approach improves accuracy on classification tasks relative to both unimodal and early/late fusion baselines. In addition, it yields significantly better models than alternative semi-supervised methods when only a limited amount of labeled data is available.

The remainder of the paper is organized as follows: in Section 2 we discuss related work in more detail. In Section 3 we describe our approach for extracting visual and textual features, and provide background on building a co-training classifier. In Section 5.1 we present results for learning from captioned images, while in Section 5.2 we present results for videos with commentary, and in Sections 6 and 7 we suggest future directions and present our conclusions.

2 Related Work

In previous work using captioned images, Barnard *et al.* [2] and Duygulu *et al.* [10] generate models to annotate image regions with words. Bekkerman and Jeon [3] exploit multi-modal information to cluster images with captions using an unsupervised learning technique. Quattoni *et al.* [30] describe a method for learning representations from large quantities of unlabeled images that have associated captions to improve learning in future image classification problems with no associated captions.

Many researchers have worked on activity recognition in videos using only visual cues [32,12,35,20,5]. Everingham *et al.* [13] incorporate visual information (facial and clothing matching), closed-captioned text, and movie scripts to automatically annotate faces with names in a video. They utilize textual information only for finding names of actors who are speaking at a particular time. Nitta *et al.* [28] annotate sports video by associating text segments with the image segments. Their approach is based on previous knowledge of the game and the key phrases generally used in its commentary. Fleischman and Roy [15] use text

commentary and motion description in baseball video games to retrieve relevant video clips given a textual query. Duygulu and Hauptmann [11] associate news videos with words and improve video retrieval performance. These papers focus on video retrieval rather than classification. Our results provide a novel way to incorporate text information when learning about visual human activity. Wang *et al.* [34] use co-training to combine visual and textual 'concepts' to categorize TV ads. They retrieved text from videos using OCR and used external sources to expand the textual features. Our paper focuses on using visual and textual features from explicitly captioned images and videos without exploiting external sources.

Co-training has previously been shown to be useful for various applications [21,8,31]. Levin *et al.* [23] use co-training to improve visual detectors by training two disparate classifiers. Cheng and Wang [7] suggest a new SVM algorithm called Co-SVM that uses a co-training approach and achieved better results than a normal SVM on classifying images using color and texture as separate views, and Nigam *et al.* [27] compares the effectiveness of co-training with semi-supervised EM.

However, none of the prior work has explored using low-level visual cues and text captions as two views for co-training. We present the first results showing how to learn about human activities based on both visual cues and spoken commentary, and provide a thorough evaluation of our co-training approach relative to several other relevant methods. Since image and video classification is a difficult problem and many videos and images have associated text, we believe that our co-training approach is a novel contribution to two important practical applications.

3 Approach

The main idea of our approach is to use image or video content together with its textual annotation (captions, commentary) to learn scene and action categories. To design such a system, the main components we must define are the feature representations for linguistic and static or dynamic visual cues, and the learning procedure. In this section we describe each of these elements in turn.

3.1 Visual Features

Static Image Features. To describe a captioned photograph, we want to capture the overall texture and color distributions in local regions. Following [3], we compute region-based features as follows. Each image is broken into a 4-by-6 grid of uniformly sized cells. For each region, we compute texture features using Gabor filters with three scales and four orientations, and also record the mean, standard deviation, and skewness of the per-channel RGB and Lab color pixel values. The resulting 30-dimensional feature vectors for each of the 24 regions of all images are then clustered using k-Means in order to define the prototypical region responses. Each region of each image is then assigned one of k discrete values based on the cluster centroid closest to its 30-dimensional image feature vector.

The final "bag of visual words" representing an image consists of a vector of k values, where the i'th element represents the number of regions in the image that belong to the i'th cluster. While other descriptors are certainly possible (e.g., using scale and affine invariant interest point detectors [25]), we chose these features based on their demonstrated suitability for the image-caption dataset provided in [3], which we also use in our experiments.

Motion Descriptors from Videos. To represent video clips, we use features that describe both salient spatial changes and interesting movements. In order to capture non-constant movements that are interesting both spatially and temporally, we use the spatio-temporal motion descriptors developed by Laptev [22]. We chose the spatio-temporal interest point approach over a dense optical flow-based approach in order to provide a scale-invariant, compact representation of activity in the scene.

To detect spatio-temporal events, Laptev builds on Harris and Forstner's interest point operators [18,16] and detects local structures where the image values have significant local variation in both space and time. They estimate the spatio-temporal extent of the detected events by maximizing a normalized spatio-temporal Laplacian operator over both spatial and temporal scales. Specifically, the extended spatio-temporal "cornerness" H at a given point is computed as

$$\mu = g(.; \sigma_i^2, \tau_i^2) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix} \tag{1}$$

$$H = \det(\mu) - k\ trace^3(\mu), \tag{2}$$

where '$*$' represents convolution, $g(.; \sigma_i^2, \tau_i^2)$ is a 3D Gaussian smoothing kernel with a spatial scale σ and a temporal scale τ, and L_x, L_y, L_z, and L_t are the gradient functions along the x, y, z, and t directions, respectively. In (1), μ represents a second order spatio-temporal matrix. The points that have a large value of H are selected as interest points.

At each interest point, we extract a HOG (Histograms of Oriented Gradients) feature [9] computed on the 3D video space-time volume. The patch is partitioned into a grid with 3x3x2 spatio-temporal blocks, and four-bin HOG descriptors are then computed for all blocks and concatenated into a 72-element descriptor. The motion descriptors from all the video clips in the training pool are then clustered to form a vocabulary. Finally, a video clip is represented as a histogram over this vocabulary, just as a static image's features are summarized by a histogram of prototypical region descriptors.

3.2 Textual Features

The text features for the images or videos consist of a standard "bag of words" representation of the captions or transcribed video commentary, respectively. We pre-processed the captions to remove stop words and stemmed the remaining words using the Porter stemmer [1]. The frequency of the resulting word stems comprised the final textual features.

Table 1. Co-training Algorithm

- **Inputs:** A set of labeled and unlabeled examples, each represented by two sets of features, one for each view.
- **Algorithm:** Train a classifier for each view using the labeled data with just the features for that view.
- Loop until there are no more unused unlabeled instances:
  1. Compute predictions and confidences of both classifiers for all of the unlabeled instances.
  2. For each view, choose the m unlabeled instances for which its classifier has the highest confidence. For each such instance, if the confidence value is less than the threshold for this view, then ignore the instance and stop labeling instances with this view, else label the instance and add it to the supervised training set.
  3. Retrain the classifiers for both views using the augmented labeled data.
- **Outputs:** Two classifiers whose predictions can be combined to classify new test instances. A test instance is labeled with the category predicted by the classifier with the highest confidence.

3.3 Building the Classifier Using Co-training

Blum and Mitchell introduced co-training, a semi-supervised learning algorithm that requires two distinct "views" of the training data [6]. It assumes that each example is described using two different feature sets that provide different, complementary information about the instance. Ideally, the two views are *conditionally independent* (i.e., the two feature sets of each instance are conditionally independent given the class) and each view is *sufficient* (i.e., the class of an instance can be accurately predicted from each view alone). Co-training first learns a separate classifier for each view using any labeled examples. The most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data.

Co-training was initially used to classify web-pages using the text on the page as one view and the anchor text of hyperlinks on other pages that point to the page as the other view. In this work, we use the extracted visual and textual features as the two views for co-training classifiers to detect scenes and actions.

We followed the basic algorithm suggested by [6] with one additional constraint: an unlabeled example is only labeled if a pre-specified confidence threshold for that view is exceeded. The algorithm is outlined in Table 1. In each iteration, it finds the m most confidently labeled unlabeled examples for each view. If such instances pass the threshold test, they are added to the supervised training set with the predicted label and both classifiers are retrained. The entire process continues until there are no more unlabeled instances.

4 Experimental Design

4.1 Baselines

In order to evaluate the relative strength of co-training with multi-modal data, we compare co-training with several other supervised and semi-supervised techniques that are reviewed in this section.

Early and Late Fusion. Besides co-training, multi-modal fusion methods are an alternative way to utilize both sets of features. The visual and linguistic information can be 'fused' in two ways: early and late fusion [33]. In early fusion, unimodal features are extracted and then combined into a single representation. In our case, we extract visual and textual features and concatenate them into a single vector. In contrast, late fusion learns separate unimodal classifiers directly from unimodal features and then combines their results when labeling test instances. In particular, we combine the two unimodal classifiers by using the decision of the classifier with the highest confidence.

Semi-supervised EM and Transductive SVMs. Semi-supervised Expectation Maximization (Semi-Sup EM) and transductive Support Vector Machines (TSVM) are two other standard approaches to semi-supervised learning. These methods can be applied to either of the two views individually, or employ both feature sets using early or late fusion.

Although typically used for unsupervised learning, Expectation Maximization (EM) can also be used in a semi-supervised setting [26]. First, Semi-Sup EM learns an initial probabilistic classifier from the labeled training data. Next, it performs EM iterations until convergence. In the E step, it uses the currently trained classifier to probabilistically label the unlabeled training examples. In the M step, it retrains the classifier on the union of the labeled data and the probabilistically labeled unsupervised examples. Semi-sup EM has typically been applied using Naive Bayes as its probabilistic classifier. For text learning, the multinomial version of Naive Bayes [24] is typically used [26]; however, for our data we found that a standard multivariate model using Gaussian distributions for continuous features gave better results. Specifically, we used the NaiveBayesSimple classifier in Weka [36].

Transductive SVMs [19] find the labeling of the test examples that results in the maximum-margin hyperplane that separates the positive and negative examples of *both* the training and the test data. This is achieved by including variables in the SVM's objective function representing the predicted labels of the unlabeled test examples. Although TSVMs were originally designed to improve performance on the *test* data by utilizing its availability during training, they can also be directly used in a semi-supervised setting [4] where unlabeled data is available during training that comes from the same distribution as the test data but is not the actual data on which the classifier is eventually to be tested. In our experiments we evaluate the strength of our co-training approach relative to these other semi-supervised methods.

4.2 Methodology

For co-training, we use a Support Vector Machine (SVM) as the base classifier for both image and text views. We compare co-training with other supervised and semi-supervised methods, and use the Weka [36] implementation of sequential minimal optimization (SMO) [29] for SVMs (except for TSVMs as described below). SMO is set to use an RBF kernel (γ=0.01) and a logistic model to produce proper output probabilities; otherwise, default parameters are used throughout. We use a batch size of $m = 5$ for co-training. For co-training on static images, we use a confidence threshold of 0.65 for the image view and 0.98 for the text view (determined empirically through cross-validation). For video classification (where there are more classes) we use a threshold of 0.6 for the video view and 0.9 for the text view.

We evaluate all algorithms using ten iterations of ten-fold cross validation to get smoother and more reliable results. For co-training and the other semi-supervised algorithms, the test set is disjoint from both the labeled *and* unlabeled training data.

To evaluate accuracy as the amount of labeled data increases, we generate learning curves where at each point some fraction of the training data is labeled and the remainder is used as unlabeled training data. Thus, for the last point on the curve, all of the training data is labeled. With this methodology, we expect to see an advantage for semi-supervised learning early in the learning curve when there is little labeled data and significant unlabeled data. Once all of the data is labeled, we expect the predictive accuracies of semi-supervised learning and supervised learning to converge.

5 Results

This section presents our experimental results on image and video classification. Some part of our datasets and full results are available on the web at `http://www.cs.utexas.edu/users/ml/co-training`.

5.1 Learning to Categorize Captioned Images

In this section we provide results on classifying captioned static images.

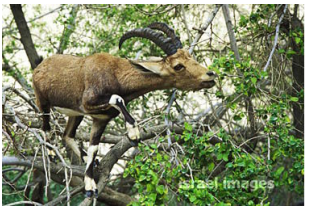
Dataset. Our image data is taken from the Israel dataset[1] introduced in [3], which consists of images with short text captions. In order to evaluate the co-training approach, we used two classes from this data, Desert and Trees. These two classes were selected since they satisfy the sufficiency assumption of co-training, which requires that both views be effective at discriminating the classes (given sufficient labeled data). We refer to this set as the Desert-Trees dataset. Some examples of each class are shown in Figure 1. The complete dataset contains 362 instances. To create the vocabulary of visual words, we used k-means with k=25 (see Section 3.1). The total number of textual features for this dataset is 363.

[1] `http://www.israelimages.com`

(a) Caption: Ibex in Judean Desert

(b) Caption: Bedouin Leads His Donkey That Carries Load Of Straw

(c) Caption: Nuns In Seclusion Near Santa Catherina In Sinai



(d) Caption: Ibex Eating In The Nature

(e) Caption: Entrance To Mikveh Israel Agricultural School

(f) Caption: Olive Trees At The Bell Garden In Jerusalem

Fig. 1. Some images and their corresponding captions of the image dataset. Figures 1(a)-1(c) are of class 'Desert' and the rest are of class 'Trees'.

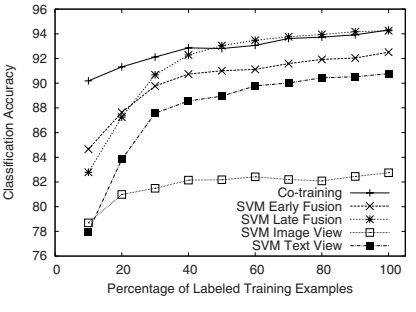
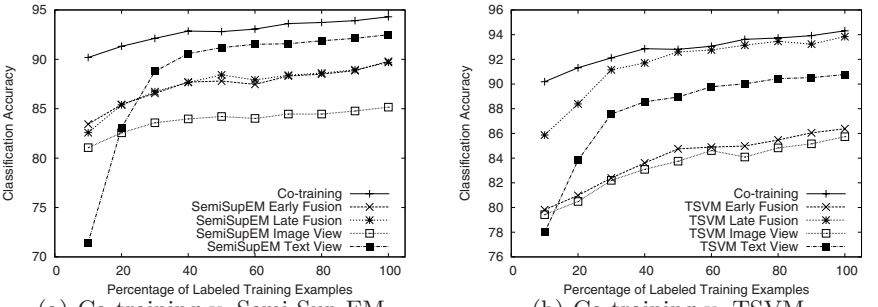


Fig. 2. Comparison of co-training with supervised classifiers on the Desert-Trees dataset. Co-training performs the best, converging with late-fusion for larger amounts of labeled data.

Results and Discussion. Our results comparing co-training with various other classification methods are shown in Figures 2 to 4. In the figures, "Image View" and "Text View" refers to using only the named view's features. The significance of the results were evaluated using a two-tailed paired t-test with a 95% confidence level. Based on preliminary experiments, an RBF kernel (γ=0.01) was used for the SVM in all experiments.



(a) Co-training v. Semi-Sup EM (b) Co-training v. TSVM

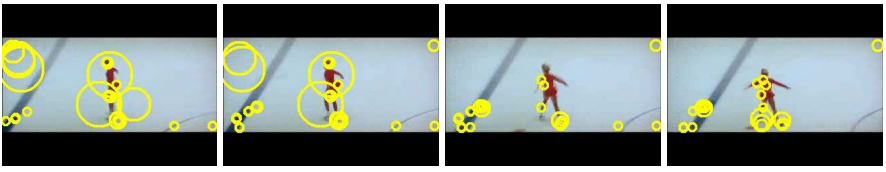
Fig. 3. Comparison of co-training with other semi-supervised techniques on the Desert-Trees captioned images dataset. Co-training outperforms all other methods.

Comparison of Co-training to Supervised Learning. Figure 2 compares co-training using an SVM as the base classifier to supervised classification using an SVM, which is known to often be one of the best performing methods for high-dimensional data in practice. The results show that co-training is more accurate than a supervised SVM using unimodal data and early fusion of multi-modal data, with statistically significant differences at all points on the learning curve. With respect to the individual views, except at the start of the learning curve, the text view performs better than the image view. This is reasonable given that the image cues are often more indirect than the text features. The much smaller number of features in the image view allows it to do a bit better than the text view when training data is extremely limited. Both early and late fusion perform better than the unimodal classifiers since they exploit both views. Co-training is more accurate than late fusion, except for later in the learning curve where they converge. Once all the data is labeled (the last point on the learning curve), co-training and late-fusion are exactly the same since co-training has no unlabeled data to exploit.

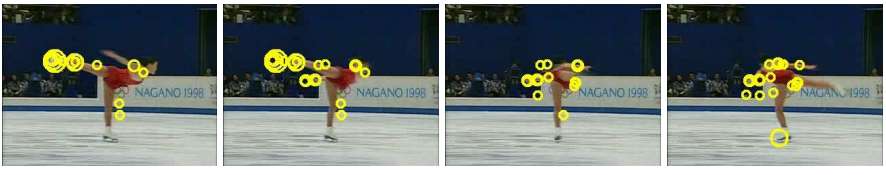
Comparison of Co-training to other Semi-Supervised Methods. Many evaluations of semi-supervised learning only show that the proposed method performs better than supervised learning but do not compare to other semi-supervised methods [6,19,30]. Here we present results comparing co-training with two other well-known semi-supervised techniques: Semi-supervised EM [26] and transductive SVMs [19]. Results are shown in Figures 3(a) and 3(b).

Figure 3(a) shows that co-training with SVM as the base classifier outperforms Semi-Sup EM irrespective of the view it considers, with statistically significant differences across the learning curve.

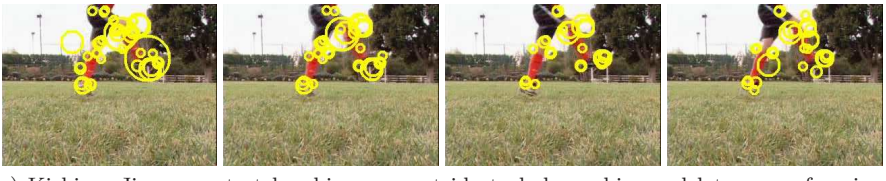
In order to compare with transductive SVM, we have used SVM^{light} [19], with an RBF kernel (γ=0.01) and default values for all other parameters. The figure shows that co-training performs better than transductive SVM irrespective of the view it considers. The difference in accuracy is statistically significant across the learning curve, except when compared to TSVM using late fusion. When compared to TSVM using late fusion, the difference is statistically significant when 40% or less of the training data is labeled.



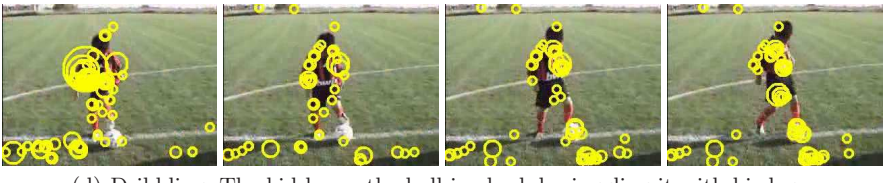
(a) Dancing: Her last spin is going to make her win.



(b) Spinning: A female skating player is revolving in the current position many times, with her posture changing over time.



(c) Kicking: Jim uses stretches his arms outside to balance him and let goes a ferocious drive.



(d) Dribbling: The kid keeps the ball in check by juggling it with his legs.

Fig. 4. Randomly selected consecutive frames of video with detected spatio-temporal interest points. Interest points are displayed as yellow circles around the detected points. One clip per each class of dancing, spinning, kicking, and dribbling is shown above. In addition, the text commentary is also shown below each clip.

Our results are consistent with previous results on text data showing that in domains with two independent and sufficient views, co-training is more effective than Semi-Sup EM [27]. By directly exploiting the redundant nature of the visual and linguistic information, our results indicate that co-training can classify captioned images more accurately than than other semi-supervised methods.

5.2 Learning to Recognize Actions from Commentated Videos

Next we report results using our co-training approach to learn human action categories from commentated videos of athletic events.

Dataset. For this experiment, we collected video clips of soccer and ice skating. One set of video clips is from the DVD titled '1998 Olympic Winter Games:

Figure Skating Exhibition Highlights', which contains highlights of the figure skating competition at the 1998 Nagano Olympics. Another set of video clips is on soccer playing, acquired either from the DVD titled 'Strictly Soccer Individual Skills' or downloaded from YouTube. These videos mostly concentrate on the player in the middle of the screen and usually the motions are repeated several times with different viewpoints. The soccer clips are mainly about soccer specific actions such as kicking and dribbling. There is significant variation in the size of the person across the clips.

The video clips are resized to 240x360 resolution and then manually divided into short clips. The clip length varies from 20 to 120 frames, though most are between 20 and 40 frames. While segmenting activities in video is itself a difficult problem, in this work we specifically focus on classifying pre-segmented clips. The clips are labeled according to one of four categories: kicking, dribbling, spinning and dancing. The first two are soccer activities and the last two are skating activities. The number of clips in each category are, dancing: 59, spinning: 47, dribbling: 55 and kicking: 60. Example frames from each class with detected motion features and their captions are shown in Figure 4. The illustrated features are useful in discriminating between the classes and few features are detected in the background. We used k=200 in the k-means algorithm to create the vocabulary of video features (see Section 3.1).

As the video clips were not originally captioned, we recruited two colleagues unaware of the goals of the project to supply the commentary for the soccer videos. The skating commentary was provided by two of the authors. Additional sample captions are shown in Figure 5. The total number of textual features is 381 for this dataset.

Results and Discussion. In Figure 6 (a), we compare co-training with a supervised SVM using unimodal views and early/late fusion of multi-modal views. Co-training performs better than all other methods early in the learning curve. This demonstrates that utilizing unlabeled data and multi-modal views improves accuracy when supervised data is limited, a valuable advantage. Both co-training and late fusion exploit both views of the dataset, but co-training outperforms late fusion since it also uses the unlabeled data to improve accuracy. It is interesting that early fusion actually performs worse than supervised learning using the text view; we attribute this to the higher-dimensional feature vector, which increases the complexity of learning and impairs generalization.

In many real-life situations, we may not have textual commentary on the novel test videos that we wish to classify. However, even in cases where commentary is not available at test-time, we would still like to benefit from the commentary that was available during training. Therefore, we also examine the case where text is unavailable during testing and an instance must be classified using only video input. Figure 6(b) compares co-training using only the motion view during testing with a supervised SVM using the motion view. In this case, co-training performs better than SVM when only a few labeled examples are available. We also evaluated an analogous situation with the image dataset, but in that case

Spin:
That was a very nice forward camel
Well I remember her performance last time
After gliding, she just starts to make many revolutions while maintaining her current position with her head back.
Her angular movement seems so dizzy because he spins round with her head up and down and also the movement is so fast.
Elizabeth is able to clear this one
Her beautiful performance of revolving herself makes the entire audience impressed due to her perfect posture.
Dancing:
Wow those were some great steps
He has some delicate hand movement
She gave a small jump while gliding
He does slight spins and tries to express bird's motion by dancing like it and goes forward very fast.
The crowd is cheering him a lot
She is drawing a big circle with her arms very fast while moving her body backward and shows lightweightness.
Kick:
His balance is a bit shaky but he manages to execute the kick in the end.
He runs in to chip the ball with his right foot.
He runs in to take the instep drive and executes it well.
He plants his ankle level with the ball and swings though to get the kick and makes sure he has his eyes on the ball all the time.
He come from behind and hits the rolling ball with power just as it rolls in front of him.
He runs behind the ball and has to stretch himself to kick the ball with the inside of his toes.
Dribbling:
Again the striker turns around effortlessly and kicks the ball away from the defender making it look too easy.
At fast speed as the ball is juggled between the legs it becomes difficult to control it.
The small kid pushes the ball ahead with his tiny kicks.
He does the scissors over the ball quickly to move the ball ahead.
He takes the ball with him by alternately pushing the ball forward and swinging the leg over it and using the other leg to distract the defender.
Ran uses the combination of right leg scissor and roll to take the ball ahead

Fig. 5. Captions of some video clips in the four classes

results were comparable to a supervised SVM. All the results are statistically significant until 30% of the data is labeled.

6 Future Work

The image test corpus used in the current experiments is fairly small and requires only binary classification. We would like to test multi-modal co-training on a larger multi-class corpus of captioned images. We would also like to extend our approach to images that do not have explicit text captions but are

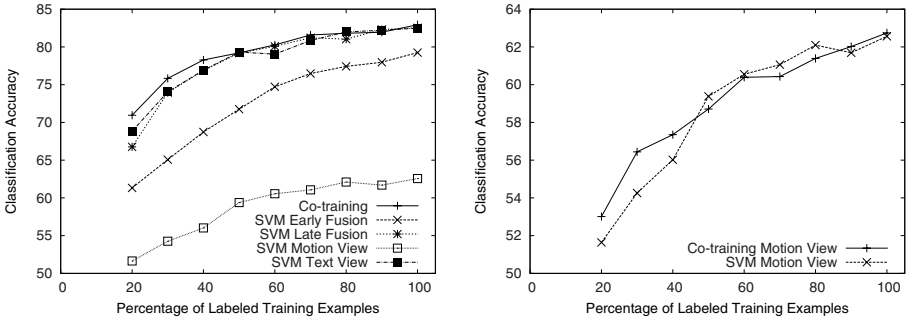


Fig. 6. (a) Comparison of co-training with early fusion, late fusion, motion view and text view on the commentated video dataset. Co-training performs better when only a small fraction of labeled data is available. (b) Co-training compared with supervised learning when text commentary is not available during testing. Co-training performs better when few labeled examples are available.

surrounded by related text. In particular, images on the web rarely come with explicit captions; however, it is natural to use surrounding text productively to find relevant images. By automatically extracting the appropriate surrounding text as a "pseudo-caption," multi-modal co-training could be used to improve the classification of web images. The video commentary in our experiments was added specifically for this project, although we strove to make it natural. In the future, we hope to expand our results to include video with existing closed-captioned commentary and automate the segmentation of video into clips.

7 Conclusion

Recognizing scenes in images and actions in videos are important, challenging problems. We have proposed a solution that uses co-training to exploit both visual and textual features from labeled and unlabeled data to improve classification accuracy. Our results show that such multi-modal co-training can outperform several other standard learning algorithms. By exploiting the redundant information inherent in images or videos and their textual descriptions, we have shown that the amount of supervision required to accurately classify images and videos can be significantly reduced.

Acknowledgment

We thank Ivan Laptev for releasing his code for computing spatio-temporal motion descriptors. This work was funded by grant IIS-0712907 from the U.S. National Science Foundation. The second author is additionally supported by a Samsung Scholarship.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York (1999)
2. Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D.M., Jordan, M.I.: Matching words and pictures. Journal of Machine Learning Research 3, 1107–1135 (2003)
3. Bekkerman, R., Jeon, J.: Multi-modal clustering for multimedia collections. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007). IEEE Computer Society, Los Alamitos (2007)
4. Bennett, K., Demiriz, A.: Semi-supervised support vector machines. Advances in Neural Information Processing Systems 11, 368–374 (1999)
5. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV 2005), pp. 1395–1402 (2005)
6. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory, Madison, WI, pp. 92–100 (1998)
7. Cheng, J., Wang, K.: Active learning for image retrieval with Co-SVM. Pattern Recognition 40(1), 330–334 (2007)
8. Christoudias, C.M., Saenko, K., Morency, L.-P., Darrell, T.: Co-adaptation of audio-visual speech and gesture classifiers. In: ICMI 2006: Proceedings of the 8th international conference on Multimodal interfaces, pp. 84–91. ACM, New York (2006)
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), Washington, DC, USA, vol. 1, pp. 886–893. IEEE Computer Society, Los Alamitos (2005)
10. Duygulu, P., Barnard, K., de Freitas, N., Forsyth, D.: Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)
11. Duygulu, P., Hauptmann, A.G.: What's news, what's not? associating news videos with words. In: Enser, P.G.B., Kompatsiaris, Y., O'Connor, N.E., Smeaton, A.F., Smeulders, A.W.M. (eds.) CIVR 2004. LNCS, vol. 3115, pp. 132–140. Springer, Heidelberg (2004)
12. Efros, A.A., Berg, A.C., Mori, G., Malik, J.: Recognizing action at a distance. In: IEEE International Conference on Computer Vision, Nice, France, pp. 726–733 (2003)
13. Everingham, M., Sivic, J., Zisserman, A.: Hello! My name is... Buffy – Automatic naming of characters in TV video. In: Proceedings of the British Machine Vision Conference (2006)
14. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2004), Washington, DC, USA, vol. 12, p. 178. IEEE Computer Society, Los Alamitos (2004)
15. Fleischman, M., Roy, D.: Situated models of meaning for sports video retrieval. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Rochester, New York, April 2007, pp. 37–40. Association for Computational Linguistics (2007)

16. Forstner, W., Gulch, E.: A fast operator for detection and precise location of distinct points, corners and centres of circular features. In: ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, pp. 281–305 (1987)
17. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
18. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference, pp. 147–152 (1988)
19. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 1999, pp. 200–209 (1999)
20. Ke, Y., Sukthankar, R., Hebert, M.: Event detection in crowded videos. In: IEEE International Conference on Computer Vision (October 2007)
21. Kiritchenko, S., Matwin, S.: Email classification with co-training. In: Proceedings of CASCON 2001, Toronto, Canada, pp. 192–201 (2001)
22. Laptev, I.: On space-time interest points. International Journal of Computer Vision 64(2-3), 107–123 (2005)
23. Levin, A., Viola, P., Freund, Y.: Unsupervised improvement of visual detectors using co-training. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), p. 626. IEEE Computer Society, Los Alamitos (2003)
24. McCallum, A., Nigam, K.: A comparison of event models for naive Bayes text classification. In: Papers from the AAAI 1998 Workshop on Text Categorization, Madison, WI, July 1998, pp. 41–48 (1998)
25. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision (IJCV 2004) 60(1), 63–86 (2004)
26. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39, 103–134 (2000)
27. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM 2000), pp. 86–93 (2000)
28. Nitta, N., Babaguchi, N., Kitahashi, T.: Extracting actors, actions and events from sports video - a fundamental approach to story tracking. In: ICPR 2000: Proceedings of the International Conference on Pattern Recognition, Washington, DC, USA, p. 4718. IEEE Computer Society, Los Alamitos (2000)
29. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Bartlett, P.J., Schölkopf, B., Schuurmans, D., Smola, A.J. (eds.) Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Boston (1999)
30. Quattoni, A., Collins, M., Darrell, T.: Learning visual representations using images with captions. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), June 2007, pp. 1–8. IEEE CS Press, Los Alamitos (2007)
31. Rosenberg, C., Hebert, M., Schneiderman, H.: Semi-supervised self-training of object detection models. In: Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION 2005), Washington, DC, USA, vol. 1, pp. 29–36. IEEE Computer Society, Los Alamitos (2005)
32. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR 2004), Washington, DC, USA, vol. 3, pp. 32–36. IEEE Computer Society, Los Alamitos (2004)

33. Snoek, C.G.M., Worring, M., Smeulders, A.W.M.: Early versus late fusion in semantic video analysis. In: MULTIMEDIA 2005: Proceedings of the 13th annual ACM international conference on Multimedia, pp. 399–402. ACM, New York (2005)
34. Wang, J., Duan, L., Xu, L., Lu, H., Jin, J.S.: Tv ad video categorization with probabilistic latent concept learning. In: Multimedia Information Retrieval, pp. 217–226 (2007)
35. Wang, Y., Sabzmeydani, P., Mori, G.: Semi-latent Dirichlet allocation: A hierarchical model for human action recognition. In: 2nd Workshop on Human Motion Understanding, Modeling, Capture and Animation (2007)
36. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, 2nd edn. Morgan Kaufman Publishers, San Francisco (2005)

Parameter Learning in Probabilistic Databases: A Least Squares Approach

Bernd Gutmann[1], Angelika Kimmig[1], Kristian Kersting[2], and Luc De Raedt[1]

[1] Dept. of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, POBox 2402, BE-3001 Heverlee, Belgium
{firstname.lastname}@cs.kuleuven.be
[2] Fraunhofer IAIS, Schloß Birlinghoven, 53754 Sankt Augustin, Germany
kristian.kersting@iais.fraunhofer.de

Abstract. We introduce the problem of learning the parameters of the probabilistic database ProbLog. Given the observed success probabilities of a set of queries, we compute the probabilities attached to facts that have a low approximation error on the training examples as well as on unseen examples. Assuming Gaussian error terms on the observed success probabilities, this naturally leads to a least squares optimization problem. Our approach, called LeProbLog, is able to learn both from queries and from proofs and even from both simultaneously. This makes it flexible and allows faster training in domains where the proofs are available. Experiments on real world data show the usefulness and effectiveness of this least squares calibration of probabilistic databases.

1 Introduction

Many real-world application today depend on managing enormous volumes of uncertain data. Such "dirty" databases arise for example when integrating data from various sources, when analyzing social, biological, and chemical networks, within privacy-preserving data mining where only aggregated data is available, and within pervasive computing. These are only some of the many real-world applications showing the abundance of uncertain data and the need for probabilistic databases, i.e., generalizations of traditional relational databases that can deal with uncertainty.

Over the last years, the statistical relational learning community has devoted a lot of attention to learning both the structure and parameters of probabilistic logics, cf. [1,2], but so far seems to have devoted little attention to the learning of probabilistic database formalisms. Probabilistic databases [3,4] associate probabilities to facts, indicating the probabilities with which these facts hold. This information is then used to define and compute the success probability of queries or derived facts or tuples, which are defined using background knowledge (in the form of predicate definitions). As one example, imagine a life scientist mining a large network of biological entities in an interactive querying session. The biological network is a probabilistic graph, in which the edges are represented by probabilistic facts about the biological entities [4]. Interesting questions can

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 473–488, 2008.

then be asked about the probability of the existence of *a* connection between two nodes, or the most reliable path between them.

The key contribution of the present paper is the introduction of a novel setting for learning the parameters of a probabilistic database from examples together with their target probability. The task then is to find those parameters that minimize the least squared error w.r.t. these examples. The examples themselves can either be queries or proofs, where a proof is a conjunction of all facts in the database needed to proof a query by SLD-resolution. This learning setting is then incorporated in the probabilistic database ProbLog [4], though it can easily be integrated in other probabilistic databases as well. This yields the second key contribution of the paper, namely an effective learning algorithm called *LeProbLog*[1]. It performs gradient-based optimization utilizing advanced data-structures for efficiently computing the gradient. This efficient computation of the gradient allows us to estimate a ProbLog program from a large real-world network of biological entities in our experiments, which can then be used for example by a life scientist in interactive querying sessions.

We proceed as follows. After reviewing related work in Section 2 and ProbLog in Section 3, we will formally introduce the parameter estimation problem for probabilistic databases in Section 4. Section 5 will then present our least-squares approach LeProbLog for solving it. Before concluding, we will present the results of an extensive set of experiments on a real-world data set.

2 Related Work

The probabilistic database setting differs from the usual statistical relational learning approach in that there is no underlying generative model. Indeed, consider for instance the learning of stochastic logic programs (SLPs) [5], PRISM programs [6], probabilistic relational models (PRMs) [7] or Bayesian logic programs (BLPs) [8]. In all these approaches, a generative model is assumed. For SLPs (and stochastic context-free grammars) as well as for PRISM, the learning procedure assumes that ground atoms for a *single* predicate (or in the grammar case, sentences belonging to the language) are sampled and that the sum of the probabilities of all different atoms obtainable in this way is at most 1. Recently, Chen *et al.* [9] also proposed a learning setting similar to ours. The probabilities associated with examples, however, are viewed as specifying the degree of being sampled from some distribution specified by a generative model, which does not hold in our case. Furthermore, they only provide an algorithm for learning from probabilistic facts and not queries and proofs as we do. PRMs and BLPs are relational extensions of Bayesian networks using entity relationship models or logic programming respectively. In both frameworks, possible worlds, i.e. interpretations, are sampled, and the sum of the probabilities of such worlds is 1. Consider now learning in the probabilistic network sketched above. It is unclear how different paths could be sampled and, clearly, the sum of the probabilities

[1] French for Least square estimation for ProbLog.

of such paths need not be equal to 1. These difficulties explain – in part – why so far only few learning techniques for probabilistic databases have been developed.

The learning setting, however, is in line with the general theory of probabilistic logic learning [10] and inductive logic programming. From an inductive logic programming perspective, a query corresponds to a formula that is entailed by the database, and hence, queries correspond to well-known learning from entailment setting. On the other hand, a proof does not only show *what* was proven but also *how* this was realized. An analogy with a probabilistic context-free grammar is useful here. One can learn the parameters of such a grammar starting from sentences belonging to the grammar (learning from entailment / from queries), or alternatively, one could learn it from parse-trees (learning from proofs), cf. the work on tree-bank grammars [11,12]. The former setting is typically a lot harder than the later one because one query may have multiple proofs, which introduces hidden parameters into the learning setting, which are not present when learning from parse-trees. In the present paper, both types of examples can be combined, and as far as the authors are aware, it is the first time within relational learning and inductive logic programming that learning from proofs is integrated with learning from entailment.

Within the probabilistic database community, parameter estimation has received surprisingly few attention. Nottelmann and Fuhr [13] consider learning probabilistic Datalog rules in a similar setting where the underlying distribution semantics is similar to ProbLog. However, their setting and approach also significantly differ from ours. First, a single probabilistic target predicate only is estimated whereas we consider estimating the probabilities attached to definitions of multiple predicates. Second, their approach employs the training probabilities only. Specifically, they generate training examples labeled with 0/1 randomly according to the observed probabilities whereas we use the observed probabilities directly. Finally, whereas LeProbLog follows a principled gradient approach employing (in principle) all combinations of proofs or explanations, they follow a two-steps bootstrapping approach first estimating parameters as empirical frequencies among matching rules and then selecting the subset of rules with the lowest expected quadratic loss on an hold-out validation set. Gupta and Sarawagi [14] also consider a closely related learning setting but only extract probabilistic facts from data.

Finally, the new setting and algorithm compromise a natural and interesting addition to the existing learning algorithms for ProbLog. It is most closely related to the theory compression setting of [15]. There the task was to remove all but the k best facts from the database (that is to set the probability of such facts to 0), which realizes an elementary form of theory revision. The present task extends the compression setting in that parameters of *all* facts can now be *tuned* starting from evidence. This realizes a more general form of theory revision [16], albeit that only the parameters are changed and not the structure.

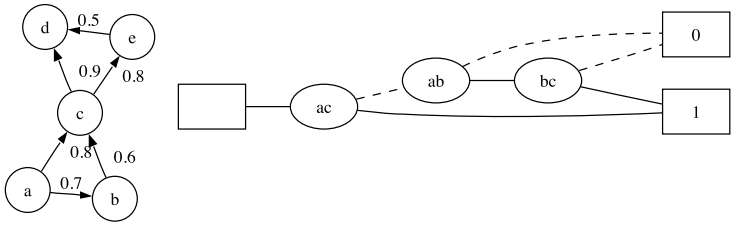


Fig. 1. (a) Example of a probabilistic graph, where edge labels indicate the probability that the edge is part of the graph. (b) Binary Decision Diagram encoding the DNF formula $ac \vee (ab \wedge bc)$, corresponding to the two proofs of query *path(a,c)* in the graph. An internal node labeled xy represents the Boolean variable for the edge between x and y, solid/dashed edges correspond to values true/false.

3 ProbLog

As one example of a probabilistic database, we employ ProbLog, a simple probabilistic extension of Prolog introduced in [4]. Alternatively, the database formalism of [3] or [13] could be used. A ProbLog program consists – as Prolog – of a set of definite clauses. However, in ProbLog every fact c_i is labeled with the probability p_i that its instances $c_i\theta$ are true. It is also assumed that the probabilities of each ground instance $c_i\theta$ (that is, each instance not containing variables) are assumed to be mutually independent. In the following we repeat the main ideas of ProbLog, see [4] for a more detailed explanation.

For ease of illustration, we will consider probabilistic graphs encoded in ProbLog, but the entire discussion carries over to arbitrary ProbLog programs. Figure 1(a) shows a small example that can be encoded in ProbLog as follows:

$0.8 : \texttt{edge(a,c)}.$ $\quad 0.7 : \texttt{edge(a,b)}.$ $\quad 0.8 : \texttt{edge(c,e)}.$
$0.6 : \texttt{edge(b,c)}.$ $\quad 0.9 : \texttt{edge(c,d)}.$ $\quad 0.5 : \texttt{edge(e,d)}.$

It is straightforward to sample subgraphs of a probabilistic graph by tossing a biased coin for each edge. The corresponding ProbLog program $T = \{p_1 : c_1, \cdots, p_n : c_n\}$ therefore defines a probability distribution over subgraphs $L \subseteq L_T = \{c_1, \cdots, c_n\}$ in the following way:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i).$$

It is straightforward to add background knowledge in the form of Prolog clauses, say, the definition of a path by combining edges. We can then ask for the probability that there exists e.g. a path between nodes a and c in our probabilistic graph, i.e. the probability that a randomly sampled subgraph contains the edge from a to c, or the path from a to c via b (or both of them). Formally, the *success probability* $P_s(q|T)$ of a query q in a ProbLog program T is defined as

$$P_s(q|T) = \sum_{L \subseteq L_T} P(q|L) \cdot P(L|T)\ , \tag{1}$$

where $P(q|L) = 1$ if there exists a θ such that $L \models q\theta$, and $P(q|L) = 0$ otherwise. In other words, the success probability of query q corresponds to the probability that the query q is *provable* in a randomly sampled logic program.

As a consequence, the probability of a *specific* proof, also called explanation, corresponds to that of sampling a logic program L that contains all the facts needed in that explanation or proof. The *explanation probability* $P_x(q|T)$ is then defined as the probability of the most likely explanation or proof of the query q:

$$P_x(q|T) = \max_{e \in E(q)} P(e|T) = \max_{e \in E(q)} \prod_{c_i \in e} p_i \tag{2}$$

where $E(q)$ is the set of all explanations for query q [17].

For our example graph and query *path(a,c)*, the set of all explanations contains the edge from a to c (with probability 0.8) as well as the path consisting of the edges from a to b and from b to c (with probability $0.7 \cdot 0.6 = 0.42$). Thus, $P_x(path(a,c)|T) = 0.8$.

Calculating the explanation probability can easily be realized using a best-first search – guided by the probability of the current derivation – through standard logic programming techniques based on the SLD-tree [18]. On the other hand, evaluating the success probability of ProbLog queries is computationally hard, as different proofs of a query are not independent in general. As shown in [4], the problem can be tackled by reducing the problem to that of computing the probability of a monotone DNF formula, an NP-complete problem.

$$P_s(q|T) = P\Big(\bigvee_{e \in E(q)} \bigwedge_{a_i \in cl(e)} a_i \Big) \tag{3}$$

This DNF formula describes each proof in $E(q)$ as a conjunction of Boolean variables, and the entire set as disjunction of these conjunctions. The formula corresponding to our example query *path(a,c)* is $ac \vee (ab \wedge bc)$, where we use xy as Boolean variable representing edge(x,y). To effectively calculate the probability of such a monotone DNF formula, we employ Binary Decision Diagrams (BDDs) [19], an efficient graphical representation of a Boolean function over a set of variables, see Section 6 for more details.

As the size of the DNF formula grows with the number of proofs, its evaluation can become expensive. For instance, when searching for paths in graphs or networks, even in small networks with a few dozen edges there are easily $O(10^6)$ possible paths between two nodes. In [4], an approximation algorithm is proposed that computes both an upper and a lower bound on the probability of a query and searches for more explanations until the difference between the upper and the lower bound becomes sufficiently small.

When learning parameters, we will have to repeatedly evaluate BDDs for all examples. In this context, using a fixed number of proofs allows better control of the overall complexity. We therefore introduce the k-probability $P_k(q|T)$, which approximates the success probability by using the k best (that is, most likely) explanations instead of all proofs when building the DNF formula used in Equation 3:

$$P_k(q|T) = P\Big(\bigvee_{e \in E_k(q)} \bigwedge_{a_i \in cl(e)} a_i \Big) \tag{4}$$

where $E_k(q) = \{e \in E(q) | P_x(e) \geq P_x(e_k)\}$ with e_k the kth element of $E(q)$ sorted by non-increasing probability. Setting $k = \infty$ and $k = 1$ leads to the success and the explanation probability respectively. Using $k = 1$ in parameter learning has also been called Viterbi learning. Finding the k best proofs can be realized using a simple branch-and-bound approach (cf. also [20]).

To illustrate k-probability, we consider again our example graph, but this time with query *path(a,d)*. This query has four proofs, represented by the conjunctions $ac \wedge cd$, $ab \wedge bc \wedge cd$, $ac \wedge ce \wedge ed$ and $ab \wedge bc \wedge ce \wedge ed$, with probabilities 0.72, 0.378, 0.32 and 0.168 respectively. As P_1 corresponds to the explanation probability P_x, we obtain $P_1(path(a,d)) = 0.72$. For $k = 2$, overlap between the best two proofs has to be taken into account: the second proof only adds information if the first one is disconnected. As they share edge *cd*, this means that edge *ac* has to be missing, leading to $P_2(path(a,d)) = P((ac \wedge cd) \vee (\neg ac \wedge ab \wedge bc \wedge cd)) = 0.72 + (1 - 0.8) \cdot 0.378 = 0.7956$. Similarly, we obtain $P_3(path(a,d)) = 0.8276$ and $P_k(path(a,d)) = 0.83096$ for $k \geq 4$. For reasons of memory-efficiency, the implementation used in our experiments below employs iterative deepening for the calculation of lower and upper bounds as well as for P_k with finite k.

4 Parameter Learning in Probabilistic Databases

When new data is added to a database, there is often uncertainty about the data. Text extraction algorithms return the confidence, experimental data is averaged over several runs and so on. Consider for instance populating a probabilistic database of genes from MEDLINE [2] abstracts using off-the-shelves information extraction tools. For example, we could extract that gene *a* is located in region *b* and interacting with *c*. State-of-the art extraction tools, however, often additionally provide a sound probability distribution over the possible outcomes. Hence, we should deal with weighted examples such as 0.6:locatedIn(a,b) and 0.7:interacting(a,c) as already argued e.g. by Gupta and Sarawagi [14] and Chen *et al.* [9]. The situation fits the general learning setting stated in [21]:

> Given is a set of examples E, a probabilistic coverage relation $P(e|D)$ that denotes the probability that the database D covers the example $e \in E$, a theory T in a probabilistic logic, and a scoring function *score*. The goal is to find parameters of T such that the *score* function yields an optimal value.

Concretely instantiating this definition to ProbLog requires us to determine what the examples will be, which probabilistic coverage relation shall be employed, and also determining the scoring function to be optimized. We shall address each of these in turn.

In probabilistic inductive logic programming, examples can be available in different forms. [21] show how one can learn from entailment, from proofs and from interpretations. When learning from entailment, examples are atoms or

[2] `http://medline.cos.com/`

clauses that are logically entailed by a theory, and in the case of a probabilistic logic, assigned a non-zero probability value. Transforming this setting to ProbLog leads to examples that are logical queries. When learning from proofs, examples are proofs, which correspond to concrete explanations in the ProbLog setting. Learning from interpretation in the ProbLog setting is less natural because it requires interpretations containing all the facts that logically follow from the theory. On the other hand, the former two settings can easily be incorporated and actually integrated in ProbLog. The reason is that the logical form of the example will be translated to the monotone DNF formula and it is this last form that is employed by the learning algorithm anyway. The key difference between learning from entailment and learning from proofs in ProbLog is that the DNF formula is a conjunction when learning from proofs and a more general DNF formula when learning from queries. So, using the query *path(a,c)* as example results in $ac \vee (ab \wedge bc)$, whereas the explanation *edge(a,b),edge(b,c)* results in $ab \wedge bc$ only. To the best of our knowledge, this is the first time that an integrated learning from proofs / entailment setting is considered within (probabilistic) inductive logic programming.

Before determining the scoring function and learning setting, it is important to realize that there is also a major difference between probabilistic databases and alternative probabilistic logics, such as PRISM [6] and SLPs [5], even though the probabilistic database semantics seems closely related at first sight. To see this, assume that we now want to estimate the parameters of a ProbLog program starting from example queries, possibly together with their target probability. Continuing our illustration, assume that we are given a number of path queries together with their true probabilities. It is important to observe that the probabilistic database model does not provide a generative model for sampling such queries because the sum of the probabilities of all path queries is not equal to 1 (and in general will be a lot higher). Therefore, we cannot directly apply standard maximum likelihood techniques for parameter estimation based on the EM algorithm as is usually done for statistical relational learning models [1]. The learning mechanisms developed for both PRISM and SLPs assume that there is a generative model from which the examples (ground atoms for a single predicate) can be sampled and the probability mass associated to the set of all examples is maximum 1. This observation explains also why we consider a different setting for probabilistic databases, in which parameter learning is viewed as a function optimization problem. The problem then is that we seek a set of parameters that approximates the actual query probabilities as close as possible, which in turn explains why rather than maximizing the likelihood of the data, we shall minimize the least squared error between the target probabilities of the examples and the probability of the model, but see below.

Finally, let us remark that the choice of probabilistic coverage relation is open, and that therefore, within ProbLog we choose the k-probability as this allows for maximal flexibility. By now we can instantiate the above definition to obtain the problem-setting tackled in this paper:

Definition 1 (Parameter Learning in Probabilistic Databases). ***Given*** *a set of training examples* $\{q_i, \tilde{p}_i\}_{i=1}^{M}$, $M > 0$, *where each* $q_i \in \mathcal{H}$ *is a query or proof and* $\tilde{p}_i$ *is the k-probability of* q_i, ***find*** *a function* $h : \mathcal{H} \to [0, 1]$ *with low approximation error on the training examples as well as on unseen examples.* $\mathcal{H}$ *comprises all parameter assignments for a given database* T.

This framework allows to naturally combine *learning from entailment* and *learning from proofs*, two learning settings that so far have been considered separately. In ProbLog, proofs correspond to conjunctions of probabilistic facts, and can be seen as a conjunction of queries. Therefore, a learning algorithm can use examples of both forms, (atomic) queries and proofs, at the same time. To realize *learning from interpretations*, probability estimates could be obtained using simple counting. However, this is infeasible for domains where interpretations contain high fractions of facts assigned value true. Finally, the *error function* that we want to minimize is the mean squared error:

$$MSE(T) = \frac{1}{M} \sum\nolimits_{1 \leq i \leq M} \left(P_s(q_i|T) - \tilde{p}_i \right)^2 . \tag{5}$$

It is easy to see that minimizing the squared error corresponds to finding a maximum likelihood hypothesis, provided that each training example $(q_i, \tilde{p}_i)$ is disturbed by a Gaussian error $\tilde{p}_i$, i.e. $\tilde{p}_i = p(q_i) + e_i$, with $p(q_i)$ the actual probability of query q_i and e_i drawn independently from a zero-mean Gaussian. See [22, Chapter 6.4] for a detailed derivation.

Gradient descent is a standard way of minimizing a given error function. The tunable parameters are initialized randomly. Then, as long as the error did not converge, the gradient of the error function is calculated, scaled by the learning rate η, and subtracted from the current parameters. In the following sections, we derive the gradient of the MSE and show how it can be computed efficiently.

5 Gradient of the Mean Squared Error

Applying the sum and chain rule to Eq. (5) yields the partial derivative

$$\frac{\partial MSE(T)}{\partial p_j} = \frac{2}{M} \sum_{1 \leq i \leq M} \underbrace{\left(P_s(q_i|T) - \tilde{p}_i \right)}_{\text{Part 1}} \cdot \underbrace{\frac{\partial\ P_s(q_i|T)}{\partial p_j}}_{\text{Part 2}} . \tag{6}$$

Part 1 can be calculated by a ProbLog inference call computing (1). It does not depend on j and has to be calculated only once in every iteration of a gradient descent algorithm. Part 2 can be calculated as following

$$\frac{\partial P_s(q_i|T)}{\partial p_j} = \sum_{\substack{S \subseteq L_T \\ S \models q_i}} \delta_{jS} \prod_{\substack{c_x \in S \\ x \neq j}} p_x \prod_{\substack{c_x \in L_T \setminus S \\ x \neq j}} (1 - p_x) , \tag{7}$$

where $\delta_{jS} := 1$ if $c_j \in S$ and $\delta_{jS} := -1$ if $c_j \in L_T \setminus S$. It is derived by first deriving the gradient $\partial P(S|T)/\partial p_j$ for a fixed subset $S \subseteq L_T$ of facts, which is straight-forward, and then summing over all subsets S where q_i can be proven.

Algorithm 1. Evaluating the gradient of a query efficiently by traversing the corresponding BDD, calculating partial sums, and adding only relevant ones

function GRADIENT(BDD b, fact to derive for n_j)
  $(val, seen) =$ GRADIENTEVAL$(root(b), n_j)$
  If $seen = 1$ return $val \cdot \sigma(a_j) \cdot (1 - \sigma(a_j))$
  Else return 0
function GRADIENTEVAL(node n, target node n_j)
  If n is the 1-terminal return $(1, 0)$
  If n is the 0-terminal return $(0, 0)$
  Let h and l be the high and low children of n
  $(val(h), seen(h)) =$ GRADIENTEVAL(h, n_j)
  $(val(l), seen(l)) =$ GRADIENTEVAL(l, n_j)
  If $n = n_j$ return $(val(h) - val(l), 1)$
  ElseIf $seen(h) = seen(l)$ return $(\sigma(a_n) \cdot val(h) + (1 - \sigma(a_n)) \cdot val(l), seen(h)))$
  ElseIf $seen(h) = 1$ return $(\sigma(a_n) \cdot val(h), 1)$
  ElseIf $seen(l) = 1$ return $((1 - \sigma(a_n)) \cdot val(l), 1)$

To ensure that all p_j stay probabilities during gradient descent, we reparameterize the search space and express each $p_j \in]0, 1[$ in terms of the sigmoid function[3] $p_j = \sigma(a_j) := 1/(1 + \exp(-a_j))$ applied to $a_j \in \mathbb{R}$. This technique has been used for Bayesian networks and in particular for sigmoid belief networks [23]. We derive the partial derivative $\partial P_s(q_i|T)/\partial a_j$ in the same way as (7) but we have to apply the chain rule one more time due to the σ function

$$\sigma(a_j) \cdot (1 - \sigma(a_j)) \cdot \sum_{\substack{S \subseteq L_T \\ L \models q_i}} \delta_{jS} \prod_{\substack{c_x \in S \\ x \neq j}} \sigma(a_x) \prod_{\substack{c_x \in L_T \setminus S \\ x \neq j}} (1 - \sigma(a_x)).$$

We also have to replace every p_j in Eq. (1) by $\sigma(p_j)$. Going over all subprograms S in the last equation is infeasible. But there is an efficient algorithm to compute $P_s(q_i|T)$ relying on BDDs [4]. In the following section we update this towards the gradient and introduce LeProbLog, the gradient descent algorithm for ProbLog.

6 LeProbLog

To compute the success probability P_∞ for a query q efficiently, De Raedt *et al.* [4] collect all proofs and compactly represent them in a Binary Decision Diagram (BDD) [19]. BDDs are one of the best understood data structures today. They have been used to solve a wide variety of computer science problems. Given a fixed variable ordering, a Boolean function f can be represented as a full Boolean decision tree where each node on the ith level is labeled with the ith variable

[3] The sigmoid function can induce plateaus which might slow down a gradient-based search. However, it is unlikely that a plateau will spread out over several dimensions and we did not observe such a behavior in our experiments. If it happens though, one can take standard counter measures like simulated annealing or random restarts.

Algorithm 2. LeProbLog, the algorithm takes a program without probabilities as input, minimizes the MSE on the training set by gradient descent and returns a ProbLog program with probabilities

Require: a ProbLog program without probabilities L_T
training set $q_j, \tilde{p}_j$ $1 \leq j \leq M$
learning rate η
k, the number of proofs used to generate the BDDs
Ensure: parameters p_i $1 \leq i \leq n$
initialize all a_j randomly
while not converged **do**
$\Delta\mathbf{a} := \mathbf{0}$
for $1 \leq i \leq M$ **do**
find k best proofs and generate BDD_i for q_i
$y := \frac{2}{M} \cdot \left(P_s(q_i|T) - \tilde{p}_i\right)$
for $1 \leq j \leq n$ **do**
$\Delta a_j := \Delta a_j + y \cdot \frac{\partial P_s(q_i|T)}{\partial a_j}$ {call GRADIENT$(BDD_i, node_j)$}
$\mathbf{a} := \mathbf{a} - \eta \cdot \Delta\mathbf{a}$
return T, that is $\{\sigma(a_j) : c_j \mid c_j \in L_T\}$ {A ProbLog program with probabilities}

and has two children called low and high. Leaves are labeled by the outcome of f for the variable assignment corresponding to the path to the leaf, where in each node labeled x, the branch to the low (high) child is taken if variable x is assigned 0 (1). Starting from such a tree, one obtains a BDD by merging isomorphic subgraphs and deleting redundant nodes until no further reduction is possible. A node is redundant if the subgraphs rooted at its children are isomorphic. Figure 1(b) shows the BDD for the existence of a path between a and c in our earlier example.

The algorithm of De Raedt *et al.* [4] calculates the probability of a Boolean formula by traversing the BDD bottom-up, in each node summing the probability of the high and low child, weighted by the probability of the node's variable being assigned true and false respectively. We extended this to the computation of the gradient (7). Both algorithms have a time and space complexity of O(number of node in the BDD) when intermediate results are cached.

Let us first consider a full decision tree instead of a BDD. Each branch in the tree represents a product $n_1 \cdot n_2 \cdot \ldots \cdot n_i$, where the n_i are the probabilities associated to the corresponding variable assignment of nodes on the branch. The gradient of such a branch b with respect to n_j is $g_b = n_1 \cdot n_2 \cdot \ldots n_{j-1} \cdot n_{j+1} \cdot \ldots \cdot n_i$ if n_j is true, and $-g_b$ if n_j is false in b. As all branches in a full decision tree are mutually exclusive, the gradient w.r.t. n_j can be obtained by simply summing the gradients of all branches ending in a leaf labeled 1. In BDDs however, isomorphic sub-parts are merged, and obsolete parts are left out. This implies that some paths from the root to the 1-terminal may not contain n_j, therefore having a gradient of 0. So, when calculating the gradient on the BDD, we have to keep track of whether n_j appeared on a path or not. Given that the variable order is the same on all paths, we can easily propagate this information in our bottom-up algorithm. This is exactly what is described in Algorithm 1. Specifically,

GradientEval(n, n_j) calculates the gradient w.r.t. n_j in the sub-BDD rooted at n. It returns two values: the gradient on the sub-BDD and a Boolean indicating whether or not the target node n_j appears in the sub-BDD. When at some node n the indicator values for the two children differ, we know that n_j does not appear above the current node, and we can drop the partial result from the child with indicator 0. The indicator variable is also used on the top level: Gradient returns the value calculated by the bottom-up algorithm if n_j occurred in the BDD and 0 otherwise.

LeProbLog combines the BDD-based gradient calculation with a standard gradient descent search. Starting from parameters $\mathbf{a} = a_1, \ldots, a_n$ initialized randomly, the gradient $\Delta\mathbf{a} = \Delta a_1, \ldots, \Delta a_n$ is calculated, parameters are updated by subtracting the gradient, and updating is repeated until convergence. When using the k-probability with finite k, the set of k best proofs may change due to parameter updates. After each update, we therefore recompute the set of proofs and the corresponding BDD. Algorithm 2 shows the pseudocode of LeProbLog.

7 Experiments

We set up experiments to investigate the following questions:

Q1 Does our approach reduce the mean squared error on training and test data?
Q2 Is our approach able to recover the original parameters?

Answering these first questions will serve as a sanity check for the algorithm and our implementation.

Q3 Is it necessary to update the set of k best proofs in each iteration?

As building BDDs for all examples is expensive, building BDDs once and using them during the entire learning process can save significant amounts of resources and time. We are therefore interested in the effects this strategy has on the results.

Q4 Can we obtain good results approximating P_∞ by P_k for finite (small) k?

Given that using BDDs to calculate P_∞ is infeasible for huge sets of proofs, as they occur in our application, where we easily get hundreds of thousands of proofs, we are interested in fast, reliable approximations.

Q5 Do the results improve when parts of the training examples are given as proof?

Here we are interested in exploring the effects of providing more information in the form of proofs, which is one of the main distinguishing features of LeProbLog.

To answer these questions, we extracted graphs around both Alzheimer disease and asthma from a collection of databases. For each disease, we obtained a set of related genes by searching Entrez for human genes with the relevant annotation (AD or asthma); corresponding phenotypes for the diseases are from OMIM. Most of the other information comes from EntrezGene, String, UniProt, HomoloGene, Gene Ontology, and OMIM databases. Weights were assigned to

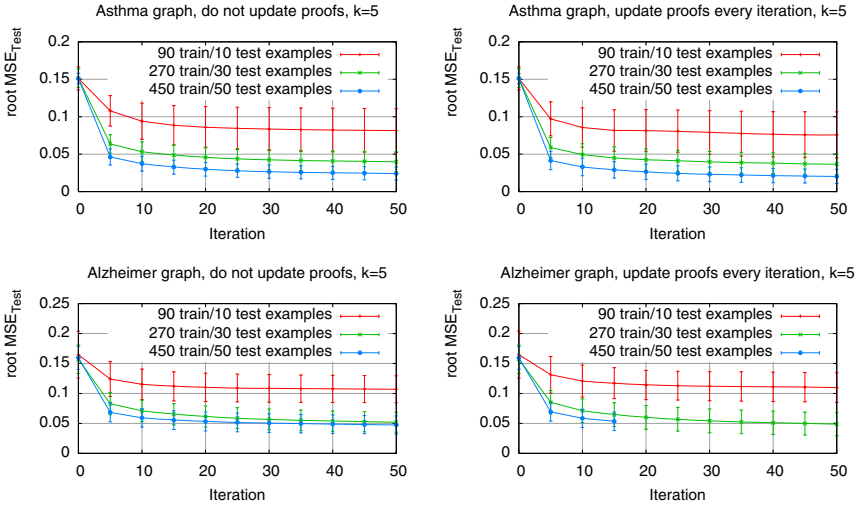


Fig. 2. $\sqrt{MSE_{\text{Test}}}$ for asthma and Alzheimer using the 5 best proofs ($k = 5$); when the BDDs and proofs are not updated (left column); when they are updated every iteration (right column) (**Q2** and **Q3**)

edges as described in [24]. In the experiments below, we used a fixed number of randomly chosen (Alzheimer disease or asthma) genes for graph extraction. Subgraphs were extracted by taking all acyclic paths of no more than length 4, with a probability of at least 0.01, between any given gene and the corresponding phenotype. Some of the genes did not have any such paths to the phenotype and are thus disconnected from the rest of the graph. The resulting graph around Alzheimer contains 122 nodes and 259 edges, that around Asthma 127 nodes and 241 edges. From these graphs we generated 3 sorts of training sets:

1. We sampled 500 random node pairs from the asthma and Alzheimer graph and estimate the query probability for path(a,b) using P_5, the probability of the 5 best proofs. These two sets are used to answer **Q1**, **Q2**, and **Q3**.
2. We sampled 200 random node pairs from the asthma graph and estimated $P_\infty(path(a,b))$ using the lower bound of the approximative inference algorithm [4] with interval width $\delta = 0.01$. This set is used to answer **Q4**.
3. We sampled 300 random node pairs and calculated P_1 for path(a,b), the probability of the best path between a and b. We then build several sets where different fractions of the examples where given as proof, the edges of the best path, instead of the path(a,b) query, and used them to answer **Q5**.

To assess results, we use the root mean squared error on the test data $\sqrt{MSE_{\text{test}}}$, and the mean absolute difference MAD_{facts} between learned p_j and original fact probabilities p_j^{true}: $MAD_{\text{facts}} := n^{-1} \sum_{j=1}^{n} |p_j - p_j^{\text{true}}|$. We always sampled the initial fact parameters uniformly in the interval $[-0.5, 0.5]$. Applying the sigmoid function yields probability values with mean 0.5 ± 0.07. The datasets used, had

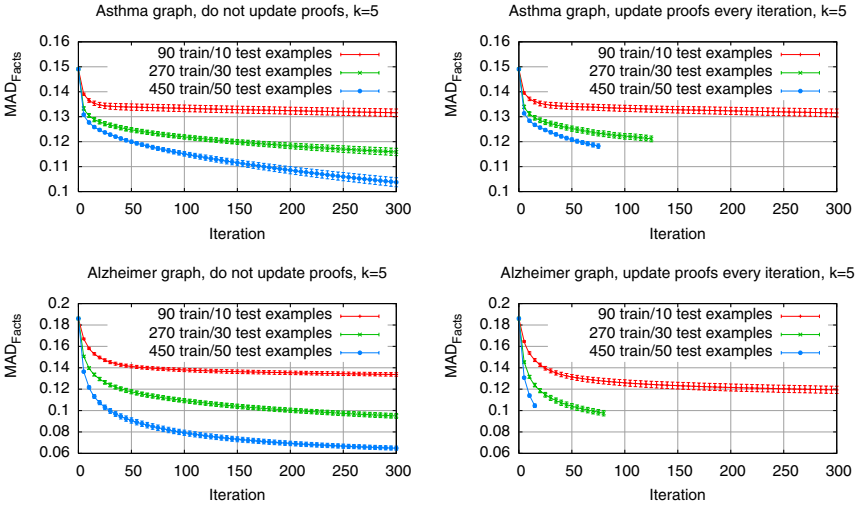


Fig. 3. MAD_{facts} for asthma and Alzheimer using the 5 best proofs ($k = 5$); when the BDDs and proofs are not updated (left column); when they are updated every iteration (right column) (**Q2** and **Q3**)

fact probabilities in this range and we therefore got lower initial errors than by completely random initialization. In general, one can utilize prior knowledge to initialize the parameters. We perform 10-fold crossvalidation in all experiments. The learning rate η was always set to the number of training examples. LeProbLog was implemented in Prolog (Yap-5.1.3) using CUDD for BDD operations.

Q1, Q2: Sanity Check. We attach probabilities to queries in the training set based on the best $k = 5$ proofs. The same approximation is used in the gradient descent algorithm, where the set of proofs to build the BDD is determined anew in every iteration as stated in Algorithm 2. We repeated the experiment using a total of 100, 300, and 500 examples, which we each split in ten folds for crossvalidation. We thus use 90, 270, and 450 training examples. The more training examples are used, the more time each iteration takes. In the same amount of time, the algorithm therefore performs less iterations when using more training examples. The right column of Figure 2 shows the evolvement of the root mean squared error on the test data during learning. The gradient descent algorithm reduces the MSE on both training and test data, with significant differences in all cases (two-tailed t-test, $\alpha = 0.05$). These results affirmatively answer **Q1**.

The MAD_{facts} error is reduced as can be seen in the right column of Figure 3. Again, all differences are significant (two-tailed t-test, $\alpha = 0.05$). Using more training examples results in faster error reduction. This answers **Q2** affirmatively. It should be noted however that in other domains, especially with limited or noisy training examples, minimizing the MSE might not reduce MAD_{facts}, as the MSE is a non-convex non-concave function with local minima.

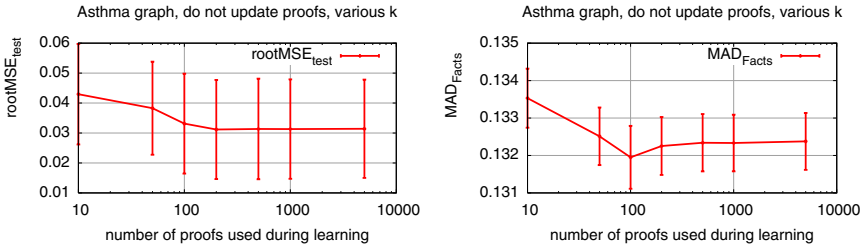


Fig. 4. MAD_{facts} and $\sqrt{MSE_{\text{Test}}}$ after 50 iterations for different k (number of best proofs used) on the asthma graph where training examples carry P_∞ probabilities (**Q4**)

Q3: Error made when the best proofs are not updated. We repeated the same series of experiments, but without updating the set of proofs used for constructing the BDDs. The evolvement of $\sqrt{MSE_{\text{Test}}}$ as well as of MAD_{Facts} is plotted in the left column of Figures 2 and 3 respectively.

The plots for the asthma graph are hardly distinguishable and there is indeed no significant difference (two-tailed t-test, $\alpha = 0.05$). However, the runtime decreases by orders of magnitude, since searching for proofs and building BDDs are expensive operations which had to be done only once in the current experiments. Not updating the BDDs gave a speedup of 10 for the Alzheimer graph. For the Alzheimer graph there is no significant difference for the MSE_{test} (two-tailed t-test, $\alpha = 0.05$), but MAD_{facts} is reduced a little slower (in terms of iterations) when the BDDs are kept constant. However, in terms of time this is not the case. These results indicate that BDDs can safely be kept fixed during learning in this domain which affirmatively answers **Q3**.

Q4: Less proofs, more speed, and still the right results?. In the next experiment, we studied the influence of the number k of best proofs used during learning on the results. We consider the asthma graph with the second dataset, where training example probabilities are lower bounds obtained from the approximation algorithm with interval width 0.01. During learning, P_k is employed to approximate probabilities.

We ran LeProbLog on this dataset and used different values of k between 10 and 5000. We thus aim at learning parameters using an underestimate of the true function, as k best proofs may ignore a potentially large number of proofs used originally. Figure 4 shows the results for this experiment after 50 iterations of gradient descent. As can be seen, the average absolute error per fact (MAD_{facts}) goes down slightly with higher k. The difference is statistically significant for $k = 10$ and $k = 100$ (two-tailed t-test, $\alpha = 0.05$), but using more than 200 proofs has no significant influence on the error. The MSE also decreases significantly (two-tailed t-test, $\alpha = 0.05$) comparing the values for $k = 10$ and $k = 200$, but using more proofs has no significant influence. It takes more time to search for more proofs and to build the corresponding BDDs. These results indicate that using only 100 proofs is a sufficient approximation in this domain and affirmatively answer **Q4**.

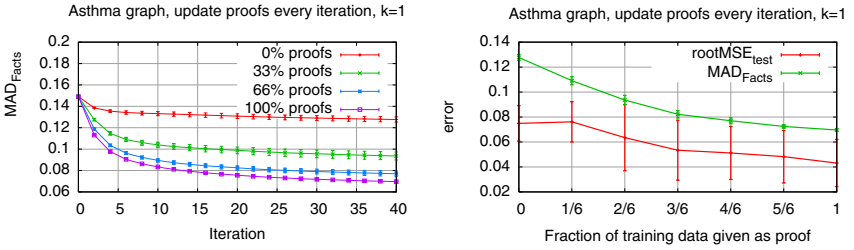


Fig. 5. MAD_{facts} and $\sqrt{MSE_{\text{Test}}}$ after 40 iterations on the asthma graph when different fractions of the data are given as proof (**Q5**)

Q5: Learning from Proofs and Queries. To investigate the effect of using both proofs and queries as examples, we compute the best proof and its probability for 300 examples per graph. For each example, we either use the query or the best proof, both with the probability of the best proof. Learning uses $k = 1$. We use proofs for $0, 50, \ldots, 300$ examples and queries for the remaining ones, and perform stratified 10-fold crossvalidation, that is the ratio of examples given as queries and as proofs was the same in every fold. We updated BDDs in every iteration. Figure 5 shows the results of this experiment. The curve on the left side indicates that the error per fact (MAD_{facts}) goes down faster in terms of iterations when increasing the fraction of proofs. Furthermore, the plot on the right side shows that the root MSE on the test set decreases. These results answer **Q5** affirmatively.

8 Conclusions

We have introduced an approach to learning the parameters of the probabilistic database ProbLog and successfully shown it at work on a real biological application. Interesting directions for future research include conjugate gradient techniques and regularization-based cost functions. Those enable domain experts to successively refine probabilities of a database by stating training examples.

Acknowledgments

Angelika Kimmig and Bernd Gutmann are supported by the Research Foundation-Flanders (FWO-Vlaanderen), Kristian Kersting by a Fraunhofer ATTRACT fellowship. This work is supported by the GOA project 2008/08 Probabilistic Logic Learning and uses HPC resources `http://ludit.kuleuven.be/hpc`.

References

1. Getoor, L., Taskar, B. (eds.): Statistical Relational Learning. MIT Press, Cambridge (2007)
2. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911, pp. 1–27. Springer, Heidelberg (2008)

3. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: Proceedings of VLDB, pp. 864–875 (2004)
4. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: Veloso, M. (ed.) IJCAI, pp. 2462–2467 (2007)
5. Cussens, J.: Parameter estimation in stochastic logic programs. MLJ 44(3), 245–271 (2001)
6. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res. (JAIR) 15, 391–454 (2001)
7. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: IJCAI, pp. 1300–1309 (1999)
8. Kersting, K., De Raedt, L.: Basic principles of learning bayesian logic programs. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.) Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911, pp. 189–221. Springer, Heidelberg (2008)
9. Chen, J., Muggleton, S., Santos, J.: Learning probabilistic logic models from probabilistic examples (extended abstract). In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 22–23. Springer, Heidelberg (2008)
10. De Raedt, L., Kersting, K.: Probabilistic Logic Learning. ACM-SIGKDD Explorations: Special issue on Multi-Relational Data Mining 5(1), 31–48 (2003)
11. Charniak, E.: Tree-bank grammars. In: AAAI/IAAI, vol. 2, pp. 1031–1036 (1996)
12. De Raedt, L., Kersting, K., Torge, S.: Towards learning stochastic logic programs from proof-banks. In: AAAI, pp. 752–757 (2005)
13. Nottelmann, H., Fuhr, N.: Learning probabilistic datalog rules for information classification and transformation. In: CIKM, pp. 387–394. ACM, New York (2001)
14. Gupta, R., Sarawagi, S.: Creating probabilistic databases from information extraction models. In: VLDB, pp. 965–976 (2006)
15. De Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic prolog programs. Machine Learning 70(2-3), 151–168 (2008)
16. Wrobel, S., Wettschereck, D., Sommer, E., Emde, W.: Extensibility in data mining systems. In: KDD, pp. 214–219 (1996)
17. Kimmig, A., De Raedt, L., Toivonen, H.: Probabilistic explanation based learning. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 176–187. Springer, Heidelberg (2007)
18. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer, Berlin (1989)
19. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers 35(8), 677–691 (1986)
20. Poole, D.: Logic programming, abduction and probability - a top-down anytime algorithm for estimating prior and posterior probabilities. New Generation Comput. 11(3), 377–400 (1993)
21. De Raedt, L., Kersting, K.: Probabilistic inductive logic programming. In: Ben-David, S., Case, J., Maruoka, A. (eds.) ALT 2004. LNCS (LNAI), vol. 3244, pp. 19–36. Springer, Heidelberg (2004)
22. Mitchell, T.M.: Machine Learnning. McGraw-Hill, New York (1997)
23. Saul, L., Jaakkola, T., Jordan, M.: Mean field theory for sigmoid belief networks. JAIR 4, 61–76 (1996)
24. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: Link discovery in graphs derived from biological databases. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 35–49. Springer, Heidelberg (2006)

Improving k-Nearest Neighbour Classification with Distance Functions Based on Receiver Operating Characteristics

Md. Rafiul Hassan[1], M. Maruf Hossain[1], James Bailey[1,2], and Kotagiri Ramamohanarao[1,2]

[1] Department of Computer Science and Software Engineering, The University of Melbourne, Australia
[2] NICTA Victoria Laboratory, The University of Melbourne, Australia
{mrhassan,hossain,jbailey,rao}@csse.unimelb.edu.au

Abstract. The k-nearest neighbour (k-NN) technique, due to its interpretable nature, is a simple and very intuitively appealing method to address classification problems. However, choosing an appropriate distance function for k-NN can be challenging and an inferior choice can make the classifier highly vulnerable to noise in the data. In this paper, we propose a new method for determining a good distance function for k-NN. Our method is based on consideration of the area under the Receiver Operating Characteristics (ROC) curve, which is a well known method to measure the quality of binary classifiers. It computes weights for the distance function, based on ROC properties within an appropriate neighbourhood for the instances whose distance is being computed. We experimentally compare the effect of our scheme with a number of other well-known k-NN distance metrics, as well as with a range of different classifiers. Experiments show that our method can substantially boost the classification performance of the k-NN algorithm. Furthermore, in a number of cases our technique is even able to deliver better accuracy than state-of-the-art non k-NN classifiers, such as support vector machines.

Keywords: Receiver Operating Characteristics (ROC), k-Nearest Neighbour, Feature Weighting, Classification, Gene Expression.

1 Introduction

The k-nearest neighbour (k-NN) technique is a classic, simple and appealing method to address classification problems. It has a very intuitive interpretation and its predictions are easily explained to domain experts. Although k-NN has been applied for classification in many domains, it tends to suffer from poor classification accuracy when i) there are very many features, ii) when there are very few instances, or iii) the data is very noisy. These weaknesses make it difficult to use k-NN for datasets such as gene expression data, which are very noisy and typically have thousands of features, but only tens or at most hundreds of instances. Indeed in the gene expression domain, k-NN has been adopted by very few researchers (e.g. [1, 2]), due to its generally inferior performance.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 489–504, 2008.
Springer-Verlag Berlin Heidelberg 2008

Example 1. *In the ALL-AML leukaemia gene expression dataset, k-NN has an average accuracy of only* 83.33±3.49% *using* 10×10*-fold cross validation, whereas Support Vector Machines (SVM) yield an average accuracy of* 98.61±1.26%. *This lower accuracy of the k-NN classifier can discourage biologists to use it.*

A wide range of proposals have been made to improve k-NN. These principally propose alternative ways of computing the distance function, since using different distance functions can yield vastly different classification performance. Indeed ideally, the distance function used for k-NN should be adapted to the particular problem being solved [3].

Our aim in the paper is to improve the classification performance of k-NN using a new type of distance function. It is based on a feature weighting scheme that considers receiver operating characteristics that are appropriate to the points whose distance is being computed.

In a nutshell, we present a method to derive a distance function for k-NN based on feature weighting. The weight for each feature is calculated by considering the area under the Receiver Operating Characteristics (ROC) curve [4]. This value is equivalent to the Mann-Whitney U statistic normalized by the number of possible pairings of positive and negative values, also known as the two sample Wilcoxon rank-sum statistic [5]. The area under the ROC curve (AUC) actually represents the probability that a randomly chosen positive example is correctly ranked with greater suspicion than a randomly chosen negative example. Moreover, this probability of correct ranking is the same quantity estimated by the non-parametric Wilcoxon statistic [6].

The intuitive outline of the technique is as follows:

For a given dataset $\mathcal{D}$ of n instances comprising m features: $x_1, x_2, x_3, \ldots, x_m$, each feature x_i (where $1 \leqslant i \leqslant m$) has some discriminative power, i.e., the influence of each feature on the classification accuracy can be measured. The ROC curve is plotted for a series of pairs which are each formed by a threshold value for the "classifier" feature x_i and the corresponding class label Y_i. Then, when calculating the distance of a new test instance from a training example, the distance measure is modified using the AUC score as weight for that feature.

A crucial question faced by the technique is which values of a feature should be used to help derive the ROC curve whose area corresponds to each weight. We shall show that, surprisingly, using all values of a feature is not the best strategy. Instead, considerably better performance can be obtained by selecting from an appropriate neighborhood for each feature, specific to the instances whose distance is being calculated.

Related Work. Since there exist many works related to k-NN, we can only briefly cover a representative selection. Early works addressing the improvement of k-NN include Kira and Rendell [7] and Salzberg [8], whose approaches rely on an interactive system architecture in which users are asked to rate a given similarity prediction, and then use reinforcement learning to enhance the distance function based on the user feedback. Kononenko [9] proposed an extension to this for updating feature weights based on intracluster weights.

Klein *et al.* [10] proposed a shortest path algorithm to modify a Euclidean distance function based on prior knowledge. Stein and Niggemann [11] used a neural network approach to learn weights of distance functions based on training examples.

Other approaches rely on an underlying class structure to evaluate distance functions. Han *et al.* [12] employed a randomized hill-climbing approach to learn weights of distance functions for classification tasks. In their approach, k-NN queries were used to evaluate distance functions; the k-neighbourhood of each object is analysed to determine to which extent the class labels agree with the class label of each object. Zhang [13] suggested the use of kernel functions and multidimensional scaling to learn Euclidean metrics. Hastie and Tibshirani [3] proposed algorithms that learn adaptive rectangular neighborhoods (rather than distance functions) to enhance nearest-neighbour classifiers.

Other types of approaches includes work by Hastie and Tibshirani [14] and Domeniconi *et al.* [15], who considered schemes for locally adaptive distance functions that vary throughout the input space. In particular, Domeniconi *et al.* [15] suggested using the decision boundaries of SVMs to induce a locally adaptive distance function for k-NN.

Driessens *et al.* [16] presented a two-stage classifier, YATSI, that improves its predictive accuracy by making use of the available unlabeled data. It used a weighted nearest neighbor classification algorithm using the combined example-sets as a knowledge base.

Feature weighting schemes for k-NN have also been proposed. Im and Park [17] proposed a hybrid expert system of case-based reasoning and neural network, which uses a value difference metric as the distance function for symbolic features. In another study, Vivencio *et al.* [18] proposed a feature weighting method based on the χ-squared test for k-NN.

Receiver operating characteristics have previously been used to improve classifiers. These include algorithms such as ROC-tree [19] and work by Ferri *et al.* [20], which both propose using ROC information to build decision trees. A justification for using ROC for feature evaluation, is given in Deng *et al.* [21], who demonstrate that in the context of gene expression data, ROC is a superior method to the t-test.

Contributions. Our main contributions in this paper are as follows:

- Development of a new feature weighting technique to derive a distance function for k-NN. This technique employs a range-wise feature weighting scheme based on ROC, that can dynamically determine which weights are most appropriate for the points whose distance is being measured.
- An experimental investigation which demonstrates that our new algorithm (known as *ROC-kNN*) performs strongly compared to and often outperforms well-known techniques like C4.5 [22], Random Forest, Vivencio *et al.*'s [18] χ^2-FW weighted k-NN, YATSI [16], Ferri *et al.*'s [20] AUCsplit, ROC-tree [19], and SVMs in terms of accuracy as well as overall AUC value. Surprisingly, the predictive power of *ROC-kNN* is comparable to the "black box" SVM approach, making it a very attractive tool for domain experts.

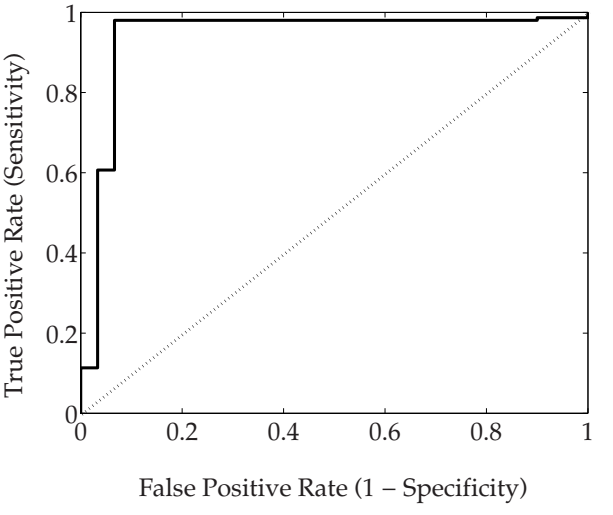


Fig. 1. A typical ROC curve

2 The Receiver Operating Characteristic Curve (ROC): Preliminaries

ROC curves were first used in signal detection theory [4]. In machine learning, the ROC curve is used to evaluate the discriminative performance of binary classifiers. This is obtained by plotting the curve of the true positive rate (*Sensitivity*) versus the false positive rate (1 – *Specificity*) for a binary classifier by varying the discrimination threshold. Figure 1 shows a typical ROC curve.

All the calculations of true positive rate and false positive rate are attained when using a particular classifier threshold. By varying the threshold, a set of values for these measurements is obtained. This set of values is plotted in a two-dimensional Cartesian graph to yield the ROC curve. The ROC curve takes into account all the possible solutions by varying the discriminative threshold. The best performance would be produced, if the ROC curve matches with the upper left corner of the ROC space (which yields 100% *sensitivity* and 100% *specificity*). The closer the ROC curve is to the upper part of the ROC space, the better the performance of the classifier.

An ROC curve is a two dimensional illustration of classifier performance. Reducing ROC performance to a single scalar value to represent expected performance helps compare classifiers. A popular method is to calculate the *area under the ROC curve (AUC)* [5].

The AUC, being a part of the area of the unit square, has a value between 0 and 1. Since random guessing could produce the diagonal line between $(0, 0)$ and $(1, 1)$ with an area of 0.5, a classifier with an AUC less than 0.5 is undesirable [23]. An AUC value close to 1 indicates better performance for a binary classifier. [24].

3 ROC-kNN: A *k*-NN Algorithm Using ROC Information

We now describe the steps in our algorithm, which we call *ROC-kNN*.

3.1 ROC for Feature Weighting

Previous work [25, 19, 20] has established the use of an ROC curve for feature ranking and selection, to identify the discriminative features in the context of gene expression microarray data. First, the ROC curve is plotted for each of the pairs formed by each of the features and the class label. This means treating a single feature as a classifier and calculating the classification in terms of the *sensitivity* and *specificity* by varying the operating point. We shall build on this kind of idea to derive a feature weighting method to use in distance functions. For each feature, the AUC is calculated.

EXAMPLE 2. *Let us consider a dataset $\mathcal{D}$ of N instances, where each instance comprises m features: x_1, x_2, x_3, ..., x_m. Each of the m features has a differing discriminative power reflected by its respective AUC. To calculate the discriminative power that is expressed in terms of AUC, we plot the ROC curve for each feature paired with the class label, (i.e., $\{x_i, Y_i\}$, where $1 \leqslant i \leqslant m$ and Y is the vector of class labels) and calculate the AUC of this ROC curve.*

As alluded to earlier, there is a strong mathematical justification for using the ROC to measure discriminative power. It is equivalent to the Mann-Whitney U Test (also known as Wilcoxon Rank sum), a non-parametric statistical test. Not employing any distributional assumptions makes it especially useful for small sample size, noisy datasets [21], such as gene expression microarrays.

3.2 Weighted Distance Metrics

For calculating the distance of a new test instance from a training instance, we modify the standard distance measure using the AUC score as a weight. Example 3 describes how we employ the weight in **Minkowski distance** of order p (ℓ_p**-norm distance**). Recall that Euclidean distance is ℓ_2 distance, rectilinear, Manhattan or Hamming distances are ℓ_1 distance, and Chebyshev distance is ℓ_∞ distance.

EXAMPLE 3. *Consider a training instance and a test instance each with m feature values: x_1, x_2, x_3, ..., x_m and y_1, y_2, y_3, ..., y_m, respectively. Then the Minkowski distance between the instances is $\delta = \left(\sum_{i=1}^{m} |x_i - y_i|^p \right)^{\frac{1}{p}}$. We can associate a weight for each feature based on its AUC score. The weighted distance function is $\Delta = \left(\sum_{i=1}^{m} (\mathcal{A}_i \cdot |x_i - y_i|)^p \right)^{\frac{1}{p}}$, where each $\mathcal{A}_i$ is the AUC value for the i-th feature.*

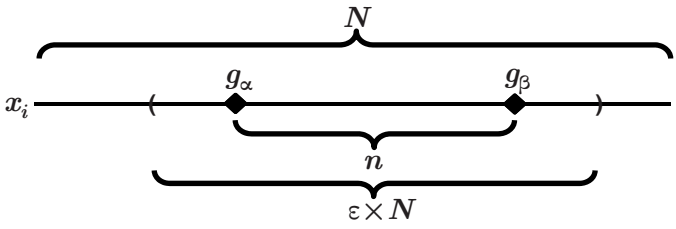


Fig. 2. Widening $[g_\alpha, g_\beta]$ so that it covers $\epsilon \times N$ values of feature x_i

3.3 Considering a Smaller Range of Feature Values to Calculate AUC

Using all values of a feature to derive the ROC curve whose area will be calculated, may not be the best way to measure the weight or "importance" of a feature. Consider the following (somewhat artificial) example.

Example 4. *Consider the power of the feature voice_pitch for predicting the class label* sex *(male/female). Suppose the* voice_pitch *feature values for the population range between 50Hz (low pitch) to 350Hz (high pitch). In general, voice_pitch is likely to be a good feature for discriminating between males and females. However, if we are only dealing with a sub-population of young children, then voice_pitch is likely to be a far less useful feature, since there is much less variation for values of this feature across young children.*

Thus, the power of a feature may need to be evaluated within some context or sub-population. A natural way to form such a context or range for the ROC calculation, is to consider the two points between which the distance is being computed. Suppose we are computing the distance between two instances P_1 and P_2 with respect to feature x_i an $P_1[x_i] = g_\alpha$ and $P_2[x_i] = g_\beta$[1]. Rather than using all values that occur for feature x_i in ROC calculation, we just use the values that lie in the interval $[g_\alpha, g_\beta]$. This range of values $[g_\alpha, g_\beta]$ corresponds to a sub-population that is more appropriate for computing the ROC of feature x_i, in the context of P_1 and P_2.

Now, it could be the case that the range $[g_\alpha, g_\beta]$ is very small. This could then to lead to low confidence in the resulting AUC calculation and estimate of feature significance, since the sample size would be too small to be statistically significant. We, therefore, generalise this idea by employing a parameter, ϵ, which can be thought of as a "coverage factor". The interval $[g_\alpha, g_\beta]$ covers some number of values (say n) of the total number of values (say N) that instances can have for feature x_i [2]. Thus, the number of values not covered by $[g_\alpha, g_\beta]$ on x_i is $N - n$. Now, ϵ varies between 0% and 100% and it is a lower bound on how much coverage we require for our interval. For example, suppose $\epsilon = 50\%$, then we require at least half the values in x_i to be covered by the

[1] Where $P_i[x_i]$ denotes the value for the instance P_i on feature x_i.

[2] Of course different instances can share the same value on x_i. Any such value will be counted more than once when calculating n or N.

Algorithm 1. CalculateROC

Input(s): $\mathcal{C}$: A two column matrix of training examples with the first column being the values for feature x_i and the last column being values for the class label, ϵ: The percentage of instance values to be covered, g_α: Value of the training instance on x_i, g_β: Value of the test sample on x_i
Output: $\mathcal{A}$: The AUC of the attribute x_i

Sort $\mathcal{C}$ in descending order of x_i
if $g_\alpha \leqslant g_\beta$ **then**
  $startPoint \leftarrow g_\alpha$
  $endPoint \leftarrow g_\beta$
else
  $startPoint \leftarrow g_\beta$
  $endPoint \leftarrow g_\alpha$
end if
$N \leftarrow$ The number of instances
$SamplesToUse \leftarrow \epsilon \times N$
$SamplesInRange \leftarrow$ The collection of all values for x_i which are between $[startPoint, endPoint]$
$SizeSIR \leftarrow$ Calculate the size of $SamplesInRange$
while $SizeSIR < SamplesToUse$ **do**
  $SamplesInRange \leftarrow SamplesInRange$ union one instance value that is adjacent to it
  Update $startPoint$ and $endPoint$ accordingly
  if No more samples can be added to either $startPoint$ or $endPoint$ **then**
    $startPoint \leftarrow$ The last available sample's on that side
    $endPoint \leftarrow$ The last available sample on that side
  end if
  $SizeSIR \leftarrow$ Calculate the size of $SamplesInRange$
end while
$\mathcal{RC} \leftarrow$ The samples between the range $[startPoint, endPoint]$
$\mathcal{A} \leftarrow$ AuROC($\mathcal{RC}$)
return $\mathcal{A}$
end

interval $[g_\alpha, g_\beta]$. Now, if $[g_\alpha, g_\beta]$ doesn't cover 50% of N, then our strategy is to widen it (symmetrically) just enough until it does achieve this level of coverage. If $\epsilon = 0$, then the interval $[g_\alpha, g_\beta]$ will never widened. If $\epsilon = 100\%$, then $[g_\alpha, g_\beta]$ will always be widened to encompass the entire range of values for feature x_i. Figure 2 illustrates the general idea.

Intuitively, range adjustment using ϵ allows the interval to be widened sufficiently so that enough feature values are available to ensure statistical significance of the ROC calculation. However, we do not want to widen the interval too much (by always choosing a high ϵ), since the resulting measure of feature discriminative power may not be focused on an appropriate population for the chosen instances P_1 and P_2 (recall Example 4). Hence, choosing a good ϵ is important. In practice, this can be done by empirically comparing classification performance for different choices of ϵ.

Algorithm 2. ROC-kNN

Input(s): $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$: The matrix of n training examples with the last column being the class, $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n)$: The test sample, k: The number of neighbours, p: The order for Minkowski distance function, ϵ: The percentage of training instances to be covered when weighting each feature

Output: $\mathcal{C}$: The class label for the test sample $\boldsymbol{\tau}$

1: **for each** labelled instance $(\mathbf{x}_i, Y_i), (i = 1, \ldots, n)$ **do**
2: **for each** feature $a_j, 1 < j <$ Number of features, m **do**
3: $\mathcal{A}_j = \text{CALCULATEROC}(\{x_j, Y\}, \epsilon, \mathbf{x}_i, \boldsymbol{\tau})$ /* $\mathcal{A}$ is a vector of AUC scores for all features */
4: **end for**
5: Calculate $\Delta(\mathbf{x}_i, \boldsymbol{\tau}) = \left(\sum_{j=1}^{m} (\mathcal{A}_j \cdot |\mathbf{x}_i[a_j] - \boldsymbol{\tau}[a_j]|)^p \right)^{\frac{1}{p}}$
6: **end for**
7: Sort $\Delta(\mathbf{x}_i, \boldsymbol{\tau})$ in ascending order
8: $\mathcal{D}^k_{\boldsymbol{\tau}}$=$k$ nearest instances to $\boldsymbol{\tau}$
9: $\mathcal{C} \leftarrow$ most frequent class in $\mathcal{D}^k_{\boldsymbol{\tau}}$
10: **return** $\mathcal{C}$
11: **end**

EXAMPLE 5. *Suppose the 15 instances of the training dataset have values* {1, 2, 2, 3, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 12} *for feature* x_i *and* $P_1[x_i] = 8$, $P_2[x_i] = 9$ *and* $\epsilon = 40\%$. *Then the interval* $[8, 9]$ *must be widened so that it includes* $0.4 \times 15 = 6$ *of the values that occur for* x_i *in training instances. This can be accomplished by (symmetrically) widening it to be* $[6, 11]$. *So the discriminative power of* x_i *in this situation will be measured by the AUC of the ROC curve derived from when* x_i *takes values* $\{6, 7, 8, 9, 10, 11\}$.

DEFINITION 1. *Given instances* P_1 *and* P_2 *from a dataset with* m *features* $x_1, \ldots, x_m$ *and values for parameters* ϵ *and* p. *The weighted distance between* P_1 *and* P_2 *using range adjusted ROC is calculated as*

$$\left(\sum_{i=1}^{m} (\mathcal{A}_i \cdot |P_1[x_i] - P_2[x_i]|)^p \right)^{\frac{1}{p}},$$

where each $\mathcal{A}_i$ *measures the discriminative power using ROC of feature* x_i *in the interval* $r = [P_1[x_i] - \alpha_1, P_2[x_i] + \alpha_2]$ *for some* α_1, α_2. *The interval* r *covers at least* $\epsilon\%$ *of the values taken by feature* x_i *for instances in the training dataset.*

Thus, the weighting of each feature x_i is specific to the points whose distance is being computed. Carrying out the weight calculation can be done at either runtime (k-NN classification time) or during training. If the latter, then one must assume test instances do not contain any feature values that are not present in the training data. In that situation, one precomputes the ROC for every contiguous interval of values for each feature x_i and then selects the appropriate ROC weight value at runtime, according to the values on x_i of the instances which are being compared.

Table 1. Properties of the datasets used in this study

Dataset	No. of Attributes	No. of Instances	Collected from	First used by
GE1	24,481	97	Integrated Tumor Transcriptome Array and Clinical data Analysis database [27]	van 't Veer *et al.* [26]
GE2	3,226	22	National Human Genome Research Institute	Hedenfalk *et al.* [28]
GE3	12,533	181	Division of Thoracic and Surgery [30], Brigham Women's Hospital, Boston	Gordon *et al.* [29]
GE4	12,600	21	Cancer Program [31], Broad Institute of MIT and Harvard	Singh *et al.* [32]
GE5	12,600	136	Cancer Program [31], Broad Institute of MIT and Harvard	Singh *et al.* [32]
GE6	7,129	72	Cancer Program [33], Broad Institute of MIT and Harvard	Golub *et al.* [34]
Hepatitis	19	155	UCI ML Repository [35]	–
Ionosphere	34	351	UCI ML Repository [35]	–
Pima	8	768	UCI ML Repository [35]	–
WBC	9	699	UCI ML Repository [35]	–
WDBC	30	569	UCI ML Repository [35]	–
WPBC	33	198	UCI ML Repository [35]	–

Algorithm 1 presents the pseudocode for calculating the AUC for a specified feature x_i. It relies the existence on the function AUROC function provided in [19]. The overall procedure is described in Algorithm 2 (which assumes weight computation is done at runtime).

4 Experimental Design

k-NN is actually a family of techniques, according to k value and distance function used. In our evaluation, we tested using different values of k (1, 3 and 5) and different choices of p for the Minkowswki ℓ_p-norm distance (1, 2 and ∞). For our algorithm, *ROC-kNN*, we also needed to test using different values for the parameter ϵ (100%, 95%, 90%, ..., 0%).

Based on this testing, for k-NN and *ROC-kNN* on each dataset, we identified the values of p, k and ϵ that produced the best classification performance and this is what is reported in the result tables.

In addition to comparing against traditional k-NN, we compared *ROC-kNN* against thirteen other techniques. These are: Vivencio *et al.*'s [18] χ^2-FW weighted k-NN, YATSI [16], ROC-tree [19], C5.0, its predecessor C4.5 [22], Random Forest, Ferri *et al.*'s [20] AUCsplit technique for decision trees, Naïve Bayes and SVMs using two different kernels: polynomial and radial basis function (RBF). Where applicable, each of these classifiers was run multiple times on each dataset by varying its parameters. We report the best result of each such classifier on each dataset across the variation of its parameters. Since the χ-squared test can only handle

Table 2. Comparison of accuracy results from 10×10-fold cross validation on six gene expression datasets

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-kNN	63.29 ± 3.13	71.07 ± 4.22	98.66 ± 0.39	61.49 ± 2.12	84.65 ± 1.22	90.33 ± 0.89
k-NN	58.45 ± 2.88	61.82 ± 6.14	94.64 ± 0.27	57.14 ± 3.89	82.35 ± 1.80	88.89 ± 0.93
χ^2-FW	49.90 ± 3.64	50.00 ± 10.05	90.77 ± 0.45	58.57 ± 5.96	80.81 ± 1.78	89.72 ± 1.63
YATSI	56.70 ± 3.21	45.45 ± 4.19	93.92 ± 1.09	57.14 ± 1.93	72.06 ± 1.99	84.72 ± 1.57
ROC-tree	$\mathbf{72.16 \pm 4.32}$	$\mathbf{77.27 \pm 2.45}$	98.34 ± 0.89	38.10 ± 5.95	88.24 ± 2.33	94.44 ± 2.96
AUCsplit	63.58 ± 4.59	74.39 ± 1.63	96.14 ± 1.36	34.01 ± 2.87	82.47 ± 3.96	81.61 ± 3.28
C5.0	64.95 ± 6.21	59.09 ± 4.52	92.82 ± 1.21	23.81 ± 4.65	81.62 ± 4.12	80.55 ± 3.74
C4.5	62.89 ± 3.11	72.73 ± 1.36	95.03 ± 1.05	33.33 ± 4.59	79.42 ± 5.45	79.17 ± 4.87
ADTree	61.86 ± 4.28	68.18 ± 5.68	92.82 ± 2.19	32.86 ± 3.44	86.76 ± 2.63	86.11 ± 3.77
REPTree	52.18 ± 5.45	59.09 ± 3.92	95.03 ± 1.28	32.86 ± 3.46	80.88 ± 3.33	81.94 ± 4.26
Random Tree	55.67 ± 3.54	63.64 ± 2.58	79.56 ± 2.69	32.86 ± 3.12	62.50 ± 5.23	75.00 ± 3.90
Random Forest	62.89 ± 6.43	50.00 ± 5.33	93.92 ± 1.22	38.10 ± 5.27	80.88 ± 2.56	79.17 ± 2.36
Naïve Bayes	54.64 ± 3.38	59.09 ± 4.58	98.34 ± 0.03	33.33 ± 0.78	55.88 ± 4.76	98.61 ± 1.03
SVM (poly)	68.04 ± 2.14	59.09 ± 2.98	$\mathbf{99.45 \pm 0.11}$	47.62 ± 5.63	$\mathbf{91.18 \pm 3.12}$	$\mathbf{98.61 \pm 1.26}$
SVM (RBF)	67.01 ± 2.36	63.64 ± 0.94	98.34 ± 1.41	$\mathbf{61.90 \pm 1.39}$	69.12 ± 5.31	80.56 ± 2.18

discrete data, we used entropy-based discretization [36] to discretize the datasets before applying χ^2-FW.

Each of the classifiers was applied on 12 datasets, of which 6 were gene expression datasets and 6 are non-gene expression datasets having rather different characteristics. The properties of the datasets are illustrated in Table 1. Recall that gene expression datasets represent one of the most challenging scenarios for k-NN algorithms. For each classifier and dataset, a 10-fold cross validation (CV) scheme was used 10 times.

Using a fixed 10-fold cross validation scheme, we also conducted a win-draw-loss analysis based on a paired t-test with 5% significance level, for six of the classifiers.

5 Results and Discussion

The classification results for all 14 techniques on the considered datasets are presented in Table 2 and 3. The best performances among that of the reported classifiers are marked in bold. Since for the non-gene expression datasets, different ϵ values yielded best classification accuracy on different datasets, we also include a separate entry for *ROC-kNN* with $\epsilon = 100\%$. For the gene expression datasets, $\epsilon = 100\%$ always yielded the best result. We show the chosen ϵ values for *ROC-kNN* in Table 4. We also show in that table the chosen values for k and and p for both *ROC-kNN* and k-NN.

ROC-kNN versus k-NN Techniques. Looking at Table. 2 and 3, the classification performance of *ROC-kNN* shows considerable improvement over the traditional k-NN technique. For all datasets and different k values, *ROC-kNN* has improved accuracy over k-NN.

Results of statistical significance tests we conducted (Bonferroni-corrected t-tests [37]) confirm that the improvement in accuracy is statistically significant. For example, when comparing the best results of the *ROC-kNN* with that of the k-NN, on 5 of the 6 non-gene expression datasets (hepatitis, ionosphere, WBC,

Table 3. Comparison of accuracy results from 10 × 10-fold cross validation on six non-gene expression datasets

Method	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
ROC-kNN	$\mathbf{85.59 \pm 0.55}$	90.56 ± 0.22	$\mathbf{98.26 \pm 0.23}$	$\mathbf{98.49 \pm 0.21}$	79.26 ± 0.51	$\mathbf{88.51 \pm 0.96}$
ROC-kNN ($\epsilon = 100\%$)	82.98 ± 0.87	88.60 ± 0.12	97.07 ± 0.14	97.47 ± 0.24	77.02 ± 0.68	86.60 ± 1.09
k-NN	82.58 ± 0.80	87.35 ± 0.59	96.88 ± 0.39	97.24 ± 0.29	76.21 ± 1.31	85.28 ± 1.87
χ^2-FW	78.84 ± 1.60	53.05 ± 13.27	95.55 ± 0.28	95.87 ± 0.59	70.76 ± 1.36	70.83 ± 0.61
YATSI	80.00 ± 2.12	83.48 ± 2.39	96.28 ± 1.01	94.73 ± 1.48	71.21 ± 2.76	73.70 ± 1.98
ROC-tree	78.71 ± 7.65	84.05 ± 9.87	92.56 ± 5.43	90.69 ± 6.78	69.67 ± 8.33	63.54 ± 8.65
AUCsplit	82.10 ± 3.43	86.00 ± 7.31	95.88 ± 1.94	93.75 ± 3.39	70.53 ± 9.67	73.82 ± 5.35
C5.0	76.13 ± 2.35	89.46 ± 1.23	93.64 ± 1.65	93.29 ± 2.23	70.70 ± 4.12	73.94 ± 2.76
C4.5	80.00 ± 4.45	91.45 ± 3.36	93.84 ± 2.63	93.15 ± 1.26	75.25 ± 3.32	73.83 ± 2.89
ADTree	76.13 ± 2.96	$\mathbf{93.16 \pm 1.65}$	95.14 ± 1.77	94.02 ± 1.06	77.78 ± 5.42	72.92 ± 3.23
REPTree	78.71 ± 4.23	89.46 ± 1.46	93.99 ± 2.14	92.44 ± 2.33	73.74 ± 4.85	75.39 ± 4.55
Random Tree	72.91 ± 9.21	87.75 ± 3.64	94.13 ± 2.85	89.46 ± 3.67	68.18 ± 5.45	67.97 ± 6.49
Random Forest	81.94 ± 1.26	92.59 ± 3.26	95.99 ± 1.45	95.25 ± 1.37	78.28 ± 3.47	73.70 ± 4.98
Naïve Bayes	83.87 ± 1.71	82.62 ± 3.48	95.99 ± 0.74	92.97 ± 2.58	67.68 ± 5.08	76.30 ± 3.49
SVM (poly)	76.77 ± 4.23	88.60 ± 2.43	96.85 ± 1.07	97.72 ± 1.04	76.26 ± 4.78	77.34 ± 5.01
SVM (RBF)	84.52 ± 4.02	91.74 ± 5.15	96.85 ± 1.29	96.07 ± 3.12	$\mathbf{81.31 \pm 2.36}$	77.47 ± 3.73

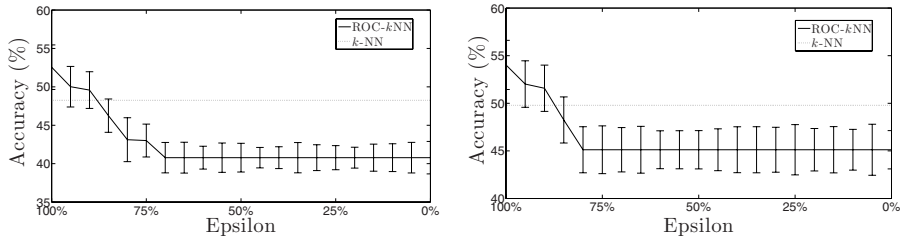
Table 4. The parameters of k-NN and *ROC-kNN*

Dataset	k-NN		*ROC-kNN*			
	k	p	k	p	ϵ	$\epsilon \times N$
GE1	3	∞	1	∞	100%	88 ± 0.0
GE2	1	2	1	2	100%	20 ± 0.0
GE3	1	2	3	∞	100%	163 ± 0.0
GE4	5	2	1	2	100%	19 ± 0.0
GE5	3	1	1	1	100%	123 ± 0.0
GE6	1	1	1	1	100%	65 ± 0.0
Hepatitis	3	2	5	2	65%	91 ± 1.8
Ionosphere	1	∞	5	∞	70%	221 ± 2.3
WBC	5	2	5	2	60%	378 ± 1.5
WDBC	3	1	3	1	75%	384 ± 2.75
WPBC	3	1	3	1	85%	152 ± 2.15
Pima Indians	3	2	3	2	55%	380 ± 2.6

WDBC and Pima Indians) the p-value is less than 0.05 (0.041533, 0.011329, 0.029657, 0.046574 and 0.034136, respectively). WPBC showed milder improvement (p-value is 0.1056256). For gene expression data, Bonferroni-corrected t-tests confirmed significant improvement of *ROC-kNN* over k-NN (p-values are $0.04123, 0.01298, 0.03192, 0.04871, 0.03907$ and 0.04516 for GE1 to GE6 datasets, respectively).

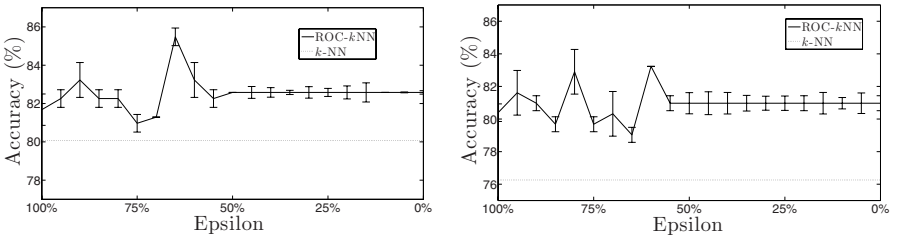
The results for the χ^2-FW weighted k-NN technique show it mostly failed to improve classification accuracy over k-NN. This is interesting, since both *ROC-kNN* and χ^2-FW use feature weighting to extend k-NN, yet each uses a rather different method. The poor performance of the χ^2-FW technique may be because it is intended more for discrete data, rather than the continuous datasets used in our evaluation (and thus it required an extra discretisation step).

ROC-kNN versus non k-NN Classifiers. Surprisingly and pleasingly, the classification performance of *ROC-kNN* on the UCI ML repository datasets is extremely strong. It outperforms all the other non k-NN classifiers, except on



(a) ℓ_1-norm distance (b) ℓ_2-norm distance

Fig. 3. The effect of ϵ on classification accuracy for GE1 dataset



(a) ℓ_2-norm distance (b) ℓ_∞-norm distance

Fig. 4. The effect of ϵ on classification accuracy for the hepatitis dataset

Table 5. Comparison of AUC value from 10×10-fold cross validation on six gene expression datasets

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-kNN	0.5992 ± 0.02	$\mathbf{0.6721 \pm 0.11}$	0.9665 ± 0.02	0.4812 ± 0.12	$\mathbf{0.8386 \pm 0.02}$	0.8743 ± 0.03
k-NN	0.5793 ± 0.04	0.5064 ± 0.14	0.8463 ± 0.04	0.4378 ± 0.13	0.8213 ± 0.02	0.8585 ± 0.03
χ^2-FW	0.4986 ± 0.08	0.4958 ± 0.12	0.7418 ± 0.05	$\mathbf{0.5174 \pm 0.11}$	0.8015 ± 0.03	0.8682 ± 0.04
YATSI	$\mathbf{0.6230 \pm 0.00}$	0.4150 ± 0.02	$\mathbf{0.9820 \pm 0.00}$	0.4760 ± 0.01	0.8050 ± 0.00	$\mathbf{0.905 \pm 0.00}$

ionosphere and WPBC. This provides some evidence that *ROC-kNN* is a classifier which is able to compete with and even surpass mainstream state-of-the-art techniques in these circumstances. On the gene expression datasets, *ROC-kNN* performs quite strongly compared to the non-k-NN classifiers, but it is not the standout performer. This could be because for these datasets, the number of instances is very small and thus the ROC measure of feature discriminative power is less reliable.

Influence of the ϵ parameter: Figure 3 and 4 show the effect on classification performance for varying ϵ on both the GE1 dataset and the hepatitis dataset along with the improvement over k-NN, which is represented by a dotted line in the figures. Table 4 also shows the best ϵ values for each dataset. Though the effect of ϵ is not entirely the same on every dataset, there are some clear trends.

Table 6. Comparison of AUC value from 10×10-fold cross validation on six non-gene expression datasets

Method	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
ROC-kNN	0.7491 ± 0.02	0.8790 ± 0.02	0.9816 ± 0.01	0.9799 ± 0.00	$\mathbf{0.7815 \pm 0.04}$	$\mathbf{0.8728 \pm 0.01}$
k-NN	0.7306 ± 0.03	0.8322 ± 0.02	0.9675 ± 0.00	0.9694 ± 0.01	0.6278 ± 0.03	0.7894 ± 0.03
χ^2-FW	0.5928 ± 0.04	0.5139 ± 0.03	0.9530 ± 0.01	0.9499 ± 0.01	0.5228 ± 0.04	0.6508 ± 0.01
YATSI	$\mathbf{0.7820 \pm 0.00}$	$\mathbf{0.9200 \pm 0.00}$	$\mathbf{0.9890 \pm 0.00}$	$\mathbf{0.9830 \pm 0.00}$	0.575 ± 0.01	0.7600 ± 0.01

Table 7. Win-Draw-Loss results for the top six top classifiers using t-test on 72 test combinations for each classifier

Method	Win	Draw	Loss
ROC-kNN	**46**	**20**	**6**
k-NN	29	26	17
χ^2-FW	8	20	44
YATSI	11	24	37
ROC-tree	22	28	22
SVM (poly)	19	38	15
SVM (RBF)	18	42	12

For gene expression data, $\epsilon = 100\%$ is (unsurprisingly) always the best choice, due to the small number of instances in this type of data and classification accuracy falls monotonically with decreasing ϵ. However, in non-gene expression data, the trend is quite different. Classification accuracy can increase and decrease according to varying ϵ and the best accuracy is achieved with different ϵ values for each dataset. It is worth noting that for all datasets, the performance of the classifier becomes constant once ϵ drops below a certain value (see Fig. 3 and 4). This is because, after a point, the ϵ constraint becomes too loose and never forces any widening of a feature interval.

Comparison of AUC Values: We also computed the overall AUC value of selected k-NN style classifiers, shown in Table 5 and 6, resulting from the 10×10 cross validation over the gene expression and UCI ML repository datasets. The AUC values of the four considered k-NN classifiers: *ROC-kNN*, k-NN, χ^2-FW weighted k-NN and YATSI reflect the much the same behaviour as seen for classification accuracy, except for the YATSI classifier. Though its classification accuracy is not as good as *ROC-kNN* or even traditional k-NN, YATSI performed slightly better than *ROC-kNN* for 3 out of 6 gene expression datasets and 4 out of 6 non-gene expression data in terms of AUC. *ROC-kNN* has improved AUC compared to k-NN in both the gene expression and non-gene expression data.

Statistical significance tests: We also carried out win-draw-loss analysis based on paired t-test with 5% significance level for the seven most promising classifiers (see Tab. 7). In this analysis, *ROC-kNN* arguably outperforms all the other techniques, as it has a significant number of wins and only a few losses, compared to the other classifiers. A more detailed look at *ROC-kNN*'s performance against the other classifiers (see Tab. 8), also reveals that it has significant

Table 8. Win-Draw-Loss result for *ROC-kNN* versus the top five classifiers using t-test

Method	GE1	GE2	GE3	GE4	GE5	GE6	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
k-NN	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win
χ^2-FW	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win	Win
YATSI	Win	Win	Draw	Win	Win	Win	Win	Win	Draw	Win	Win	Win
ROC-tree	Loss	Draw	Draw	Loss	Draw	Draw	Win	Win	Win	Win	Win	Win
SVM (poly)	Draw	Win	Draw	Win	Loss	Loss	Win	Draw	Draw	Draw	Draw	Win
SVM (RBF)	Loss	Draw	Draw	Loss	Draw	Draw	Win	Draw	Draw	Draw	Draw	Win

improvements over all other considered k-NN techniques. Against ROC-tree, *ROC-kNN* significantly improves for the non-gene expression datasets, but performs mostly the same on gene expression data. Against the two SVM classifiers, *ROC-kNN* no worse and indeed actually improves for two of the six considered non-gene expression datasets. However, on gene expression data, the improvement is more marginal.

6 Conclusion

This paper has presented a new k-NN style algorithm for classification, based on a new method for defining a weighted distance function. Our method is based on considering the ROC characteristics of each feature, within a neighbourhood appropriate to the instances whose distance is being computed.

Experimental analysis shows our technique, *ROC-kNN*, is able to substantially improve over basic k-NN. Furthermore, it performs very strongly compared to other state-of-the-art classifiers and is even able to deliver improved accuracy in many cases.

For future work, we would like to consider extending our approach i) to incorporate pruning of irrelevant features and ii) to situations where there are three or more classes in the data, a well known challenge for ROC computation.

Acknowledgements. This work is partially supported by National ICT Australia. National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative in part through the Australian Research Council.

References

1. Theilhaber, J., et al.: Finding genes in the C2C12 osteogenic pathway by k-nearest-neighbor classification of expression data. Genome Research 12(1), 165–176 (2002)
2. Wu, W., Xing, E., Mian, I., Bissell, M.: Evaluation of normalization methods for cDNA microarray data by k-NN classification. BMC Bioinformatics 6(191), 1–21 (2005)
3. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest-neighbor classification. IEEE Transactions on Pattern Analysis Machine Intelligence 18(6), 607–616 (1996)

4. Green, D.M., Swets, J.M.: Signal detection theory and psychophysics. John Wiley & Sons Inc., New York (1966)
5. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143, 29–36 (1982)
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
7. Kira, K., Rendell, L.: A practical approach to feature selection. In: Proc. of ICML, pp. 249–256 (1992)
8. Salzberg, S.: Distance metrics for instance-based Learning. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1991. LNCS, vol. 542, pp. 399–408. Springer, Heidelberg (1991)
9. Kononenko, I.: Estimating attributes: Analysis and extensions of Relief. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994)
10. Klein, D., Kamvar, S.D., Manning, C.D.: From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Proc. of ICML, pp. 307–314 (2002)
11. Stein, B., Niggemann, O.: Generation of similarity measures from different sources. In: 14th International Conference on Industrial Engineering Applications of Artificial Intelligence and Expert Systems, pp. 197–206 (2001)
12. Han, E.H., Karypis, G., Kumar, V.: Text categorization using weight-adjusted nearest-neighbor classification. In: Conference on Knowledge Discovery and Data Mining, Hong Kong, China, pp. 53–65 (2001)
13. Zhang, Z.: Learning metrics via discriminant kernels and multidimensional scaling: toward expected Euclidean representation. In: Proc. of ICML, pp. 872–879 (2003)
14. Hastie, T., Tibshirani, R.: Discriminant Adaptive Nearest Neighbor Classification and Regression. In: Advances in Neural Information Processing Systems, vol. 8, pp. 409–415 (1996)
15. Domeniconi, C., Gunopulos, D., Peng, J.: Large margin nearest neighbor classifiers. IEEE Transactions on Neural Networks 16(4), 899–909 (2005)
16. Driessens, K., Reutemann, P., Pfahringer, B., Leschi, C.: Using Weighted Nearest Neighbor to Benefit from Unlabeled Data. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 60–69. Springer, Heidelberg (2006)
17. Im, K.H., Park, S.C.: Case-based reasoning and neural network based expert system for personalization. Expert Systems with Applications 32, 77–85 (2007)
18. Vivencio, D.P., et al.: Feature-weighted k-Nearest Neighbor Classifier. In: Proc. of FOCI, pp. 481–486 (2007)
19. Hossain, M.M., Hassan, M.R., Bailey, J.: ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data. In: Proc. of 8th SIAM Int'l Conf. on Data Mining (SDM 2008), pp. 455–465 (2008)
20. Ferri, C., Flach, P., Hernández-Orallo, J.: Learning decision trees using the area under the ROC curve. In: Proc. of ICML, pp. 139–146 (2002)
21. Deng, L., Pei, J., Ma, J., Lee, D.L.: A Rank Sum Test Method for Informative Gene Discovery. In: Proc. of KDD, pp. 410–419 (2004)
22. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
23. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers. Technical Report MS 1143, HP Laboratories (2004)

24. Egan, J.P.: Signal Detection Theory and ROC Analysis. Academic Press Series in Cognition and Perception. Academic Press, London (1975)
25. Mamitsuka, H.: Selecting features in microarray classification using ROC curves. Pattern Recognition 39(12), 2393–2404 (2006)
26. van 't Veer, L.J., et al.: Gene expression profiling predicts clinical outcome of breast cancer. Nature 415, 530–535 (2002)
27. Integrated Tumor Transcriptome Array and Clinical Data Analysis (2006), http://bioinfo-out.curie.fr/ittaca
28. Hedenfalk, I., et al.: Gene-expression profiles in hereditary breast cancer. The New England Journal of Medicine 344(8), 539–548 (2001)
29. Gordon, G.J., et al.: Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer And Mesothelioma. Cancer Research 62, 4963–4967 (2002)
30. The Division of Thoracic Surgery (2002), http://www.chestsurg.org/publications/2002-microarray.aspx
31. Broad Institute Cancer Program (2002), http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=75
32. Singh, D., et al.: Gene Expression Correlates of Clinical Prostate Cancer Behavior. Cancer Cell 1, 203–209 (2002)
33. Broad Institute Cancer Program (1999), http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43
34. Golub, T.R., et al.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, 531–537 (1999)
35. Newman, D., et al.: UCI Repository of machine learning databases. Online Repository (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
36. Perner, P.: Methods for Data Mining. In: Data Mining on Multimedia Data. LNCS, vol. 2558, pp. 23–89. Springer, Heidelberg (2002)
37. Glantz, S.A.: Primer of BioStatistics, pp. 309–310. McGraw-Hill, NY (1992)

One-Class Classification by Combining Density and Class Probability Estimation

Kathryn Hempstalk, Eibe Frank, and Ian H. Witten

Department of Computer Science, University of Waikato, Hamilton, NZ
{kah18,eibe,ihw}@cs.waikato.ac.nz

Abstract. One-class classification has important applications such as outlier and novelty detection. It is commonly tackled using density estimation techniques or by adapting a standard classification algorithm to the problem of carving out a decision boundary that describes the location of the target data. In this paper we investigate a simple method for one-class classification that combines the application of a density estimator, used to form a reference distribution, with the induction of a standard model for class probability estimation. In this method, the reference distribution is used to generate artificial data that is employed to form a second, artificial class. In conjunction with the target class, this artificial class is the basis for a standard two-class learning problem. We explain how the density function of the reference distribution can be combined with the class probability estimates obtained in this way to form an adjusted estimate of the density function of the target class. Using UCI datasets, and data from a typist recognition problem, we show that the combined model, consisting of both a density estimator and a class probability estimator, can improve on using either component technique alone when used for one-class classification. We also compare the method to one-class classification using support vector machines.

1 Introduction

In most classification problems, training data is available for all classes of instances that can occur at prediction time. In this case, the learning algorithm can use the training data for the different classes to determine decision boundaries that discriminate between these classes. However, there are some problems that are suited to machine learning, but exhibit only a single class of instances at training time. At prediction time, new instances with unknown class labels can either belong to this target class or a new class that was not available during training. In this scenario, two different predictions are possible: *target*, meaning an instance belongs to the class learned during training, and *unknown*, where the instance does not appear to belong to the previously learned class. This type of learning problem is known as one-class classification.

In many cases, it may seem sensible to suggest that one-class problems should be reformulated into two-class ones because there is actually data from other classes that can be used for training. However, there are genuine one-class applications where it is inappropriate to make use of negative data during training.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 505–519, 2008.

For example, consider password hardening, which is a biometric system that strengthens the login process on a computer by not only requiring the correct password to be typed, but also requiring it to be typed with the correct rhythm. Password hardening is clearly a one-class problem; a single user must be verified and during training time only data from that user is available—we cannot ask anyone else to provide data without supplying them with the password.

Even in applications where instances from multiple classes are available at training time, it may be opportune to focus solely on the target class under consideration and contemplate a one-class set-up. In these applications, new classes may occur at prediction time that are different from *all* classes available during the training process. This is the case in the continuous typist recognition problem that motivated the work presented in this paper. Continuous typist recognition is similar to password hardening, only the text underlying the patterns is not fixed: the current typist is verified on a block of free text. In this situation we rely on one-class classification because we need to be able to refuse an attacker the system has never seen before.

One-class classification is often called outlier (or novelty) detection because the learning algorithm is being used to differentiate between data that appears normal and abnormal with respect to the distribution of the training data. A common statistical approach to this view on one-class classification is to identify outliers as instances that are greater than a distance, d, to a percentage, p, of the training data [2,9]. Machine learning techniques that have been employed include clustering the data and determining a suitable boundary that encloses all the clusters [16], adapting kernel-based support vector machine classifiers [12], and utilising densities to estimate target class membership [13].

In this paper we investigate a principled approach for applying two-class classification algorithms to one-class classification. The only requirement is that these algorithms can produce class probability estimates at prediction time. This is not an impediment in general because most algorithms either provide these estimates directly or can be modified to do so. The basic method we use to apply a supervised approach to an unsupervised learning problem is not new: it is described by Hastie *et al.* [6] in the context of association rule learning. They also mention that "Although this approach... seems to have been part of the statistics folklore for some time, it does not appear to have had much impact despite its potential".

The technique we explore is based on the generation of artificial data that comes from a known reference distribution such as a multi-variate normal distribution, which can be estimated from the training data for the target class. This artificial data takes the role of a second class in the construction of a two-class classification model. Using Bayes' rule, we show how the density function of the reference distribution can be combined with the class probability estimates of this classification model to yield a description of the target class. We show that the combined model, which employs both the density function and the classification model, can yield improved performance when compared to using the density function alone for one-class classification; the latter being a standard

technique that has been used for one-class classification in the past [13,14]. It also improves on using the classification model alone. The resulting method yields numeric membership scores that can be used to rank test instances according to their predicted likelihood of belonging to the target class. This property can be an advantage when compared to techniques that provide a single decision boundary, because it makes it possible to adjust the trade-off between false positives and false negatives at prediction time.

The next section discusses previous approaches to one-class classification. Following this, we explain the approach to one-class classification that we evaluate in this paper. In Sections 4 and 5 we present the evaluation of this technique, on both standard UCI datasets and our continuous typist recognition data, and compare it to existing approaches. The final section concludes the paper and proposes some future work.

2 Related Work

Existing models for one-class classification either extend current methods for multi-class classification or are based on density estimation. In the latter approach, density estimation is performed by fitting a statistical distribution, such as a Gaussian, to the target data; any instances with a low probability of appearing (more precisely, low density value) can be marked as outliers [9]. This is a sensible approach in cases where the target data follows the selected distribution very closely. The challenge is to identify an appropriate distribution for the data at hand. Alternatively one can use a non-parametric approach, such as kernel density estimation, but this can be problematic because of the curse-of-dimensionality and the resulting computational complexity.

Extensions of current multi-class classifiers to one-class classification involve fitting a boundary around the target data, such that instances that fall outside the boundary are considered outliers. The boundary can be generated by adapting the inner workings of an existing multi-class classifier [12], or by using artificial data as a second class, in conjunction with a standard multi-class learning technique [1]. Methods in the former category generally rely heavily on a parameter that defines how much of the target data is likely to be classified as outlier [14]. This parameter defines how conservative the boundary around the target class will be. If it is chosen too liberally, then the model will overfit and we risk identifying too much legitimate target data as outliers. A drawback of these techniques is that an appropriate parameter value needs to be manually chosen at training time.

In contrast, when density estimation is used for one-class classification, a threshold on the density can be adjusted at prediction time to obtain a suitable rate of outliers. In some situations, where parametric density estimation fails, using classification-based methods may be favourable; these techniques are generally able to define boundaries on data that cannot be tightly modelled by a standard statistical distribution. In some cases there is a close link between classification-based techniques and density estimators: for example, it has been

shown that one-class kernel Fisher discriminant classifiers can be used to perform non-parametric density estimation [11]. However, this only applies to very specific learning techniques.

The method presented in this paper is based on the generation of artificial data from a reference distribution to form a two-class classification problem. However, unlike earlier work on using artificial data for one-class classification (see [1] and references therein), it is based on using a two-class probability estimator, and combines the estimated reference density function with the resulting class probability estimator to form an overall prediction.

The method is generic in the sense that it is applicable in conjunction with an arbitrary density estimator and an arbitrary class probability estimation technique. An alternative generic approach is ensemble learning, where the predictions of different one-class classifiers are combined in an ensemble [15]. In contrast, we combine a density estimator and a class probability estimator to form a single one-class classifier, along the lines of the generic approach to unsupervised learning outlined in [6].

3 Combining Density Functions and Class Probability Estimators

Given the large number of classification algorithms that have been developed, it would be useful to be able to utilize them for one-class problems. A possible approach for doing this is to generate artificial data to take the role of the second class. The most straightforward method for implementing this is to generate uniformly distributed data and learn a classifier that can discriminate this data from the target. A problem with this method is that different decision boundaries are obtained for different amounts of artificial data: if too much artificial data is generated, then the target class will be overwhelmed by it, and, assuming the learning algorithm aims to minimize classification error, it will learn to always predict the artificial class. However, this problem can be avoided when the objective of learning is viewed as accurate class probability estimation rather than minimization of classification error, and a suitable configuration of the learning algorithm is chosen. An example of a suitable inductive approach is bagging of unpruned decision trees, which has been shown to yield good class probability estimators [10].

Once a class probability estimation model has been obtained in this fashion, different thresholds on the resulting class probability estimates for the target class correspond to different decision boundaries surrounding the instances of the target class. This means that, as in the density estimation approach to one-class classification, the rate of obtaining "outliers" can be adjusted at prediction time to yield an outcome appropriate for the application at hand.

There is one significant problem with the approach that we have just described: as the number of attributes in the learning problem increases (i.e. as the dimensionality of the instance space grows), it quickly becomes infeasible to generate enough artificial data to obtain sufficient coverage of the instance space,

and the probability that a particular artificial instance occurs inside or close to the target class diminishes to a point that makes any kind of discrimination impossible.

The solution to this problem is to generate artificial data that is as close as possible to the target class. In this case, because the data is no longer uniformly distributed, it becomes necessary to take the distribution of this artificial data into account when computing the membership scores for the resulting one-class model. As in [6], we call the distribution that is used to generate the artificial data the "reference" distribution. In the following we explain how the class probability estimates of the two-class classifier are combined with the values of the density function of this reference distribution to obtain membership scores for the target class.

Let T denote the target class for which we want to build a one-class model. We have training data for this class. Let A be the artificial class, for which we generate artificial data using a known reference distribution. Let X denote an instance and let $P(X|A)$ denote the density function of the reference distribution.

What we would like to obtain is $P(X|T)$, the density function for the target class. If we had this density function, we could use it for one-class classification by imposing a threshold on its values. Let us assume for the moment that we know the true class probability function $P(T|X)$. In practice, we need to estimate this function using a class probability estimator learned from the training data.

The following shows how we can compute the density function for T, namely $P(X|T)$, given the class probability function $P(T|X)$, the reference density $P(X|A)$, and $P(T)$, which is the prior probability of observing an instance of the target class. We start with Bayes' theorem:

$$P(T|X) = \frac{P(X|T)P(T)}{P(X)}$$

For a two-class situation, the probability of X is the probability of seeing an instance of X with either class label, so the equation becomes:

$$P(T|X) = \frac{P(X|T)P(T)}{P(X|T)P(T) + P(X|A)P(A)}$$

Now we solve for $P(X|T)$, the density function for the target class, which we want to use for one-class classification. We first bring the denominator on the right to the left:

$$(P(X|T)P(T) + P(X|A)P(A))P(T|X) = P(X|T)P(T)$$

Now we expand the product on the left, and bring the term involving $P(X|T)$ to the right:

$$P(X|T)P(T)P(T|X) + P(X|A)P(A)P(T|X) = P(X|T)P(T)$$

$$P(X|A)P(A)P(T|X) = P(X|T)P(T) - P(X|T)P(T)P(T|X)$$

Then we extract out $P(X|T)$ and bring the remainder to the left:

$$P(X|A)P(A)P(T|X) = P(X|T)(P(T) - P(T)P(T|X))$$

$$\frac{P(X|A)P(A)P(T|X)}{P(T) - P(T)P(T|X)} = P(X|T)$$

We swap the two sides and extract $P(T)$ in the denominator:

$$P(X|T) = \frac{P(X|A)P(A)P(T|X)}{P(T)(1 - P(T|X)}$$

Now we make use of the fact that $P(A) = 1 - P(T)$, because there are only two classes, and rearrange:

$$P(X|T) = \frac{(1 - P(T))P(T|X)}{P(T)(1 - P(T|X))}P(X|A) \tag{1}$$

This equation relates the density of the artificial class $P(X|A)$ to the density of the target class $P(X|T)$ via the class probability function $P(T|X)$ and the prior probability of the target class $P(T)$.

To use this equation in practice, we choose $P(X|A)$ and generate a user-specified amount of artificial data from it. Each instance in this data receives the class label A. Each instance in the training set for the target class receives class label T. Those two sets of labeled instances are then combined. The proportion of instances belonging to T in this combined dataset is an estimate of $P(T)$, and we can apply a learning algorithm to this two-class dataset to obtain a class probability estimator that takes the role of $P(T|X)$. Assuming we know how to compute the value for $P(X|A)$ given any particular instance X—and we can make sure that this is the case by choosing an appropriate function—we then have all the components to compute an estimate of the target density function $\hat{P}(X|T)$ for any instance X.

Note that, because we are using estimates for $P(T|X)$ and $P(T)$, the function $\hat{P}(X|T)$ will not normally integrate to one and is thus not a proper density. However, this is not a problem for one-class classification because we can empirically choose an appropriate threshold on $\hat{P}(X|T)$ to perform classification, and we can adjust this threshold to tune the probability of an instance being identified as an outlier. Obviously, we can also use this function to rank instances. Note that this also means the prior odds ratio $\frac{(1-P(T))}{P(T)}$ is irrelevant in practice, because it corresponds to a constant factor, and it is sufficient to use an estimate of the posterior odds $\frac{P(T|X)}{1-P(T|X)}$ in conjunction with $P(X|A)$.

It is instructive to relate this method to the simple approach discussed at the beginning of this section, which was based on generating uniformly distributed data for the artificial class. This corresponds to using the uniform distribution as the reference distribution, i.e. using the uniform density for $P(X|A)$. Based on Equation 1, this means that $P(X|T)$ becomes proportional to the posterior odds ratio $P(T|X)/(1-P(T|X))$—which is monotonic in $P(T|X)$—meaning that test

cases can simply be ranked according to the class probability function $P(T|X)$, and classification can be performed based on an appropriately chosen threshold for this function. This means the two-class classification model that is learned from the target data and the uniformly distributed data can directly be used for one-class classification; for the reason given above, this is only a worthwhile option to consider for very low-dimensional learning problems.

One question remains, namely how to choose the reference density $P(X|A)$. Essential requirements for this function are that (a) we need to be able to generate artificial data from it and (b) we need to be able to compute its value for any instance X. Another requirement, as indicated at the beginning of this section, is that the data generated based on this distribution is close to the target class. In fact, based on Equation 1 we now know that, ideally, the reference density is identical to the target density, in which case $P(T|X)$ becomes a constant function that any well-behaved learning algorithm should be able to induce (i.e. the resulting two-class learning problem would become trivial). This is obviously not realistic because it would essentially require us to know (or somehow obtain) the density of the target class. However, this observation gives us a clue as to how we can go about getting a useful density function $P(X|A)$ for the problem at hand: we can apply *any* density estimation technique to the target data for class T and use the resulting density function to model the artificial class A. The more accurately this initial density estimate models $P(X|T)$, i.e. the better the match between $P(X|A)$ and $P(X|T)$, the easier the resulting two-class class probability estimation task should become. This discussion implies that Equation 1 can also be viewed as a mechanism for improving an initial density estimate for a dataset using a class probability estimation technique.

In practice, given the availability of powerful methods for class probability estimation, and the relative lack of such techniques for density estimation, it makes sense to apply a simple density estimation technique to the target data first, to obtain $P(X|A)$, and then employ a state-of-the-art class probability estimation method to the two-class problem that is obtained by joining the artificial data generated using $P(X|A)$ and the data from the target class. This is the strategy that we evaluate in this paper.

Because Equation 1 is used to estimate a density function for the target class, the method presented here is most closely related to the standard density-estimation-based approach to one-class classification. However, given that a class probability estimation model is used as part of the estimation process, which is obtained from a standard technique for classification learning, the method is also related to approaches that adapt standard two-class learning techniques to the one-class learning task: in a sense, it straddles the boundary between these two groups of approaches to one-class classification.

4 Evaluation Method

Our primary motivation for exploring the one-class learning technique presented in this paper was our interest in the domain of continuous typist recognition.

In the next section we present empirical results on datasets from this domain. However, we also present results for standard multi-class UCI datasets.

Evaluating one-class classifiers on datasets with multiple classes is straightforward. Each class in turn is treated as the target class and the other classes are joined into an "outlier" class. In our experiments, we ran a standard stratified 10-fold cross-validation, repeated 10 times, to estimate the AUC for a particular target class. The one-class learning methods simply ignore the data for the outlier class that occurs in the training sets. The instances in the test sets are ranked according to their predicted density values so that the AUC can be computed. In this fashion we obtain one AUC value for each class in a dataset. To summarize performance for a particular UCI dataset, we report a weighted average of these AUC values. For the weighted AUC, each two-class AUC is calculated, then weighted by the prevalence of the target class and summed. The formula for the weighted AUC is:

$$AUC_{weighted} = \sum_{\forall c_i \in C} AUC(c_i) \times p(c_i) \tag{2}$$

where C is the full set of classes in the dataset, $p(c_i)$ is the prevalence of the target class c_i in the full dataset, and $AUC(c_i)$ is the AUC value for target class c_i. Using a weighted average rather than an unweighted one prevents target classes with smaller instance counts from adversely affecting the results.

The one-class method presented in this paper combines the output of a density estimator with that of a class probability estimator. In our evaluation we used bagged unpruned C4.5 decision trees with Laplace smoothing as the probability estimator $P(T|X)$. Ten bagging iterations were used throughout. We evaluated two different simple density estimation models: a Gaussian density with a diagonal co-variance matrix containing the observed variance of each attribute in the target class, and a product of mixture of Gaussian distributions with one mixture per attribute. Each mixture is fitted to the target data for its corresponding attribute using the EM algorithm. The amount of artificial data generated using the reference distribution, which determines the estimate of $P(T)$ in Equation 1, was set to the size of the target class. Hence the data used to build the bagged unpruned decision trees was exactly balanced.

One of the main objectives of our empirical study was to ascertain that the combined model can indeed improve on its components in terms of predictive performance. Hence we also evaluated one-class classification using the Gaussian density and the EM-based density directly, with the same cross-validation-based experimental set-up described above. This means we use the reference density $P(X|A)$ from Equation 1 to rank test instances, rather than $\hat{P}(X|T)$. Additionally, we also measured the performance obtained when using only the class probability estimator from Equation 1, i.e. the bagged unpruned decision trees in our experiments. This means we only use the estimate for $P(T|X)$ from Equation 1 for ranking. Note that in this case the reference density is still used to generate the data for the artificial second class.[1]

[1] We also tried using uniformly distributed data as the artificial class, but the results were poor, so they are not shown in what follows.

As an optimistic baseline we also show the performance obtained on the UCI datasets when treating the learning problem as a standard classification problem, using the data for all classes at training time, by building a one-vs-rest model using bagged decision trees.

For completeness, we also compare to the one-class support vector machine described by [12], as it is implemented in libSVM [3]. We used RBF kernels and set the value of ν to 0.1. The parameter ν determines how much of the target data is likely to be classified as outlier. We adjusted the γ value for the RBF kernel for each dataset to obtain a false alarm rate (FAR) as close as possible to 0.1. The false alarm rate is the number of legitimate target instances incorrectly identified as outliers (also known as the false negative rate).

When comparing libSVM to the combined one-class classifier we cannot use AUC because the one-class implementation in libSVM does not return membership scores, just a yes/no decision. Hence our comparison is based on attempting to achieve a fixed FAR, namely 0.1, for both techniques, by choosing an appropriate threshold for our model, and evaluating the corresponding impostor pass rate (IPR). The impostor pass rate is the number of outlier instances that are wrongly classified as belonging to the target class (also known as the false positive rate). FAR and IPR are often used in domains such as biometrics. A higher FAR results in a lower IPR and vice versa. Note that, to calculate FAR and IPR in our experiments, false negatives and false positives were simply accumulated across all one-class learning problems that resulted from processing a multi-class dataset. As in the case of AUC, 10-fold cross-validation, repeated 10 times, was used for a single one-class learning problem.

5 Results

In the following we present the experimental results obtained using the methodology described above. We first discuss the performance of the combined classifier, its components, and the baseline multi-class classifier on standard multi-class UCI datasets. In the second subsection we introduce the typist dataset, which motivated this work, and use it to show the performance of the combined classifier on individual classes. Finally, we present a comparison between the combined one-class classifier and the one-class support vector machine [3,12].

5.1 UCI Datasets

Table 1 contains the results we obtained for the UCI datasets, comparing weighted AUC values for the baseline multi-class classifier, in the left-most column, and the variants of one-class classifiers discussed above, excluding the SVM. More specifically, there are two groups of three columns for the one-class classifiers. The first group is based on using the Gaussian density as the reference density and the second one based on using the EM method. Each group of

three columns has results for (a) the combined classifier, consisting of both the reference density and the class probability estimation model (i.e. bagged trees), (b) the reference density $P(X|A)$ used directly, and (c) the class probability estimator $P(T|X)$ used directly.

From the results presented in Table 1 it is clear that the combined classifier performs much better than its component probability estimator—i.e. the estimate of $P(T|X)$—for all of the UCI datasets. Hence it is essential to take the reference density into account when making a prediction. It is not sufficient to simply generate artificial data based on the reference density, build a class probability estimator, and then use that for prediction.

The picture is not so clear when comparing the combined classifier to the reference density. Considering the Gaussian case, there are four datasets where the latter produces better results: diabetes, ecoli, iris, and waveform-5000. This indicates that the combined model is too complex in those cases; the simple reference distribution is sufficient to model the distribution of the target class for those datasets, and adding bagged trees into the model is detrimental. On the other hand, there are six datasets where the combined model performs better: heart-statlog, letter, mfeat-karhunen, pendigits, sonar, and vehicle. In the case of the vehicle and letter datasets the difference is substantial—and is so even for the AUC values of every individual class label occurring in these datasets (for brevity these results are not shown here).

Considering the case of using the EM-based density estimate as the reference density, the overall picture is similar, but there is only one tie in this case (sonar). There are eight wins for the combined classifier (heart-statlog, letter, the three mfeat datasets, pendigits, vehicle, and waveform-5000) and six losses (diabetes, ecoli, glass, ionosphere, iris, and pendigits). The biggest wins in absolute terms for the combined method occur again on the vehicle and letter datasets.

It is instructive to compare the performance of the two different combined models. The EM-based model wins on nine datasets and loses on only five. Hence the combined model often receives a boost when the more powerful EM-based density estimator is used to obtain the reference density. Note also that the EM-based density often has an edge compared to using the Gaussian density when both are used in stand-alone mode: the former wins nine times and loses six times.

Not surprisingly, standard bagged decision trees outperform all one-class classifier variants on all datasets, often by a significant margin. The reason for this is that this learner actually gets to see data for the outlier class (i.e. the union of the non-target classes) at training time. It can thus focus on discriminating the target class against all those specific non-target classes, and its performance can be considered an optimistic target for that of corresponding one-class classifiers. We would like to emphasize here that in practical applications of one-class classifiers this kind of data is not available. Even if there is some negative data available at training time, the expectation is that completely new classes of data will be encountered at prediction time.

Table 1. Weighted AUC results on UCI data, for standard bagged decision trees, the combined one-class classifier, and its components

	Bagged	One-class Classifier Gaussian			One-class Classifier EM		
Dataset	Trees	Combined	$P(X\|A)$	$P(T\|X)$	Combined	$P(X\|A)$	$P(T\|X)$
diabetes	0.818	0.626	0.653	0.511	0.639	0.669	0.510
ecoli	0.953	0.928	0.930	0.516	0.928	0.931	0.542
glass	0.878	0.698	0.698	0.624	0.731	0.735	0.593
heart-statlog	0.880	0.796	0.790	0.671	0.768	0.751	0.629
ionosphere	0.965	0.697	0.697	0.587	0.726	0.727	0.580
iris	0.985	0.974	0.977	0.628	0.972	0.976	0.662
letter	0.996	0.904	0.887	0.701	0.931	0.921	0.783
mfeat-karhunen	0.984	0.957	0.955	0.524	0.960	0.959	0.577
mfeat-morphological	0.959	0.941	0.941	0.804	0.940	0.939	0.836
mfeat-zernike	0.959	0.898	0.898	0.418	0.904	0.902	0.622
optdigits	0.995	0.959	0.959	0.562	0.954	0.955	0.645
pendigits	0.998	0.958	0.953	0.845	0.938	0.933	0.821
sonar	0.867	0.588	0.587	0.484	0.612	0.612	0.501
vehicle	0.919	0.705	0.657	0.656	0.781	0.765	0.700
waveform-5000	0.951	0.863	0.864	0.415	0.864	0.863	0.466

5.2 Typist Dataset

The work in this paper was motivated by the need to find an appropriate classification method for a continuous typist recognition problem. As mentioned previously, continuous typist recognition is akin to password hardening, only the patterns presented to the system may contain any number of characters and sample lengths may vary. During our search for a method for improving the state-of-the-art in this research area, we investigated several different one-class classifiers—all of which were customised to the task of typist recognition [4,5,7,8]. We felt that this problem would benefit from the extensive research performed on multi-class classifiers and directed our efforts towards creating a dataset that could be used by standard machine learning algorithms.

Unfortunately, with the ethical issues surrounding key logging data, and the omission of key release events from one of the datasets we had access to, we were unable to transform an existing dataset for use in our experiments. Instead, we recorded 3000 emails from 19 people in the Computer Science Department at the University of Waikato over a period of 3 months. After technical and ethical issues were addressed, a dataset of 15 emails for each of 10 participants was created. This dataset only contained the raw sequences of input from the recordings. Each sample was broken down further into blocks of 400 events—a size that roughly equates to a small paragraph of text and is similar in size to other typist datasets [5]—resulting in between 24 and 75 samples per user. Every 400-event sample had a number of attributes calculated; these attributes formed the dataset for use with a standard machine learning algorithm.

Table 2. AUC results for the typist dataset, for standard bagged decision trees, the combined one-class classifier, and its components

	Bagged	One-class Classifier Gaussian			One-class Classifier EM		
Participant	Trees	Combined	$P(X\|A)$	$P(T\|X)$	Combined	$P(X\|A)$	$P(T\|X)$
A	0.938	0.924	0.932	0.312	0.923	0.931	0.326
B	0.970	0.934	0.931	0.365	0.929	0.930	0.433
C	0.915	0.707	0.665	0.677	0.786	0.788	0.594
D	0.958	0.924	0.928	0.335	0.902	0.916	0.456
E	0.994	0.973	0.974	0.795	0.971	0.971	0.793
F	0.913	0.852	0.843	0.619	0.867	0.862	0.636
G	0.962	0.942	0.943	0.367	0.952	0.951	0.418
H	0.892	0.909	0.909	0.613	0.914	0.913	0.618
I	0.939	0.956	0.958	0.591	0.950	0.949	0.449
J	0.975	1.000	1.000	0.792	1.000	1.000	0.747
Weighted Avg.	0.941	0.897	0.891	0.540	0.908	0.910	0.547

In total, there are 8 attributes in our typist dataset.[2] Most of the attributes are based around the typist speed (average words-per-minute (WPM) rate, peak WPM, trough WPM) or error rate (backspaces, paired backspaces, average backspace block length). There are also two attributes that relate to the slurring of key press and release events (press/release ordering, press/release rate). The final typist dataset used here contains these 8 attributes and 10 class labels (Participants A–J).[3]

Table 2 shows the results obtained for the 10 different typist classes, in each case treating one of these classes as the target class, and the union of the other classes as the outlier class. Each row states AUC values for one particular target class. The bottom row has the weighted AUC value, calculated according to Equation 2.

The results from the typist dataset are similar to those from the UCI datasets: using only the class probability estimator, $P(T|X)$, results in poor performance, whereas using just the reference density, $P(X|A)$, sometimes performs better than the combined model. More specifically, considering the case of the Gaussian reference density, the win/loss ratio for the combined model vs. the reference density is 3/5; considering the case of the EM-based density it is 4/4. According to overall weighted AUC, the combined model has an edge in the Gaussian case, but is outperformed slightly in the case of EM.

Considering performance relative to the baseline multi-class classifier, the overall picture is similar as in the case of the UCI datasets: multi-class classification outperforms one-class classification. However, surprisingly, for three of the user classes—H, I and J—the combined one-class classifier and the reference densities have a higher AUC than the baseline classifier. This is not unusual; we experienced similar results on individual class labels on many of the UCI datasets. In the context of our target application, typist recognition, this is

[2] This number is likely to change in future for the purposes of typist recognition.

[3] Available for download at http://www.cs.waikato.ac.nz/ml/data/typist.arff

Table 3. Results for one-class support vector machines (obtained with libSVM) versus the combined one-class classifier (OCC)

	libSVM			OCC Gaussian		OCC EM	
Dataset	γ	FAR	IPR	FAR	IPR	FAR	IPR
diabetes	0.00005	0.111	0.514	0.098	0.857	0.109	0.779
ecoli	0.1	0.137	0.068	0.129	0.088	0.136	0.083
glass	0.005	0.154	0.412	0.147	0.434	0.180	0.331
heart-statlog	0.0001	0.122	0.624	0.140	0.507	0.141	0.504
ionosphere	0.00005	0.128	0.738	0.150	0.732	0.169	0.697
iris	0.0005	0.120	0.073	0.125	0.076	0.137	0.077
letter	0.000005	0.101	0.516	0.100	0.291	0.105	0.215
mfeat-karhunen	0.0001	0.131	0.034	0.094	0.114	0.105	0.092
mfeat-morphological	0.0000001	0.110	0.206	0.068	0.128	0.075	0.134
mfeat-zernike	0.000001	0.116	0.253	0.085	0.324	0.099	0.276
optdigits	0.00005	0.106	0.087	0.107	0.084	0.122	0.087
pendigits	0.000001	0.103	0.203	0.100	0.116	0.102	0.137
sonar	0.001	0.120	0.705	0.123	0.815	0.163	0.751
vehicle	0.00005	0.103	0.629	0.109	0.645	0.130	0.494
waveform-5000	0.001	0.103	0.307	0.075	0.411	0.110	0.354
typist	0.00005	0.113	0.331	0.113	0.204	0.147	0.157

exciting because it demonstrates that one-class classification can be a very satisfactory substitute for multi-class classifiers, and one that is better suited to solving the problem at hand because it does not require any training data from the outlier classes in order to achieve good results: when using typist recognition for computer security we simply do not have access to training data for new classes of typists that correspond to impostors (or "attackers").

5.3 LibSVM

To compare the method presented in this paper—again, using bagged unpruned decision trees—to an established one-class classifier, we now discuss results obtained in a comparison to the one-class classifier in libSVM [3]. The results are shown in Table 3. As mentioned in Section 4, we could not use AUC for comparison, and instead resort to reporting FAR and IPR. For each dataset, the table reports the value of the γ parameter for the RBF kernel that was used to obtain the results shown.

Although we were generally unable to match FAR exactly for the methods compared, the results nevertheless enable us to make a qualitative statement about their relative performance. In particular, we can consider cases where *both* FAR and IPR are lower for one method in a pair of methods being compared. Doing this for libSVM and the Gaussian-based combined classifier, we see that there are two datasets where both FAR and IPR are lower for the latter method (letter and pendigits) and two cases where they are both lower for libSVM (sonar

and vehicle). Repeating this exercise for libSVM and the combined model with the EM-based density, we find that there is one dataset where the latter method is better for both statistics (pendigits) and three datasets where this is the case for libSVM (iris, sonar, and waveform-5000). Overall, we can say that there are datasets where the SVM appears to be the better approach and other datasets where the combined method appears to perform better. Note that, in contrast to the SVM, FAR and IPR can be adjusted at prediction time with the approach presented here, and it does not require parameter tuning to return satisfactory results.

6 Conclusion

In this paper we have combined density estimation with class probability estimation for the purpose of one-class classification. We applied a density estimator to build a reference density for the target class, then used this reference density to generate artificial data for a two-class learning problem suitable for a class probability estimation technique, and finally combined the predictions of the reference density and the class probability model to form predictions for new test cases.

Using experimental results obtained on UCI datasets and a continuous typist recognition problem, and using bagged unpruned decision trees as the underlying class probability estimator, we have shown that the combined model can indeed improve on both component techniques: the density estimator used to obtain the reference density and the class probability estimator trained on the semi-artificial two-class learning problem.

We have also compared the combined model to a one-class support vector machine with a RBF kernel. The results show that there are datasets where the former method is superior and other datasets where the one-class SVM performs better. The combined method has the advantage that—like in standard density estimation techniques for one-class classification— there is no need to specify a target rejection rate at training time.

A significant feature of the method explored here is that it is generic, and hence can be used in conjunction with arbitrary density estimators and class probability estimation techniques. We believe that this is the primary advantage of this technique, given the availability of large collections of suitable candidate base learners in machine learning workbenches (see also the relevant discussion in [6]).

An interesting avenue for future work is the experimental comparison of variants of the combined technique that can be obtained by plugging in different types of base learners. Another important question is how the quantity of artificial data that is generated using the reference distribution influences the result. There must clearly be diminishing returns, but one would expect that, in general, more data leads to a more accurate combined model. On a more fundamental level, it would be interesting to investigate whether it is possible to avoid the artificial data generation step by adapting class probability estimators

to take advantage of information in a reference distribution directly. However, it is obvious that this cannot be achieved without changing the learning algorithm involved; thus the generic aspect of the method would be lost.

References

1. Abe, N., Zadrozny, B., Langford, J.: Outlier detection by active learning. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 767–772. ACM Press, New York (2006)
2. Barnett, V., Lewis, T.: Outliers in Statistical Data. John Wiley & Sons, West Sussex (1994)
3. Chang, C., Lin, C.: LIBSVM: A Library for Support Vector Machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. Dowland, P., Furnell, S., Papadaki, M.: Keystroke analysis as a method of advanced user authentication and response. In: Proceedings of the IFIP TC11 17th International Conference on Information Security, Deventer, The Netherlands, pp. 215–226. Kluwer, Dordrecht (2002)
5. Gunetti, D., Picardi, C.: Keystroke analysis of free text. ACM Transactions on Information and System Security 8(3), 312–347 (2005)
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
7. Monrose, F., Rubin, A.: Keystroke dynamics as a biometric for authentication. In: Future Generation Computer Systems, vol. 16, pp. 351–359. Elsevier Science, Amsterdam (2000)
8. Nisenson, M., Yariv, I., El-Yaniv, R., Meir, R.: Towards behaviometric security systems: Learning to identify a typist. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 363–374. Springer, Heidelberg (2003)
9. Pearson, R.: Mining Imperfect Data. Society for Industrial and Applied Mechanics, USA (2005)
10. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning 52(3), 199–215 (2003)
11. Roth, V.: Kernel fisher discriminants for outlier detection. Neural Computing 18(4), 942–960 (2006)
12. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. In: Advances in Neural Information Processing Systems, vol. 12, pp. 582–588. MIT Press, Cambridge (2000)
13. Tarassenko, L., Hayton, P., Cerneaz, N., Brady, M.: Novelty detection for the identification of masses in mammograms. In: Proceedings of the Fourth International IEEE Conference on Artificial Neural Networks, London, pp. 442–447. IEEE, Los Alamitos (1995)
14. Tax, D.: One-class Classification, Concept-learning in the Absence of Counter-examples. PhD thesis, Delft University of Technology, Netherlands (2001)
15. Tax, D., Duin, R.: Combining one-class classifiers. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 299–308. Springer, Heidelberg (2001)
16. Ypma, A., Duin, R.: Support objects for domain approximation. In: Proceedings of the 8th International Conference on Artificial Neural Networks, pp. 719–724. Springer, Berlin (1998)

Efficient Frequent Connected Subgraph Mining in Graphs of Bounded Treewidth

Tamás Horváth[1,2] and Jan Ramon[3]

[1] Dept. of Computer Science III, University of Bonn, Germany
[2] Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany
tamas.horvath@iais.fraunhofer.de
[3] Dept. of Computer Science, Katholieke Universiteit Leuven, Belgium
jan.ramon@cs.kuleuven.be

Abstract. The frequent connected subgraph mining problem, i.e., the problem of listing all connected graphs that are subgraph isomorphic to at least a certain number of transaction graphs of a database, cannot be solved in output polynomial time in the general case. If, however, the transaction graphs are restricted to forests then the problem becomes tractable. In this paper we generalize the positive result on forests to graphs of *bounded treewidth*. In particular, we show that for this class of transaction graphs, frequent connected subgraphs can be listed in *incremental polynomial time*. Since subgraph isomorphism remains NP-complete for bounded treewidth graphs, the positive complexity result of this paper shows that efficient frequent pattern mining is possible even for computationally hard pattern matching operators.

1 Introduction

During the last decade, *graph mining* developed into a separate field of knowledge discovery in databases, motivated by various practical applications for example in bioinformatics, computational chemistry, and the WWW. A basic task in this field is the *frequent connected subgraph mining (FCSM) problem*: Given a database of labeled graphs, called *transaction* graphs, and some positive integer threshold t, *list* all *connected* graphs that are subgraph isomorphic to at least t transaction graphs. Such frequent connected patterns have successfully been used, for example, in ligand-based virtual screening as features [6].

For arbitrary transaction graphs, the FCSM problem cannot be solved in output-polynomial time (if P $\neq$ NP) [10]. While several heuristic methods have been developed for this general problem that proved to be effective on various graph datasets, surprisingly there are only few results about tractable graph classes. To the best of our knowledge, the only positive (non-trivial) result towards this direction is about forests; the FCSM problem can be solved in incremental polynomial time for forest transaction graphs (see [5] for a survey on tree mining). The exploration of the border between tractable and intractable graph classes is an important theoretical challenge because it could provide useful insights into the problem which could then be exploited in the design of practical algorithms.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 520–535, 2008.

In this paper we take a step towards this goal by generalizing the positive result on forests to graphs of *bounded treewidth*. *Treewidth* [14] is a measure of tree-likeness of graphs that proved to be a useful property in algorithmic graph theory because several NP-hard problems on graphs become tractable for the class of bounded treewidth graphs. This class is also of practical importance, as it includes many graph classes appearing in practical applications (see, e.g., [2,4]). For example, the molecular graphs of the vast majority of pharmacological compounds have treewidth at most 3.

We present a levelwise search algorithm listing frequent connected subgraphs in incremental polynomial time if the treewidth of the transaction graphs is bounded by some constant. We make use of the fact that isomorphism between graphs of bounded treewidth can be decided efficiently [2]. To calculate the support count of candidate patterns, we use a modification of the subgraph isomorphism algorithm developed for graphs of bounded treewidth and *log-bounded fragmentation* [9], where the class of log-bounded fragmentation graphs properly contains the class of *bounded degree* graphs. This algorithm is based on a fundamental generic algorithm designed for deciding various morphisms between graphs of bounded treewidth and bounded degree [13]. In a nutshell, the main result of [13] is that several graph morphisms, including subgraph isomorphism, can be decided efficiently by a dynamic programming algorithm computing polynomially many, polynomial time computable properties if the treewidth and the degree of the graphs are both bounded by some constant.

Since we do not assume any bound on the degree, the number of such properties can be *exponentially* large. We can show, however, that for a given candidate pattern H, it is sufficient to compute only a *polynomially* large subset of these properties; the rest, maybe exponentially large set, can be derived from those of the frequent subgraphs listed before H. To show this result, we utilize the levelwise generation of frequent patterns and the anti-monotonic property of frequency. In this way, the delay can be exponential in the size of the input only after the enumeration of exponentially many frequent patterns. This technique might be of some independent interest and useful to design efficient algorithms where straightforward dynamic programming would require exponential space.

We note that subgraph isomorphism remains NP-complete even for connected graphs of bounded treewidth (see, e.g., [13]). The positive result of this paper thus provides an example of the case when efficient frequent pattern mining is possible even for NP-hard pattern matching operators. A significant consequence of our result is thus immediate to the study of frequent pattern mining: *Efficient frequent pattern mining is possible even for NP-hard pattern matching operators.*

The rest of the paper is organised as follows. In Section 2 we first collect the necessary notions and fix the notations. In Section 3 we present a generic levelwise frequent connected subgraph mining algorithm and analyse its computational properties. In Section 4 we adapt this generic algorithm to graphs of bounded treewidth and show that it lists frequent connected subgraphs in incremental polynomial time. Finally, in Section 5, we conclude along with an open problem. Due to space limitations, proofs are omitted in this short version.

2 Preliminaries

In this section we first briefly review some basic concepts and fix the notations used in this paper. We start with some standard definitions from graph theory.

Graphs. An *undirected graph* is a pair (V, E), where V is a finite set of *vertices* and $E \subseteq \{e \subseteq V : |e| = 2\}$ is a set of *edges*. A *labeled undirected graph* is a triple (V, E, λ), where (V, E) is an undirected graph and $\lambda : V \cup E \to \mathbb{N}$ is a function assigning a label to every element of $V \cup E$. Unless otherwise stated, in this paper by graphs we always mean *labeled undirected* graphs and denote the set of vertices, the set of edges, and the labeling function of a graph G by $V(G)$, $E(G)$, and λ_G, respectively.

Let G and G' be graphs. Then G' is a *subgraph* of G, if $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$, and $\lambda_{G'}(x) = \lambda_G(x)$ for every $x \in V(G') \cup E(G')$; it is an *induced subgraph* of G if it is a subgraph of G satisfying $\{u, v\} \in E(G')$ iff $\{u, v\} \in E(G)$ for every $u, v \in V(G')$. For a subset $S \subseteq V(G)$, $G[S]$ denotes the induced subgraph of G with vertex set S.

A *path* connecting the vertices $v_1, v_k \in V(G)$ in a graph G is a sequence $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\} \in E(G)$ such that the v_i's are pairwise distinct. A graph is *connected* if there is a path between any pair of its vertices. A *connected component* of a graph G is a maximal subgraph of G that is connected. The set of all connected components of a graph G is denoted by $\mathcal{C}(G)$.

Isomorphism and Subgraph Isomorphism. Let G_1 and G_2 be graphs. They are *isomorphic* if there is a *bijection* $\varphi : V(G_1) \to V(G_2)$ satisfying (i) $\{u, v\} \in E(G_1)$ iff $\{\varphi(u), \varphi(v)\} \in E(G_2)$ for every $u, v \in V(G_1)$, (ii) $\lambda_{G_1}(u) = \lambda_{G_2}(\varphi(u))$ for every $u \in V(G_1)$, and (iii) $\lambda_{G_1}(\{u, v\}) = \lambda_{G_2}(\{\varphi(u), \varphi(v)\})$ for every $\{u, v\} \in E(G_1)$. In this paper, isomorphic graphs are regarded as identical graphs.

For G_1 and G_2 above we say that G_1 is *subgraph isomorphic* to G_2, denoted $G_1 \leq G_2$, if G_1 is isomorphic to a subgraph of G_2. Deciding whether a graph is subgraph isomorphic to another graph is NP-complete, as it generalizes, e.g., the Hamiltonian path problem [7].

Treewidth. The notion of treewidth was reintroduced in [14]. It proved to be a useful parameter of graphs in algorithmic graph theory. A *tree-decomposition* of a graph G, denoted $TD(G)$, is a pair $(T, \mathcal{X})$, where T is an unordered tree and $\mathcal{X} = (X_z)_{z \in V(T)}$ is a family of subsets of $V(G)$ satisfying (i) $\cup_{z \in V(T)} X_z = V(G)$, (ii) for every $\{u, v\} \in E(G)$, there is a $z \in V(T)$ such that $u, v \in X_z$, and (iii) $X_{z_1} \cap X_{z_3} \subseteq X_{z_2}$ for every $z_1, z_2, z_3 \in V(T)$ such that z_2 is on the path connecting z_1 with z_3 in T. The set X_z associated with a node z of T is called the *bag* of z. The nodes of T will often be referred to as the nodes of $TD(G)$. The treewidth of $TD(G)$ is $\max_{z \in V(T)} |X_z| - 1$, and the *treewidth* of G, denoted $\mathrm{tw}(G)$, is the minimum treewidth over all tree-decompositions of G. By graphs of bounded treewidth we mean graphs of treewidth at most k, where k is some constant.

The following notation will be used many times in what follows. Let G be a graph, $TD(G) = (T, \mathcal{X})$ be a tree-decomposition of G, and $z \in V(T)$. Then $G_{[z]}$ denotes the induced subgraph of G defined by the union of the bags of z's descendants. (We note that z is considered also as a descendant of itself.)

3 Mining Frequent Connected Subgraphs

In this section we first define the frequent connected subgraph mining problem, present a generic listing algorithm based on levelwise search for this problem, and provide sufficient conditions for the efficiency of this algorithm. Applying these conditions, we then show that the frequent connected subgraph mining problem can be solved in incremental polynomial time for *forest* transaction graphs and for transaction graphs that have *bounded treewidth* and *log-bounded fragmentation*. In the next section we will show how to adapt the generic algorithm of this section to mining frequent connected subgraphs from graphs of bounded treewidth that do not necessarily have log-bounded fragmentation. We start with the definition of the general problem setting.

The Frequent Connected Subgraph Mining (FCSM) Problem: *Given* a class $\mathcal{G}$ of graphs, a transaction database DB of graphs from $\mathcal{G}$ (i.e., a multiset of graphs from $\mathcal{G}$), and an integer threshold $t > 0$, *list* the set of frequent connected subgraphs, that is, the set of connected graphs that are subgraph isomorphic to at least t graphs in DB.

The *parameter* of the above problem is the size of DB. One can easily construct examples when the number of frequent connected subgraphs is exponential in this parameter. Thus, in general, the set of all frequent connected subgraphs cannot be computed in time polynomial in the size of DB. Since this is a common feature of listing problems, the following problem classes are usually considered in the literature (see, e.g., [12]): Let S be a set of cardinality N. Then its elements, say $s_1, \ldots, s_N$, are listed with

polynomial delay if the time before printing s_1, the time between printing s_i and s_{i+1} for every $i = 1, \ldots, N-1$, and the termination time after printing s_N is bounded by a polynomial of the size of the input,

incremental polynomial time if s_1 is printed with polynomial delay, the time between printing s_i and s_{i+1} for every $i = 1, \ldots, N-1$ (resp. the termination time after printing s_N) is bounded by a polynomial of the combined size of the input and the set $\{s_1, \ldots, s_i\}$ (resp. S),

output polynomial time (or *polynomial total time*) if S is printed in the combined size of the input and the *entire* set S.

Clearly, polynomial delay implies incremental polynomial time, which, in turn, implies output polynomial time. Furthermore, in contrast to incremental polynomial time, the delay of an output polynomial time algorithm may be exponential in the size of the input even before printing the first element of the output.

Although several frequent connected subgraph mining algorithms have been developed that proved to be effective in various practical applications, we note that, unless P = NP, the FCSM problem cannot be solved in output polynomial time for arbitrary transaction graphs [10]. It can be solved, however, in incremental polynomial time if the transaction graphs are restricted to forests (see, e.g., [5,11]). The main contribution of this work is to extend this positive result to graphs of *bounded treewidth*. To achieve this goal, we first give a generic levelwise mining algorithm designed for downward closed graph classes.

Algorithm 1. FREQUENT CONNECTED SUBGRAPH MINING

Require: database DB of graphs from a downward closed class $\mathcal{G}$ and integer $t > 0$
Ensure: all frequent connected subgraphs

```
1: let S_0 ⊆ G be the set of frequent graphs consisting of a single labeled vertex
2: print S_0
3: for (l := 0; S_l ≠ ∅; ++l) do
4:    C_{l+1} := S_{l+1} := ∅
5:    forall P ∈ S_l do
6:       forall H ∈ ρ(P) ∩ G satisfying (i) H ∉ C_{l+1} and (ii) ρ^{-1}(H) ⊆ S_l do
7:          add H to C_{l+1}
8:          if SupportCount(H) ≥ t then
9:             print H and add it to S_{l+1}

10: function SupportCount(H)
11:    counter := 0
12:    forall G in DB do
13:       if H ≤ G then
14:          counter++
15:    return counter
```

3.1 A Generic Mining Algorithm

A generic algorithm listing frequent connected subgraphs with levelwise search is given in Algorithm 1. It requires the class $\mathcal{G}$ of transaction graphs to be closed downward, i.e., for every $G \in \mathcal{G}$, $\mathcal{G}$ contains all subgraphs of G. In the algorithm we first compute the set of frequent graphs from $\mathcal{G}$ that consist of a single vertex. (We recall that by graphs we mean labeled graphs.) In the main loop we then iteratively compute the set S_{l+1} of frequent connected graphs containing $l+1$ edges from those containing l edges for every $l \geq 0$. In particular, for each frequent pattern $P \in S_l$, we compute the set $\rho(P) \cap \mathcal{G}$ of graphs, where the elements of $\rho(P)$ are obtained from P by either connecting two vertices with a labeled edge or by adding a new labeled vertex to P and connecting it with an old vertex by a labeled edge. Clearly, the graphs in $\rho(P)$ are all connected, as P is connected. For each $H \in \rho(P) \cap \mathcal{G}$, we check whether or not it has already been generated during the current iteration step of the main loop (condition (i) in line 6). If not, we also check for each graph in $\rho^{-1}(H)$ whether it is frequent, where $\rho^{-1}(H)$ is the set of *connected* graphs obtained from H by removing an edge (condition (ii) in line 6). Throughout this paper, candidate patterns generated by levelwise search that satisfy conditions (i) and (ii) in line 6 are called *strong candidates*. If H is a strong candidate, we add it to the set C_{l+1} of candidate graphs consisting of $l+1$ edges and compute its support count. If it is frequent, i.e., subgraph isomorphic to at least t graphs in DB, we print it and add it to the set S_{l+1} of frequent connected graphs containing $l+1$ edges.

As mentioned earlier, without any further assumption on $\mathcal{G}$, the FCSM problem cannot be solved in output polynomial time [10]. If, however, $\mathcal{G}$ satisfies the conditions of the theorem below, one can obtain tractable problem classes.

Theorem 1. *Let $\mathcal{G}$ be a graph class satisfying the following properties:*

(i) $\mathcal{G}$ is closed downward,
(ii) the membership problem in $\mathcal{G}$, i.e., whether $G \in \mathcal{G}$ for any graph G can be decided in polynomial time, and
(iii) for every $H, G \in \mathcal{G}$ such that H is connected, it can be decided in polynomial time whether H is subgraph isomorphic to G (i.e., if $H \leq G$).

If the transaction graphs in DB belong to $\mathcal{G}$ then Algorithm 1 lists the frequent connected subgraphs in incremental polynomial time.

Notice that condition (i) above does not require the elements of $\mathcal{G}$ to be connected. The proof of the theorem is straightforward by noting that the cardinality and hence, the sizes of the sets $\rho(H) \cap \mathcal{G}$ and $\rho^{-1}(H)$ in line 6 are bounded by a polynomial of the size of DB, and both sets can be computed in polynomial time. Thus, the size of C_{l+1} is bounded by the combined size of DB and S_l.

Without proof, we now mention two immediate applications of the above theorem. The first result on forests follows from several results (see, e.g., [5,11]).

Corollary 1. *The FCSM problem can be solved in incremental polynomial time for forest transaction graphs.*

To state a second application of Theorem 1, we first need a definition. A graph G has *k-log-bounded* (or simply, *log-bounded*) *fragmentation* [9] if

$$|\mathcal{C}(G[V(G) \setminus S])| - |\mathcal{C}(G)| = O\left(k \log |V(G)|\right)$$

for every $S \subseteq V(G)$ satisfying $|S| \leq k$. In other words, G has log-bounded fragmentation if the removal of any set of at most k vertices and all adjacent edges from G results in a graph with $|\mathcal{C}(G)| + O\left(k \log |V(G)|\right)$ connected components. As an obvious example, graphs with maximum degree bounded by some constant have log-bounded fragmentation. Using Theorem 1, for graphs of bounded treewidth and log-bounded fragmentation, the following result can be shown:

Corollary 2. *The FCSM problem can be solved in incremental polynomial time if the transaction graphs have bounded treewidth and log-bounded fragmentation.*

Consider the NCI chemical database[1] used as a benchmark graph dataset in graph mining. Out of the 250251 molecular graphs in this dataset, 243638 (i.e., 97,36%) compounds have treewidth at most 2, 250186 (i.e., 99,97%) treewidth at most 3; there are only 65 (i.e., 0,03%) compounds with treewidth at least 4. However, compounds of small treewidth have often relatively large degree.

Motivated by applications of frequent connected subgraphs in computational drug design, in particular, in ligand-based virtual screening (see, e.g., [6]), it is therefore natural to ask if the positive result of Corollary 2 can be extended to *arbitrary* graphs of bounded treewidth. This raises the question whether condition (iii) of Theorem 1 is indeed necessary because subgraph isomorphism between graphs of bounded treewidth is NP-complete even for connected patterns (see, e.g., [13]). In the rest of this paper we show that for graphs of bounded treewidth, condition (iii) is unnecessary.

[1] `http://cactus.nci.nih.gov/`

4 Mining Frequent Connected Subgraphs in Graphs of Bounded Treewidth

Consider again Theorem 1. It is easy to see that condition (i) holds for arbitrary graphs of bounded treewidth. Regarding condition (ii), for any constant k, one can decide in linear time whether a graph has treewidth at most k and if yes, compute a tree-decomposition of treewidth at most k [3]. However, condition (iii) on the efficiency of subgraph isomorphism does not hold for arbitrary connected graphs of bounded treewidth. For graphs of bounded treewidth, this condition has a side effect only in the support counting step (line 8) which is based on deciding subgraph isomorphism (line 13). Thus, to show that the FCSM problem can be solved in incremental polynomial time for graphs of bounded treewidth, it is sufficient to show that subgraph isomorphism can be decided in time polynomial in the combined size of the input and the set of frequent patterns computed before the current candidate pattern.

The subgraph isomorphism problem remains NP-complete even when both the pattern and transaction graphs have bounded treewidth and the pattern graph is connected [13]. The NP-completeness of the subforest isomorphism problem [7] implies that the connectivity of the pattern graph is necessary even for acyclic graphs. For graphs of treewidth at most k, there is a clear demarcation between tractable and intractable instances of the subgraph isomorphism problem: if the pattern is not k-connected or has more than k vertices of unbounded degree then the subgraph isomorphism is NP-complete; otherwise it can be decided in polynomial time [8]. However, as we show below, for *efficient* mining of frequent connected subgraphs, these conditions are unnecessary because the anti-monotonicity of frequency allows the mining algorithm to utilise the information computed earlier for the candidates' frequent ancestors.

The rest of this section is organised as follows. In Section 4.1 we first overview the subgraph isomorphism algorithm described in [9]. In Section 4.2 we then state, without proof, that for every connected graphs H and G of bounded treewidth, this algorithm computes only a polynomial number of *new* properties needed to decide $H \leq G$; any property from the rest, possibly exponentially many properties, can be derived from the properties computed for the frequent patterns in the previous steps. Finally, in Section 4.3, we show how to integrate this modified subgraph isomorphism algorithm into Algorithm 1.

4.1 Subgraph Isomorphism between Graphs of Bounded Treewidth and Log-Bounded Fragmentation

Throughout this section H and G denote connected graphs of treewidth at most k, where k is some constant. In fact, we need only H to be connected. Since any subgraph isomorphism maps a connected graph into a connected component of a graph, we may assume without loss of generality that G is also connected.

Following the dynamic-programming approach of [1], the algorithm described in [9] first computes a *nice* tree-decomposition $TD(G) = (T, \mathcal{X})$ of G, where a

nice tree-decomposition is a tree-decomposition such that T is a rooted binary tree composed of three types of nodes: (i) a *leaf* node has no children, (ii) a *separator* node z has a single child z' with $X_z \subseteq X_{z'}$, and (iii) a *join* node z has two children z_1 and z_2 with $X_z = X_{z_1} \cup X_{z_2}$. It follows from the results in [4] that for graphs of treewidth at most k, such a nice tree-decomposition of treewidth at most k always exists and can be constructed in linear time.

Given a nice tree-decomposition $TD(G)$ of G, the algorithm in [9] computes a set of tuples (properties) for each node in a bottom-up manner and decides whether $H \leq G$ holds by checking a condition for the set computed for the root of $TD(G)$. To define this condition precisely, we need some notions. For a node z in $TD(G)$, a *partial solution* relative to z is a subgraph isomorphism from a subgraph H' of H into $G_{[z]}$. An *iso-quadruple* relative to a node z of $TD(G)$ is a quadruple $(S, \mathcal{D}, K, \psi)$, where

- $S \subseteq V(H)$ satisfying $|S| \leq k+1$,
- $\mathcal{D} \subseteq \mathcal{C}(H[V(H) \setminus S])$,
- $K = H[S \cup V(\mathcal{D})]$, and
- ψ is an injective function mapping S to X_z.

We note that K is redundant in the above notation; we add it to the tuple only for the reader's convenience. The set of all iso-quadruples of H relative to a node z of $TD(G)$ is denoted by $\Gamma(H, z)$.

We need some further definitions. A *characteristic of a partial solution* φ relative to a node z in $TD(G)$ is an iso-quadruple $(S, \mathcal{D}, K, \psi) \in \Gamma(H, z)$ such that φ is a subgraph isomorphism from K to $G_{[z]}$ satisfying $\varphi(u) = \psi(u)$ for every $u \in S$ and $\varphi(v) \notin X_z$ for every $v \in V(\mathcal{D})$. The definitions imply that $\varphi(u) \in X_z$ for every $u \in S$. Finally, a *z-characteristic* is an iso-quadruple in $\Gamma(H, z)$ that is the characteristic of at least one partial solution relative to z. The set of all z-characteristics of H relative to z is denoted by $\Gamma_{\mathrm{ch}}(H, z)$. Clearly, $\Gamma_{\mathrm{ch}}(H, z) \subseteq \Gamma(H, z)$. The following lemma from [9] provides a characterization of subgraph isomorphism in terms of r-characteristics for the root r of $TD(G)$:

Lemma 1. *Let r be the root of a nice tree-decomposition $TD(G)$ of G. Then $H \leq G$ iff there exists an iso-quadruple $(S, \mathcal{D}, H, \psi)$ in $\Gamma_{\mathrm{ch}}(H, r)$.*

Notice that by definition, S can be the empty set. Thus, by the above lemma, we can decide $H \leq G$ by computing and testing the set of r-characteristics for the root r of $TD(G)$. This can be carried out by traversing $TD(G)$ in a postorder manner and computing the set of characteristics for each non-leaf node from those of its children. Depending on the type of the current node z visited, we perform one of the following steps (for more details on the method described below, the reader is referred, e.g., to [9]):

Leaf Nodes: If z is a leaf in $TD(G)$ then for all $(S, \mathcal{D}, K, \psi) \in \Gamma_{\mathrm{ch}}(H, z)$ it holds that $\mathcal{D} = \emptyset$ and $K = H[S]$. For this case, the following lemma holds.

Lemma 2. *Let z be a leaf in $TD(G)$ and $\xi = (S, \emptyset, H[S], \psi) \in \Gamma(H, z)$. Then $\xi \in \Gamma_{\mathrm{ch}}(H, z)$ iff ψ is a subgraph isomorphism from $H[S]$ to $G[X_z]$.*

Since S and X_z both have at most $k+1$ vertices, $|\Gamma(H,z)|$ is bounded by $(k+1)! \cdot |V(H)|^{k+1}$. Furthermore, using the above lemma, one can decide in time $(k+1)^2$, whether an iso-quadruple in $\Gamma(H,z)$ is a z-characteristic. Thus, the set of characteristics for leaf nodes can be computed in polynomial time.

Separator Nodes: If z is a separator node then, by definition, it has a single child z' with $X_z \subseteq X_{z'}$. For an iso-quadruple $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma(H,z)$, let $\Pi(\xi)$ denote the set of iso-quadruples $(S', \mathcal{D}', K', \psi') \in \Gamma(H,z')$ satisfying

(i) $S = \{v \in S' : \psi'(v) \in X_z\}$,
(ii) $\mathcal{D}' = \{D' \in \mathcal{C}(H[V(H) \setminus S']) :$ D' is a subgraph of some $D \in \mathcal{D}$ $\}$, and
(iii) $\psi(v) = \psi'(v)$ for every $v \in S$.

Using this definition, the set of characteristics relative to a separator node can be computed by the following lemma.

Lemma 3. *Let z be a separator node in $TD(G)$ with child z' and $\xi \in \Gamma(H,z)$ be an iso-quadruple. Then $\xi \in \Gamma_{\mathrm{ch}}(H,z)$ iff $\Gamma_{\mathrm{ch}}(H,z') \cap \Pi(\xi) \neq \emptyset$.*

For a separator node z with child z' and $\xi \in \Gamma(H,z)$, $|\Pi(\xi)|$ is bounded by $(k+1)! \cdot |V(H)|^{k+1}$. Using some advanced data structure for storing the set of characteristics of a node, one can decide in polynomial time whether an iso-quadruple $\xi' \in \Pi(\xi)$ is a z'-characteristic. Thus, for an iso-quadruple ξ in $\Gamma(H,z)$, we can decide in polynomial time whether ξ is a z-characteristic if the set of z'-characteristics has already been computed. However, $|\Gamma(H,z)|$ can be exponential in the size of H if only its treewidth is restricted.

Join Nodes: If z is a join node then it has two children z_1 and z_2 with bags satisfying $X_z = X_{z_1} \cup X_{z_2}$. From the sets of characteristics relative to the children of z, the set of z-characteristics can be computed by using the following lemma:

Lemma 4. *Let z be a join node in $TD(G)$ with children z_1 and z_2, and let $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma(H,z)$ be an iso-quadruple. Then $\xi \in \Gamma_{\mathrm{ch}}(H,z)$ iff there exist $(S_1, \mathcal{D}_1, K_1, \psi_1) \in \Gamma_{\mathrm{ch}}(H,z_1)$ and $(S_2, \mathcal{D}_2, K_2, \psi_2) \in \Gamma_{\mathrm{ch}}(H,z_2)$ such that*

(i) $S_i = \{v \in S : \psi(v) \in X_{z_i}\}$ for $i = 1, 2$,
(ii) the connected components of $\mathcal{D}$ are partitioned into $\mathcal{D}_1$ and $\mathcal{D}_2$,
(iii) for $i = 1, 2$, $\psi_i(v) = \psi(v)$ for every $v \in S_i$, and
(iv) ψ preserves the labels and the edges.

By condition (ii) above we have to take the set of all possible partitionings of a set $\mathcal{D}$ of connected components, which is exponential in the number of connected components of $\mathcal{D}$. Thus, the situation is much worse for join nodes than for separator nodes because not only the number of z-characteristics can be exponential, but the computation of a particular z-characteristic may involve exponentially many tests.

In case of log-bounded fragmentation graphs, the set of characteristics can be computed in polynomial time for separator and join nodes as well [9]. If, however, we do not assume any further restrictions on connected graphs of bounded treewidth, the above bottom-up evaluation may require exponential time.

4.2 Subgraph Isomorphism between Graphs of Bounded Treewidth

In this section we show that the subgraph isomorphism algorithm described above can be integrated into Algorithm 1 in such a way that for any *strong* candidate pattern H (i.e., which satisfies conditions (i) and (ii) in line 6 of Algorithm 1) and for any transaction graph $G \in DB$, $H \leq G$ can be decided in time polynomial in the combined size of H, G, and the set of *frequent* patterns listed before H. The key observation is that it is sufficient to compute only polynomially many characteristics for H and to use the characteristics of the frequent patterns computed before H. We recall that Algorithm 1 lists the set of frequent patterns with levelwise search. Thus, all connected subgraphs H' of H have already been processed in the frequency counting step. One of the changes in the algorithm is that for every transaction graph G, we compute a nice tree-decomposition $TD(G)$ in a preprocessing step and use this decomposition during the entire mining process. This is necessary for the reutilisation of the characteristics computed for the frequent patterns in earlier steps.

Non-Redundant Iso-Quadruples. We start with a definition that will be used to determine the set of characteristics of a strong candidate pattern H relative to a node, which cannot be recovered from the set of characteristics of the frequent patterns computed before H. Let H_1, H_2, and G be connected graphs of bounded treewidth, $TD(G)$ be the nice tree-decomposition of G computed in the preprocessing step, and z be a node in $TD(G)$. Let $\xi_1 = (S_1, \mathcal{D}_1, K_1, \psi_1) \in \Gamma(H_1, z)$ and $\xi_2 = (S_2, \mathcal{D}_2, K_2, \psi_2) \in \Gamma(H_2, z)$ be iso-quadruples relative to z. We say that ξ_1 is *equivalent* to ξ_2, denoted $\xi_1 \equiv \xi_2$, if there is an isomorphism π between K_1 and K_2 such that π is a bijection between S_1 and S_2 and $\psi_1(v) = \psi_2(\pi(v))$ for every $v \in S_1$. The proof of the proposition below follows directly from the definitions.

Proposition 1. *Let ξ_1 and ξ_2 be equivalent iso-quadruples relative to a node z of $TD(G)$. Then ξ_1 is a z-characteristic iff ξ_2 is a z-characteristic.*

In order to utilise the information computed previously, for each node z in $TD(G)$, we store the set of z-characteristics computed for the frequent patterns listed earlier by the algorithm. Proposition 1 above implies that it is sufficient to store only one representative z-characteristic from each equivalence class of the set of z-characteristics. The following definition will be used many times in what follows. An iso-quadruple $\xi \in \Gamma(H, z)$ of a strong candidate pattern H is *redundant* if there is an equivalent iso-quadruple in $\Gamma(P, z)$ for some frequent pattern P computed before H. The set of *non-redundant* iso-quadruples of a pattern H relative to a node z in $TD(G)$ is denoted by $\Gamma_{\mathrm{nr}}(H, z)$.

For a strong candidate pattern H and iso-quadruple $\xi \in \Gamma(H, z)$, we have to test whether ξ is a z-characteristic of $TD(G)$ only when ξ is non-redundant; if ξ is non-redundant, we add it to the set of characteristics stored for z only if it is a z-characteristic. The number of iso-quadruples of a pattern relative to a node z can be exponential. Theorem 2 below is of special importance for the main result of this paper, as it implies that the number of non-redundant iso-quadruples of

a strong candidate pattern is always bounded by a polynomial of the combined size of the pattern and the largest graph in the transaction database. In the following lemma we give an upper bound on the number of non-redundant iso-quadruples of a pattern relative to a node. Due to space limitations, we omit the proof which is based on combinatorial arguments.

Lemma 5. *Let H be a strong candidate pattern generated by levelwise search and let z be a node in $TD(G)$ for some transaction graph G. Then $|\Gamma_{\mathrm{nr}}(H,z)|$ is bounded by $O\left(|V(H)|^{k+1}\right)$.*

Using the above lemma, we can state the following result concerning non-redundant iso-quadruples of a pattern.

Theorem 2. *In order to decide $H \leq G$, it is sufficient to check for at most $O\left(|V(G)| \cdot |V(H)|^{k+1}\right)$ iso-quadruples of H, whether they are characteristics relative to some node of $TD(G)$.*

Proof. To decide $H \leq G$, it is sufficient to check non-redundant iso-quadruples. The statement then follows directly from Lemma 5 by noting that the size of $TD(G)$ is linear in that of G [3]. □

By Lemmas 3 and 4, for a node z in $TD(G)$, the set of z-characteristics is computed from the sets of z'-characteristics of the children z' of z. Looking up from the set of z'-characteristics, one can decide in time polynomial in the size of H, whether an iso-quadruple $\xi = (S, \mathcal{D}, K, \psi)$ is a z'-characteristic. We omit the details of this technical result from this short version. The key is that we color each vertex $u \in S$ of K by $\psi(u) \in X_{z'}$ and compute a canonical string representation of this colored graph obtained from K. This canonical string representation is unique modulo isomorphism and can be computed in time polynomial in the size of K. Storing the canonical string representation of the z'-characteristics in some advanced data structure, e.g., in prefix trees, we can decide in linear time, whether there exists a characteristic equivalent to ξ among the set of z'-characteristics computed so far for the frequent patterns.

Deciding Subgraph Isomorphism. We now turn to the question of how to decide subgraph isomorphism using only non-redundant characteristics. Let H be a strong candidate pattern generated by levelwise search. For a transaction graph G and node z in $TD(G)$, let $\Sigma(z)$ and $\Gamma_{\mathrm{nr,ch}}(H,z)$ denote the set of all non-redundant z-characteristics computed for the frequent patterns listed before H and the set of non-redundant z-characteristics computed for H, respectively. Let r be the root of $TD(G)$. Then Lemma 1 and Proposition 1 together imply that $H \leq G$ if and only if there are iso-quadruples $\xi = (S, \mathcal{D}, H, \psi) \in \Gamma(H,r)$ and $\xi' \in \Sigma(r) \cup \Gamma_{\mathrm{nr,ch}}(H,r)$ such that $\xi \equiv \xi'$. This condition can be decided efficiently, as (i) there are at most $O\left(|V(H)|^{k+1}\right)$ iso-quadruples of H relative to r that have to be tested, (ii) $|\Sigma(r) \cup \Gamma_{\mathrm{nr,ch}}(H,r)|$ is bounded by a polynomial of the combined size of the input and the set of frequent patterns computed before H, and (iii) equivalence between iso-quadruples can be decided in polynomial time (see the remarks before this paragraph).

Algorithm 2. NON-REDUNDANT_CHARACTERISTICS

Require: connected graphs H, G of treewidth at most k, a nice tree-decomposition $TD(G)$ of G with nodes associated with $\Sigma(w)$ for every node w in $TD(G)$, and a node z in $TD(G)$
Ensure: set $\Gamma_{\mathrm{nr,ch}}(H, z)$ of non-redundant z-characteristics
1: $\Gamma_{\mathrm{nr,ch}}(H, z) = \emptyset$
2: compute the set $\Gamma_{\mathrm{nr}}(H, z)$ of non-redundant iso-quadruples of z
3: **forall** $\xi = (S, \mathcal{D}, K, \psi) \in \Gamma_{\mathrm{nr}}(H, z)$ **do**
4: **if** z is a leaf node **then**
5: add ξ to $\Gamma_{\mathrm{nr,ch}}(H, z)$ if $\mathcal{D} = \emptyset$ and ψ is a subgraph isom. from $H[S]$ to $G[X_z]$
6: **else if** z is a separator node with child z'
7: call the algorithm recursively with parameters H, G, $TD(G)$, and z'
8: add ξ to $\Gamma_{\mathrm{nr,ch}}(H, z)$ if $\exists \xi' \in \Pi(\xi)$ and $\xi'' \in \Sigma(z') \cup \Gamma_{\mathrm{nr,ch}}(H, z')$ s.t. $\xi' \equiv \xi''$
9: **else** // z is a join node with children z_1 and z_2
10: $\Gamma_{\mathrm{nr,ch}}(H, z_1) :=$ NON-REDUNDANT_CHARACTERISTICS$(H, G, TD(G), z_1)$
11: $\Gamma_{\mathrm{nr,ch}}(H, z_2) :=$ NON-REDUNDANT_CHARACTERISTICS$(H, G, TD(G), z_2)$
12: add ξ to $\Gamma_{\mathrm{nr,ch}}(H, z)$ if $\exists \xi_i \in \Sigma(z_i) \cup \Gamma_{\mathrm{nr,ch}}(H, z_i)$ for $i = 1, 2$ s.t. $\xi \equiv \xi_1 \oplus \xi_2$
13: **return** $\Gamma_{\mathrm{nr,ch}}(H, z)$

Thus, we only have to discuss the computation of the $\Gamma_{\mathrm{nr,ch}}(H, z)$'s for the nodes in $TD(G)$. Before going into the details, we first note that if H is frequent then $\Gamma_{\mathrm{nr,ch}}(H, z)$ must be added to $\Sigma(z)$ after the frequency counting step in Algorithm 1. A recursive algorithm computing $\Gamma_{\mathrm{nr,ch}}(H, z)$ for every node z in $TD(G)$ is given in Algorithm 2. For the current node z in $TD(G)$, the algorithm first sets $\Gamma_{\mathrm{nr,ch}}(H, z)$ to $\emptyset$ and then computes the set $\Gamma_{\mathrm{nr}}(H, z)$ of non-redundant iso-quadruples relative to z. As shown earlier, $\Gamma_{\mathrm{nr}}(H, z)$ can be computed in time polynomial in the combined size of H and G, as they have bounded treewidth. Then, depending on the type of z, for every iso-quadruple $\xi = (S, \mathcal{D}, K, \psi)$ in $\Gamma_{\mathrm{nr}}(H, z)$, one of the following steps is performed:

- z **is a Leaf Node:** We can apply Lemma 2 for this case, but only for the elements in $\Gamma_{\mathrm{nr}}(H, z)$.
- z **is a Separator Node:** The proof of the lemma stated below for this case follows directly from Proposition 1 and Lemma 3. We recall that $\Pi(\xi)$ denotes the set of iso-quadruples in $\Gamma(H, z)$ satisfying the conditions before Lemma 3.

Lemma 6. *Let H be a strong candidate pattern generated by levelwise search, z be a separator node in $TD(G)$, z' be the child of z, and $\xi \in \Gamma_{\mathrm{nr}}(H, z)$. Then ξ is a z-characteristic iff there exist $\xi' \in \Pi(\xi)$ and $\xi'' \in \Sigma(z') \cup \Gamma_{\mathrm{nr,ch}}(H, z')$ such that $\xi' \equiv \xi''$.*

Since $\Pi(\xi)$ has at most $(k+1)! \cdot |V(H)|^{k+1}$ elements and it can be computed efficiently, we can decide in polynomial time whether it has an element equivalent to an element of $\Sigma(z') \cup \Gamma_{\mathrm{nr,ch}}(H, z')$, that is, whether ξ is a z-characteristic. Combining this result with Lemma 5, we have that $\Gamma_{\mathrm{nr,ch}}(H, z)$ can be computed in time polynomial in the size of H.

z **is a Join Node:** Let z_1 and z_2 be the children of z and let $\xi_1 = (S_1, \mathcal{D}_1, K_1, \psi_1)$ and $\xi_2 = (S_2, \mathcal{D}_2, K_2, \psi_2)$ be z_1- and z_2-characteristics, respectively. We may assume w.l.o.g. that K_1 and K_2 are vertex disjoint. Then the *join* of ξ_1 and ξ_2, denoted $\xi_1 \oplus \xi_2$, is an iso-quadruple $\xi = (S, \mathcal{D}, K, \psi)$, where

- $S = \{w_u : u \in \psi_1(S_1) \cup \psi_2(S_2)\}$ is a set of new vertices,
- $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$,
- K is the graph obtained by adding to the set $\mathcal{D}$ of connected components (i) the set S of vertices, (ii) the edge $\{w_{\psi_i(u)}, w_{\psi_i(v)}\}$ for all $u, v \in S_i$ such that $\{u, v\} \in E(K_i)$, and (iii) the edge $\{w_{\psi(u)}, v\}$ for every $u \in S_i$ and $v \in V(\mathcal{D}_i)$ such that $\{u, v\} \in E(K_i)$ for $i = 1, 2$, and
- $\psi : S \to X_z$ is defined by $\psi(w_u) = u$ for every $w_u \in S$.

Notice that $|S| \leq k + 1$ because $\psi_1(S_1) \cup \psi_2(S_2) \subseteq X_{z_1} \cup X_{z_2} = X_z$ and $|X_z| \leq k + 1$, as z is a join node. Using the above definition, we give a lemma characterizing the set of z-characteristics of a candidate pattern H in terms of z_1- and z_2-characteristics computed before H. Although the lemma is used only for non-redundant iso-quadruples, we formulate it in a general form. The following lemma follows from Proposition 1 and Lemma 4.

Lemma 7. *Let H be a strong candidate pattern generated by levelwise search, z be a join node in $TD(G)$, and z_1 and z_2 be the children of z. Let $\xi \in \Gamma(H, z)$. Then ξ is a z-characteristic iff there exist $\xi_1 \in \Sigma(z_1) \cup \Gamma_{\mathrm{nr,ch}}(H, z_1)$ and $\xi_2 \in \Sigma(z_2) \cup \Gamma_{\mathrm{nr,ch}}(H, z_2)$ such that $\xi_1 \oplus \xi_2 \equiv \xi$.*

Using the arguments applied to separator nodes, we have that $\Gamma_{\mathrm{nr,ch}}(H, z)$ can be computed in time polynomial in the combined size of H and the set of frequent patterns computed before H; the second term follows from the facts that (i) the number of pairs (ξ_1, ξ_2) for which the join must be computed and tested is at most $|\Sigma(z_1) \cup \Gamma_{\mathrm{nr,ch}}(H, z_1)| \cdot |\Sigma(z_2) \cup \Gamma_{\mathrm{nr,ch}}(H, z_2)|$, and (ii) for $i = 1, 2$, $|\Sigma(z_i) \cup \Gamma_{\mathrm{nr,ch}}(H, z_i)|$ is bounded by the combined size of H and the set of frequent patterns computed before H. Putting together Lemmas 6 and 7 with the remarks above, we have the following lemma:

Lemma 8. *For every strong candidate pattern H generated by levelwise search and for every G in DB, both of bounded treewidth, Algorithm 2 is correct and computes $\Gamma_{\mathrm{nr,ch}}(H, z)$ for every node z in $TD(G)$ in time polynomial in the combined size of H, G, and the set of frequent patterns computed before H.*

The following theorem states that subgraph isomorphism between graphs of bounded treewidth can be decided in incremental polynomial time.

Theorem 3. *Let DB be a database of transaction graphs of treewidth at most k and H be a strong candidate pattern of treewidth at most k which is generated by a levelwise search algorithm mining frequent connected subgraphs from DB. Let $TD(G)$ be some fixed nice tree-decomposition used by the mining algorithm and let r be the root of $TD(G)$. Then $H \leq G$ iff*

$$\exists (S, \mathcal{D}, H, \psi) \in \Gamma(H, r) \text{ equivalent to some } \xi \in \Sigma(r) \cup \Gamma_{\mathrm{nr,ch}}(H, r) \ . \qquad (1)$$

Furthermore, condition (1) can be decided in time polynomial in the combined size of H, G, and the set of frequent patterns computed before H.

Algorithm 3. Frequent Connected Subgraph Mining in Graphs of Bounded Treewidth

Require: database DB of graphs of treewidth at most k, integer $t > 0$
Ensure: all frequent connected subgraphs

```
   // preprocessing of the transaction graphs
 1: forall G in DB do
 2:   compute a nice tree-decomposition TD(G) of G
 3:   forall node z in TD(G) do Σ(z) := ∅
   // computing frequent subgraphs
 4: let S_0 be the set of frequent graphs consisting of a single labeled vertex
 5: print S_0
 6: for (l := 0; S_l ≠ ∅; ++l) do
 7:   C_{l+1} := S_{l+1} := ∅
 8:   forall P ∈ S_l do
 9:     forall H ∈ ρ(P) satisfying tw(H) ≤ k ∧ H ∉ C_{l+1} ∧ ρ^{-1}(H) ⊆ S_l do
10:       add H to C_{l+1}
11:       if SupportCount_Treewidth(H) ≥ t then          // H is frequent
12:         print H and add it to S_{l+1}
13:         forall G in DB such that G is marked do
14:           forall node z in TD(G) do Σ(z) := Σ(z) ∪ Γ_{nr,ch}(H, z)

15: function SupportCount_Treewidth(H)
16:   counter := 0
17:   forall G in DB do
18:     unmark G
19:     r := root of TD(G)
20:     Γ_{nr,ch}(H, r) := Non-Redundant_Characteristics(H, G, TD(G), r)
21:     if ∃(S, D, H, ψ) ∈ Γ(H, r) equiv. to some ξ ∈ Σ(r) ∪ Γ_{nr,ch}(H, r) then
22:       counter++                                      // H ≤ G
23:       mark G
24:   return counter
```

Proof. The first part of the theorem holds by Lemma 1 and Proposition 1. Regarding the second, i.e., complexity part, we first note that by Lemma 8, $\Gamma_{\mathrm{nr,ch}}(H, r)$ can be computed in time polynomial in the combined size of H, G, and the set of frequent patterns computed before H. Furthermore, the number of iso-quadruples in $\Gamma(H, r)$ to be checked in (1) is bounded by $(k+1)! \cdot |V(H)^{k+1}|$. The statement then follows by noting that $|\Sigma(r) \cup \Gamma_{\mathrm{nr,ch}}(H, r)|$ is bounded by a polynomial of the combined size of H and the set of frequent patterns listed before H and that it can be decided in time linear in the size of H whether an iso-quadruple $\xi \in \Gamma(H, r)$ is an element of $\Sigma(r) \cup \Gamma_{\mathrm{nr,ch}}(H, r)$. □

4.3 The Mining Algorithm

Combining the results of the previous sections, we now show that the FCSM problem can be solved in incremental polynomial time from graphs of bounded treewidth. A listing algorithm solving this problem is given in Algorithm 3.

In the preprocessing step (lines 1–3 of Algorithm 3), we first compute a nice tree decomposition $TD(G)$ for every transaction graph G and initialize the set variables $\Sigma(z)$ for every node z in $TD(G)$. Since a nice tree-decomposition of treewidth k can be computed in linear time for graphs of treewidth at most k [3], the preprocessing step can be performed in time linear in the size of DB.

Lines 4–14, the main part of Algorithm 3, is an adaptation of the generic levelwise search Algorithm 1. One of the differences is that in lines 13–14 we have to update the sets of non-redundant characteristics for frequent patterns. Since $\mathcal{G}$ in Algorithm 1 is now the class of graphs of treewidth at most k, deciding whether H is an element of $\mathcal{G}$ corresponds to checking whether $\mathrm{tw}(H) \leq k$ (see line 5 and line 9 in Algorithms 1 and 3, respectively). As mentioned above, for constant k, this can be decided in linear time [3]. Since the sizes of $\rho(H)$ and $\rho^{-1}(H)$ are both bounded by the size of DB, the size of C_{l+1} is bounded by a polynomial of the combined size of the set of frequent patterns computed before H, and isomorphism between graphs of bounded treewidth can be decided in polynomial time [2], the conditions in line 9 of Algorithm 3 can be checked in time polynomial in the combined size of DB and the set of frequent patterns listed before H. Finally, by Theorem 3, the subgraph isomorphism (lines 20–21) can be decided in time polynomial in the combined size of H, G, and the set of frequent patterns computed before H. Putting all these together, we get the main result of this paper:

Theorem 4. *For graphs of bounded treewidth, the FCSM problem can be solved in incremental polynomial time.*

5 Concluding Remarks

The main result of this paper is formulated in Theorem 4 above. There are only a few results concerning the complexity of the FCSM problem. In particular, for arbitrary transaction graphs, this problem cannot be solved in output-polynomial time, unless $P = NP$ [10]. If, however, the transaction graphs are restricted to trees then it can be solved in incremental polynomial time [5,11]. The positive result on trees is generalized to outerplanar graphs in [11]. However, this result is shown for a constrained subgraph isomorphism, called BBP subgraph isomorphism, that maps biconnected components to biconnected components and bridges to bridges. Since trees have no biconnected components, BBP subgraph isomorphism is equivalent to subgraph isomorphism between trees. As far as we know, no non-trivial tractable graph classes beyond trees have so far been identified for FCSM problem.

Since subgraph isomorphism remains NP-complete for connected graphs of bounded treewidth, our result provides an example of a frequent pattern mining problem, when the matching operator is NP-hard, but efficient mining is still possible. To the best of our knowledge, existing pattern mining algorithms are all resorted to problems with tractable matching operators. Our result shows that *efficient pattern mining is possible even for NP-hard matching operators.*

Deriving and implementing a practical algorithm from the algorithm described in this paper is an interesting task for future work. This plan is motivated by practical applications, e.g., in computational drug discovery where molecular graphs mostly have treewidth at most 3.

We close the paper with an interesting problem: Can the FCSM problem be solved with *polynomial delay* for graphs of bounded treewidth?

Acknowledgements. We thank one of the anonymous reviewers, Mario Boley, and Kristian Kersting for useful comments on improving the presentation. Tamás Horváth was partially supported by the German Federal Ministry of Economy and Technology under the Theseus Project. He thanks György Turán and Stefan Wrobel for interesting discussions on frequent subgraph mining. Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

References

1. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems on graphs embedded in k-trees. Discrete Applied Mathematics 23, 11–24 (1989)
2. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernetica 11(1-2), 1–22 (1993)
3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing 25(6), 1305–1317 (1996)
4. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoretical Computer Science 209(1–2), 1–45 (1998)
5. Chi, Y., Nijssen, S., Muntz, R.R., Kok, J.N.: Frequent subtree mining – an overview. Fundamenta Informaticae 66, 161–198 (2005)
6. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering 17(8), 1036–1050 (2005)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, San Francisco (1979)
8. Gupta, A., Nishimura, N.: The complexity of subgraph isomorphism for classes of partial k-trees. Theoretical Computer Science 164(1&2), 287–298 (1996)
9. Hajiaghayi, M.T., Nishimura, N.: Subgraph isomorphism, log-bounded fragmentation, and graphs of (locally) bounded treewidth. Journal of Computer and System Sciences 73(5), 755–768 (2007)
10. Horváth, T., Bringmann, B., De Raedt, L.: Frequent hypergraph mining. In: Muggleton, S., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 244–259. Springer, Heidelberg (2007)
11. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. In: Proc. of the 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 197–206. ACP Press, New York (2006)
12. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: On generating all maximal independent sets. Information Processing Letters 27(3), 119–123 (1988)
13. Matoušek, J., Thomas, R.: On the complexity of finding iso- and other morphisms for partial k-trees. Discrete Mathematics 108(1-3), 343–364 (1992)
14. Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of treewidth. Journal of Algorithms 7(3), 309–322 (1986)

Proper Model Selection with Significance Test

Jin Huang[1], Charles X. Ling[2], Harry Zhang[3], and Stan Matwin[1]

[1] School of Information Tech. and Eng., University of Ottawa, Canada
jhuang33@site.uottawa.ca, stan@site.uottawa.ca
[2] Department of Computer Science, The University of Western Ontario, Canada
cling@csd.uwo.ca
[3] Faculty of Computer Science, University of New Brunswick, Canada
hzhang@unb.ca

Abstract. Model selection is an important and ubiquitous task in machine learning. To select models with the best future classification performance measured by a *goal metric*, an *evaluation metric* is often used to select the best classification model among the competing ones. A common practice is to use the same goal and evaluation metric. However, in several recent studies, it is claimed that using an evaluation metric (such as AUC) other than the goal metric (such as accuracy) results in better selection of the correct models. In this paper, we point out a flaw in the experimental design of those studies, and propose an improved method to test the claim. Our extensive experiments show convincingly that only the goal metric itself can most reliably select the correct classification models.

1 Introduction

Model selection is an important task in machine learning and data mining. In classification tasks model selection attempts to select the model with the best future classification performance from (possibly a large number of) competing models. For example, when we build artificial neural networks for face recognition, we may vary the number of hidden nodes and other parameters to build many classification models, and select the best one. Other examples of model selection include choosing the optimal parameter setting for the Support Vector Machines, determining the most suitable amount of pruning in building decision trees, and so on. Clearly, model selection is ubiquitous in machine learning and machine learning applications.

A leading empirical approach for model selection in classification is to use the holdout set, and proceeds as follows: to select the model with the best future classification performance measured by a *goal metric*, competing models are evaluated by an *evaluation metric*, possibly the same as the goal metric, on the holdout set.[1] A natural preference is to use the same metric for both the goal and evaluation. For example, if the goal is to obtain a classification model with

[1] Other empirical approaches are reviewed in Section 2. The goal metric is also called performance metric, and the evaluation metric is called selection metric [1].

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 536–547, 2008.

the highest accuracy on the test sets, accuracy is used to select the most accurate model on the holdout sets. The intuition is that if your child wants to achieve the highest SAT score (the goal metric), you child should train to obtain high SAT scores (the evaluation metric) on (previous) SAT tests (the holdout sets), rather than train to obtain high scores on the GRE test (a different metric).

Rosset [2] recently conducted an empirical research on model selection for binary classification with two specific metrics: accuracy and AUC (Area Under the ROC Curve; see Appendix). He compared the suitability of AUC and accuracy as the model evaluation metrics when the goal metric is accuracy. He showed that AUC can better select the correct models than accuracy even when the goal metric is accuracy. The results are quite surprising and puzzling, and no convincing explanations were given. Nevertheless, several other papers [1,3,4] have since confirmed his finding. For example, Huang and Ling [3] studied model selection with many popular machine learning metrics and claimed that often an evaluation metric different from the goal metric can better select the correct models. Skalak and Caruana [1] used absolute loss to compare model selection abilities of various metrics, and drew similar conclusions. It now seems to be a well-regarded conclusion in the machine learning community that a different metric can do a better job in model selection.

In this paper we point out a potential flaw in the experimental design of these model selection studies: the variance of these metrics when applied to randomly sampled datasets was not taken into consideration. Suppose we have two competing models, X and Y, to be selected by an evaluation metric h. If we say X is better than Y, it should really mean that $E(h(X))$ (E is the expected value or mean of a random variable) is "reliably better" than $E(h(Y))$ on the sampled datasets, rather than simply $E(h(X)) > E(h(Y))$, as in the previous studies.[2] The same is true for the goal metric g. This is because metrics have variances when applied to randomly sampled datasets, and thus, $E(h(X)) > E(h(Y))$ by a minute amount does not really indicate that X is better than Y. Thus, a significance test must be employed in comparison so that the conclusion on which model is indeed better is reliable.

We propose an improved method for proper model selection by incorporating statistical significance tests in the comparison, and carefully re-implementing Rosset's experiments with more datasets and algorithms. Not surprisingly, "AUC can better select models measured by accuracy" is no longer true. We then include more metrics and more model selection approaches, and show convincingly that, *in all cases*, the goal metric itself is the best evaluation metric for model evaluation. We hope that with the proper model selection method proposed here, we can settle this controversial issue once for all.

2 Review of Previous Works

Model selection has been extensively studied by researchers in the machine learning and statistics communities. Here we review several relevant approaches.

[2] We normalize all metrics in this paper so that the larger the value, the better the model.

Rosset [2] performed extensive experiments to study the suitability of AUC and accuracy to select highly accurate models. He assumed that the training set is very small (10% of the original dataset), and thus no data can be used as the holdout sets in model selection. Therefore, he used "the test sets approach" in his study of model selection.

Rosset's experiments are conducted in the following way: First, the original dataset is split randomly into the training set (10%) and the test set (90%). Second, two competing models X and Y (they can be two decision trees with different pruning levels) are built on the training set. Third, X and Y's performance on the test set is estimated. This is done by splitting the test set into 100 equal-sized, stratified[3] subsets, and applying X and Y on the 100 subsets by the goal metric g (i.e., accuracy). The average (mean) g value on the 100 test subsets is denoted as $E(g(X))$ and $E(g(Y))$ respectively. If $E(g(X)) > E(g(Y))$, then model X is regarded as better than model Y on the test set, in terms of the goal metric g. If $E(g(X)) = E(g(Y))$, then they are regarded as the same. Note that no significance test is performed for the comparison, and thus, such conclusion may not be reliable.

In the next step, models X and Y are evaluated by the evaluation metric h (AUC or accuracy) on each of the 100 test subsets. For each subset, if $h(X)$ and $h(Y)$ are consistent with $E(g(X))$ and $E(g(Y))$, then h selects the model correctly. That is, if $h(X) > h(Y)$,[4] and $E(g(X)) > E(g(Y))$, then h selects the right model; otherwise h selects the wrong one. The number of times (among 100) when h selects the correct model is noted, and this represents how well h can select the correct model. The larger the number, the better the evaluation metric in model selection. Using this method, Rosset showed that AUC can better select the correct models than accuracy when the goal metric is accuracy.

Huang and Ling [3] explored the performance of model selection for classification tasks for eight popular machine learning metrics of accuracy, AUC, lift, break-even point, F-metric, average precision, RMS (Root Mean Square) error, and MXE (Mean Cross Entropy) [5] by following Rosset's method (without using the significance test). They showed that generally the metrics of RMS and MXE are the best evaluation metrics, followed by AUC, average precision, and F-metric, no matter what goal metrics are. That is, better model selection can be achieved with a different evaluation metric from the goal metric.

Skalak and Caruana [1] studied the robustness of the evaluation metric when the goal metric is unknown. They used the absolute loss for measuring the model selection error but again no variance is calculated for the loss. They showed that when the models are well calibrated for the probability outputs, and the available holdout data is limited, MXE is the most robust one, while other metrics, such as F-metric, lift and accuracy, performed poorly.

[3] Stratified subsets refer to partitions of a dataset into equal-sized subsets with the same class distribution.

[4] Note that the significance test is not performed here because each subset is just one dataset so there is no variance. But again, if $h(X) > h(Y)$ by a minute amount, it is really not reliable to conclude that X is better than Y.

[5] See Appendix for more details on some of these metrics.

In addition to "the test sets approach" for model selection [2], another (more popular) empirical approach is to use separate holdout sets to estimate model's future performance. We will study this approach extensively later in the paper. Another empirical model selection approach is called complexity-penalization, which assigns a complexity value to each model and chooses the best model that minimizes a predefined trade-off function combining the model complexity and empirical error. Many variants of this approach were proposed, including structural risk minimization [5,6], the minimum description length principle [7], and regularization [8]. Kearns and Mansour [9] theoretically and empirically compared the above two approaches and demonstrated that in some cases the holdout set approach has an advantage of small generalization errors over the complexity-penalization method. In addition, the holdout set approach is very robust and hard to beat compared to other approaches [10,11].

3 Selecting Models Properly with Significance Test

Rosset studied "the test sets approach" of model selection on accuracy and AUC on the UCI dataset "adult" [12] with naive Bayes and k-nearest neighbour (KNN) learning algorithms. In the following subsections, we first replicate Rosset's experiments (without the significance test) but we include more UCI datasets and learning models. We use five UCI datasets: "adult" (included in Rosset's experiments), "letter", "kr-vs-kp", "page blocks", and "pen digits", in our experiment. In the original datasets, only "adult" and "kr-vs-kp" are binary; the rest are multiclass. All multiclass datasets are converted into binary datasets by assigning some classes to the positive class and the rest to the negative class. The properties of the datasets are shown in Table 1. The last column is the ratio of the positive class in the original and converted binary datasets.

We choose three popular learning algorithms in our experiments: decision tree, naive Bayes and k-nearest neighbour (the last two are included in Rosset's experiments). For each learning algorithm we build two competing models with different parameter settings. For the decision tree, we build two trees with and without pruning. For naive Bayes, we use different numbers of attributes in the datasets to train two classifiers. More specifically, we use the first 8 and 10 attributes for the "adult" dataset, first 25 and 35 attributes for "kr-vs-kp", first 10 and 11 attributes for "letter", first 5 and 8 attributes for "page blocks", and

Table 1. UCI datasets used in our experiments

Dataset	Size	Attribute #	Class #	Positive Class Ratio
Adult	30162	14	2	24.8%
Kr-vs-kp	28060	36	2	47.8%
Letter	20000	16	26	38.2%
Page blocks	5473	10	5	10.2%
Pen digits	10992	16	10	30%

Table 2. Ratio of correct model selection with accuracy and AUC using Rosset's method

Dataset	Decision tree		KNN		Naive Bayes	
	accuracy	AUC	accuracy	AUC	accuracy	AUC
Adult	0.74	0.68	0.65	0.93	0.53	0.67
Kr-vs-kp	0.77	0.74	0.88	0.73	0.78	0.75
letter	0.54	0.57	1	1	0.57	0.41
Page blocks	0.72	0.71	0.96	0.51	0.59	0.5
Pen digits	0.65	0.63	0.99	0.88	0.54	0.7

first 8 and 10 attributes for "pen digits". For the k-nearest neighbour, we build two models with $k = 5$ and $k = 50$.

3.1 Model Selection Using Test Sets

We first replicate Rosset's experiments (but with more datasets and learning algorithms) to explore the suitability of accuracy and AUC in model selection when the goal metric is accuracy. (See Rosset's experimental method in Section 2). We report our experimental results in Table 2. Our results confirm Rosset's finding: the ratios measuring correct model selection are higher when AUC is used as the evaluation metric than when accuracy is used, for the adult dataset with KNN and naive Bayes. However, it is surprising that in almost all other cases, accuracy can better select models than AUC. More specifically, in a total of 15 cases (3 learning algorithms and 5 datasets), accuracy is better than AUC in 10 cases, and the same in one case. As Rosset used limited datasets and learning algorithms, his conclusion that AUC can better select models than accuracy when the goal metric is accuracy seems to be unreliable.

Nevertheless, Rosset's experimental design (as well as those in several other studies of model selection [1,3,4]) may lead to unreliable conclusions. That is, it is unreliable to conclude which model is better when a simple comparison is used on two metrics or two means of metrics. To improve the experimental design, we use the significance test (both the t-test and the sign test are studied) in comparison. More specifically, following the notation in Section 2, when deciding which model has a better performance on the 100 test subsets, instead of simply comparing $E(g(X))$ and $E(g(Y))$, a paired t-test with 95% confidence level is performed to see if one is larger, or if they are not statistically significant. To decide which model is better using the evaluation metric h, Rosset simply compared $h(X)$ and $h(Y)$ on each of the 100 test subsets. For one test subset, the t-test cannot be performed. We modify Rosset's approach by re-partitioning randomly the test set into 100 test subsets, and compare the average evaluation scores, $E(h(x))$ and $E(h(Y))$, with the same paired t-test to see if one model is better, or if they are indifferent. If this outcome is the same as the one with the goal metric, then h selects the right model. Notice that we use the same number of subsets (100 here) and the same t-test (both with 95% confidence level) on both evaluation

Table 3. Ratio of correct model selection with t-test using the test sets

Dataset	Decision tree		KNN		Naive Bayes	
	accuracy	AUC	accuracy	AUC	accuracy	AUC
Adult	0.99	0.7	0.97	0.89	0.97	0.08
Kr-vs-kp	0.92	0.37	1	0.83	0.89	0.48
letter	0.97	0.42	1	1	0.93	0.18
Page blocks	0.98	0.62	1	0	0.98	0.19
Pen digits	0.93	0.53	1	0.94	0.93	0.59

and goal metrics. Thus, the "sensitivity" on the variance of the metrics is the same. See more discussions in Section 4.

This process is repeated 100 times and we obtain the number of cases when AUC and accuracy select the right model respectively. The results are shown in Table 3. From the table, we can see that with *all* datasets and *all* learning algorithms, accuracy always does better, and in most cases much better, than AUC (except for one case they tie). Accuracy achieves a very high ratio of correctness (≥ 0.89) while AUCs scores are much lower.[6] For all 15 cases (3 learning algorithms and 5 datasets), AUC tends to choose the wrong model more often (ratios of correctness < 0.5) in 7 cases. These experimental results contradict the claim that AUC performs better than accuracy in model selection using the test sets, after the t-test is employed in comparison.

So far we use the t-test as the significance test in comparing means of metrics to draw reliable conclusion about model selection. Here we show that another popular significance test, the sign test [13] [7], is equally effective in proper model selection. Table 4 lists the outcome of correct model selection using the sign test, instead of the t-test as in Table 3. The results are similar to Table 3: in all cases, accuracy can better select more accurate models than AUC. We believe that the choice of the significance test does not matter, as long as the same significance test is applied on both the evaluation and goal metrics obtained from the sampled datasets. In the rest of the paper, only results with the t-test is reported.

3.2 Model Selection Using Holdout Sets

In this section we perform experiments using the holdout set approach of model selection with the proper significance test. In this approach, the original dataset is split randomly into three non-overlap sets: the training set (10%), the holdout

[6] Note that due to sampling variations between training, holdout, and test sets, occasionally accuracy may select the wrong model with the highest accuracy. Also, due to the consistency between accuracy and AUC (see Section 4), AUC may sometimes also choose the most accurate model. Accuracy is simply more likely than AUC in selecting the most accurate models.

[7] The sign test is a non-parametric test used to compare the distribution median with a given pair of data. This test could be used as an alternative for one-sample Student t-test. Unlike the t-test, the sign test can work with non-normal distributions.

Table 4. Ratio of correct model selection with sign test using the test sets

Dataset	Decision tree		KNN		Naive Bayes	
	accuracy	AUC	accuracy	AUC	accuracy	AUC
Adult	0.96	0.64	0.94	0.77	0.92	0.19
Kr-vs-kp	0.93	0.42	1	0.78	0.93	0.52
letter	0.94	0.33	1	1	0.90	0.24
Page blocks	0.95	0.67	1	0	0.97	0.21
Pen digits	0.88	0.54	1	0.92	0.94	0.52

set (45%) and the test set (45%). The holdout set and the test set are further partitioned into 50 subsets. Then two competing models are built on the training set, and they are applied to the 50 test subsets by the goal metric g. Their averages are compared with the paired t-test. The same is applied to the 50 holdout subsets by the evaluation metric h. If the two outcomes are the same, then h selects the correct model. The process is repeated 100 times and the number of cases of correct model selection is noted.

The results are shown in Table 5. We can see again that *in all cases* accuracy better selects the correct models than AUC (except for one case when the two tie). These results show that accuracy is again much better in selecting the correct model than AUC if the goal metric is accuracy using the holdout set approach.

We note that the numbers in Table 5 (using the holdout sets) are generally smaller than the corresponding ones in Table 3 (using the test sets). We believe that this is because in the holdout set approach, there is no overlap between the holdout set and the test set. Thus, it is less likely to select the right models determined by the test set using the holdout set.

3.3 Model Selection with More Metrics

In this section we explore the model selection ability of other popular metrics used in machine learning. For example, Root Mean Squared error (RMS) is widely used in regression to reflect the average deviation of the predicted values from the true ones. F-measure combines precision and recall, often used in information retrieval. Here we choose five commonly used metrics as both evaluation

Table 5. Ratio of correct model selection with t-test using the holdout sets

Dataset	Decision tree		KNN		Naive Bayes	
	accuracy	AUC	accuracy	AUC	accuracy	AUC
Adult	0.97	0.62	0.96	0.81	0.76	0.12
Kr-vs-kp	0.83	0.44	1	0.86	0.73	0.56
letter	0.74	0.47	1	1	0.65	0.19
Page blocks	0.88	0.61	1	0	0.78	0.13
Pen digits	0.88	0.56	1	0.9	0.55	0.54

Table 6. Average ratio of correct model selection using test sets

Goal ⇓ Evaluation ⇒	accuracy	AUC	F	RMS	MXE	Average
accuracy	0.96*	0.46	0.62	0.57	0.47	0.57
AUC	0.42	0.97*	0.32	0.77	0.74	0.7
F	0.6	0.31	0.97*	0.43	0.35	0.39
RMS	0.63	0.75	0.44	0.94*	0.77	0.78
MXE	0.49	0.74	0.34	0.78	0.98*	0.8
Average	0.6	0.68	0.37	0.78	0.81	0.96*

and goal metrics. The five metrics are: accuracy, AUC, F-measure, Root Mean Squared error (RMS), and Mean Cross Entropy (MXE). Definitions of these metrics can be found in Appendix.

Caruana and Niculescu-Mizil [14] suggested that when the goal metric is unknown during model construction and model selection, one could use the average of several different metrics as the goal metric. Here we use the average of the five single metrics mentioned above, and use it as both the goal and evaluation metrics. Thus, there is a total of six metrics in this experiment.

We perform the same model selection experiments with the t-test as in Sections 3.1 and 3.2. Each of the six metrics is used for the goal metric as well as for the evaluation metric. The results of the test set approach are shown in Table 6, with the goal metrics in the row and evaluation metrics in the column. Each number in the table is the average of 15 scores for that pair of metrics over five datasets and three learning algorithms. For example, 0.96 in Table 6 for accuracy to select accuracy is the average of 15 numbers in Table 3 for accuracy to select accuracy over five datasets and three learning algorithms. We put a * next to the largest number in each row to indicate that not only the average (of the 15 scores) is the largest, but each individual score is also larger or the same compared to another evaluation metric (as in the case for accuracy and AUC in Table 3). We can see that numbers in the diagonal line are the largest in each row, and each largest number has a * next to it. This suggests that in general when the evaluation and goal metrics are the same, correct models can always be more reliably selected. This is also true for the average of the five single metrics: when the goal metric is the average of the five metrics, using the same metric itself (the average) as the evaluation metric is best for model selection, compared to any other single metric. The results using the holdout sets are similar, as seen in Table 7. These results generalize the conclusion we obtain in the previous sections regarding accuracy and AUC. It shows convincingly that we should always use the same metric for evaluation and goal in model selection.

Although we can use the average of several metrics as a robust goal metric when the goal metric is unknown, sometimes we must choose a single metric for model construction, optimization, and selection. Tables 6 and 7 can also tell us which single metric is best if the goal metric is the average. From the bottom row of the two tables, we can see that, for both the test set approach and the holdout set approach, MXE has the largest ratio for correct model selection, followed by RMS, AUC, accuracy, and last, F. This means that MXE is the

Table 7. Average ratio of correct model selection using holdout sets

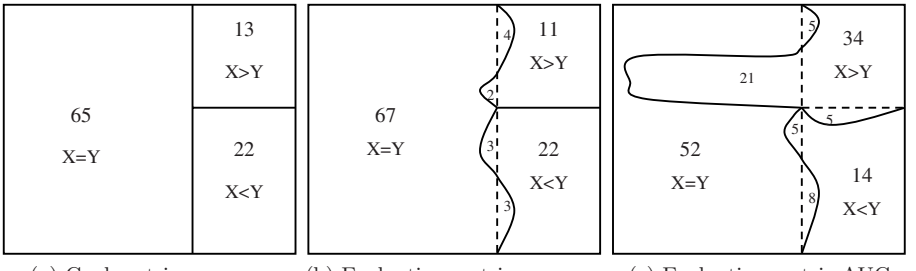
Goal ⇓ Evaluation ⇒	accuracy	AUC	F	RMS	MXE	Average
accuracy	0.85*	0.52	0.66	0.61	0.47	0.63
AUC	0.51	0.87*	0.4	0.8	0.77	0.72
F	0.67	0.39	0.91*	0.52	0.54	0.48
RMS	0.58	0.76	0.49	0.89*	0.81	0.8
MXE	0.43	0.76	0.52	0.8	0.9*	0.83
Average	0.59	0.72	0.5	0.8	0.82	0.89*

most robust metric for model evaluation if the goal metric is the average. This confirms with the conclusion of [1]. Only in this situation should we use a metric (such as MXE) different from the goal metric (such as the average) in model selection.

4 Discussion

In this section we investigate why contradicting conclusions on accuracy and AUC can be drawn with and without the t-test. We provide a detailed analysis for selecting the more accurate model by accuracy or AUC from two competing decision trees on the "pen digits" dataset using the holdout sets (as discussed in Section 3.2). We denote the two competing decision tree models as X and Y. Among 100 repeated runs (cases) of model selection with the t-test, there are 13 cases when X is better than Y, 65 cases when X equals to Y, and 22 cases when X is worse than Y, all evaluated by accuracy with the t-test on the test sets. This is regarded as the "ground truth". We depict this distribution roughly in Figure 1(a).

We then count how each of the three outcomes ($X > Y$, $X = Y$, and $X < Y$) is classified by the evaluation metrics of accuracy and AUC respectively with the t-test. The results are depicted in Figure 1 (b) and (c) respectively. From



(a) Goal metric accuracy (b) Evaluation metric accuracy (c) Evaluation metric AUC

Fig. 1. The distribution of cases that goal metric accuracy, evaluation metric accuracy and AUC evaluate models X and Y

Figure 1 (b), we can see clearly that when accuracy is used to select models, there is a small number of confusing cases between $X > Y$ and $X = Y$ (6 cases), and between $X < Y$ and $X = Y$ (6 cases). This is again due to sampling variations in the training, holdout, and test datasets. There is no confusing case between $X > Y$ and $X < Y$. However, when AUC is used as the evaluation metric, we can see from Figure 1 (c) that there is a huge number of confusing cases between $X > Y$ and $X = Y$ (26 cases), and the number of confusing cases between $X < Y$ and $X = Y$, and between $X > Y$ and $X < Y$ are also larger (13 cases and 5 cases respectively).

The explanation is that AUC and accuracy *are* different metrics. As shown in [15], although AUC and accuracy are largely consistent, there are cases when accuracy and AUC contradict each other. Furthermore, AUC is more "sensitive" (or discriminating) than accuracy [15]. That is, AUC often treats two objects with the same accuracy as different. This implies that AUC is likely to be "too sensitive" in comparing two objects when their accuracy values are statistically indifferent. We can see this from Figure 1 (c): there is a large number of $X = Y$ cases (21 cases) being identified as $X > Y$.

One might argue that overly sensitive metric may not be a problem if we only care about correctly identifying *different* models. That is, if two models are statistically different, we must identify the better one; but if they are not, it will not hurt if we say one of them is better. Under this assumption, if we calculate the correct "recall" of the $X > Y$ and $X < Y$ cases (a total of $13 + 22 = 35$ cases), we can see from Figure 1 (b) that the recall with accuracy is $(13 - 4) + (22 - 3) = 28$. On the other hand, from Figure 1 (c), the recall with AUC is $(13-5)+(22-8-5) = 17$. Thus, under the assumption that we only care about correctly identifying the statistically different models, accuracy is still better than AUC for this dataset. If one assumes that correctly identifying if a model is better or indifferent statistically is equally important in model selection, our results in previous sections show that in all cases, the goal metric should be used to select the right models, and such model selection outcomes (with the significance test) are reliable.

From this analysis, we can conclude that different metrics may contradict each other, and may have different sensitivity. Without the significance test in model selection, it is not reliable to select better models, and such a study may lead to the wrong conclusion that an evaluation metric different from the goal can better select models. When the paired t-test or sign test is used in comparing models' performance, we conclude convincingly that we should always use the goal metric to evaluate and select models.

5 Conclusion

In this paper we investigate model selection with different metrics. We first point out a flaw in the experimental design in several previous studies which led to an erroneous conclusion that a different evaluation metric can better select models. The problem lies in the lack of significance test when comparing competing

models. This may result in statistically indifferent models being regarded as different, and vice versa. With the proper use of the significance test (such as the t-test and the sign test) in model selection, we show convincingly that in all cases (with six metrics, three learning algorithms, five UCI datasets, and using the test sets approach or the holdout sets approach), the same goal metric is the best evaluation metric for model selection.

Occasionally the goal metric and evaluation metric cannot be the same. For example, the goal metric may be unknown during model selection. We will study this problem further in our future work.

References

1. Skalak, D.B., Niculescu-Mizil, A., Caruana, R.: Classifier loss under metric uncertainty. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 310–322. Springer, Heidelberg (2007)
2. Rosset, S.: Model selection via the AUC. In: Proceedings of the 21st International Conference on Machine Learning (2004)
3. Huang, J., Ling, C.: Evaluating model selection abilities of performance measures. In: Proceedings of the Workshop on Evaluation Methods for Machine Learning at the 21st National Conference on Artificial Intelligence (AAAI 2006) (2006)
4. Wu, S., Flach, P., Ferri, C.: An improved model selection heuristic for AUC. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 478–489. Springer, Heidelberg (2007)
5. Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer, New York (1982)
6. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
7. Rissanen, J.: Stochastic complexity and modeling. Annals of Statistics 10(3), 1080–1100 (1986)
8. Moody, J.E.: The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In: Advances in Neural Information Processing Systems — 4 (NIPS 1992) (1992)
9. Kearns, M., Mansour, Y., Ng, A.Y., Ron, D.: An experimental and theoretical comparison of model selection methods. Machine Learning 27, 7–50 (1997)
10. Efron, B.: Computers and the theory of statistics: Thinking the unthinkable. SIAM Review 21, 460–480 (1979)
11. Weiss, S., Kulikowski, C.: Computer Systems that Learn: classification and prediction methods from statistics, neural networks, machine learning, and expert systems. Morgan Kaufmann, San Mateo (1991)
12. Blake, C., Merz, C.: UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
13. Weisstein, E.W.: Fisher sign test. MathWorld–A Wolfram Web Resource
14. Caruana, R., Niculescu-Mizil, A.: Data mining in metric space: An empirical analysis of supervised learning performance criteria. In: Proceedings of the 10th ACM SIGKDD conference (2004)

15. Ling, C.X., Huang, J., Zhang, H.: AUC: a statistically consistent and more discriminating measure than accuracy. In: Proceedings of 18th International Conference on Artificial Intelligence (IJCAI-2003), pp. 519–524 (2003)
16. Hand, D.J., Till, R.J.: A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning 45, 171–186 (2001)

Appendix: Metrics Used in This Paper

Accuracy: Accuracy is the most commonly used performance metric in Machine Learning. For a classification task, accuracy is the percentage of the correctly classified examples in all examples.

F-metric: F-metric combines precision and recall as a single metric. It is defined as the harmonic mean of the precision and recall.

$$F = \frac{2 * precision * recall}{precision + recall}$$

AUC: The Area Under the ROC Curve, or simply AUC, is a single-number metric widely used in evaluating classification algorithms. AUC reflects the overall ranking performance of a classifier. For a binary ranked list, Hand and Till [16] present the following simple formula to calculating AUC

$$AUC = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1}$$

where S_0 is the sum of the ranked positions of all positive examples. n_0 and n_1 are the numbers of positive and negative examples.

RMS: Widely used in regression, RMS (Root Mean Square error) reflects the average deviation of all predicted values from the true values. For K instances, suppose that the true probability value and the predicted probability value for an instance I_i are $Tar(I_i)$ and $Pred(I_i)$,

$$RMS = \sqrt{\frac{1}{K}\sum_{i=1}^{K}[Tar(I_i) - Pred(I_i)]^2}$$

MXE: MXE (Mean Cross Entropy) is used to measure in average how close all predicted probabilities are to the true probabilities. It can be shown that minimizing the cross entropy gives rise the maximum likelihood hypothesis. When using the same notations as in RMS, MXE is defined as

$$MXE = -\frac{1}{K}\sum_{i=1}^{K}\{Tar(I_i) * log[Pred(I_i)] + (1 - Tar(I_i)) * log[1 - Pred(I_i)]\}$$

A Projection-Based Framework for Classifier Performance Evaluation

Nathalie Japkowicz[1], Pritika Sanghi[2], and Peter Tischer[2]

[1] School of Information Technology and Engineering
University of Ottawa, Canada
nat@site.uottawa.ca
[2] Clayton School of Information technology,
Monash University, Melbourne, Australia
{Pritika.Sanghi,Peter.Tischer}@infotech.monash.edu.au

Abstract. In this paper, we propose approaching the problem of classifier evaluation in terms of a projection from a high-dimensional space to a visualizable two-dimensional one. Rather than collapsing confusion matrices into a single measure the way traditional evaluation methods do, we consider the vector composed of the entries of the confusion matrix (or the confusion matrices in case several domains are considered simultaneously) as the performance evaluation vector, and project it into a two dimensional space using a recently proposed distance-preserving projection method. This approach is shown to be particularly useful in the case of comparison of several classifiers on many domains as well as in the case of multiclass classification. Furthermore, by providing simultaneous multiple views of the same evaluation data, it allows for a quick and accurate assessment of classifier performance.

1 Introduction

Performance evaluation in supervised classification is traditionally performed by considering the confusion matrices obtained from test runs of several classifiers on various domains, collapsing each matrix into a value (e.g., accuracy, F-measure), and comparing these values to each other. One issue with this approach is that, by the time the classifiers' performances get compared to one another on a given domain, the details of the confusion matrices have been lost. The comparison only involves a single number, be it the accuracy or F-measure of the classifiers. The problem is compounded if the comparison involves several domains, and, when dealing with multi-class rather than binary domains.

In order to defray this problem, people sometimes use pairs of values on which to base their comparisons. Precision/Recall and Sensitivity/Specificity are two commonly used pairs. While this alleviates the problem, somewhat, by providing additional information about the confusion matrix, it makes the comparison of classifiers more complex since it creates cases where one classifier obtains good results on one component and bad ones on the other, while the second classifier

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 548–563, 2008.

obtains opposite results. Furthermore, such pairs of values do not apply to multi-class domains, and the problem of how to aggregate the results obtained on various domains remains as well.

The purpose of this paper is to propose a different way to view the performance evaluation problem with the hope of addressing these issues while offering a more generalized vision of the overall problem. In particular, we can view classifier evaluation as a problem of analyzing high-dimensional data, recognizing that the performance measures currently used by the data mining community are but one class of projections that could be applied to these data. If we think of our current measures as specialized projection methods, we can then generalize the procedure by considering the fact that any projection method (standard or not) could be applied to our highly dimensional performance data, along with any distance measure (once again, standard or not). Such an approach could open up the field of classifier evaluation by allowing us to both organize and classify the existing measures within this new framework and, more importantly, to experiment with a variety of new approaches in a more systematic way. A particular benefit of this framework is the fact that projection approaches are typically intended for visualization, which is useful in that it permits both a quick assessment of the results by a human-being and the compounding of more information into the representation than in the case where a single or a pair of values are issued. This, by the way, is in line with more recent evaluation methods such as ROC Analysis [1] and cost-curves [2] which also suggest a move towards visual approaches.

The research presented in this paper demonstrates the kind of classifier performance evaluation strategies that can be derived from the consideration of this generalized framework. This paper focuses on three particular advantages brought on by this framework:

- An approach for aggregating the results of a classifier on several domains;
- An approach for dealing with multiclass domains;
- An approach for the quick generation of easily interpretable multiple views of classifier performance

Please note that this paper restricts itself to a small number of options with respect to the projection approaches, distance functions and result data representation that could be used, with the understanding that future work will explore these possibilities further. It is also important to note that although we focus on the evaluation of supervised classification algorithms, here, our approach is universal and could be applied to any performance evaluation problem domain.

The remainder of the paper is organized as follows: Section 2 details our framework and its particular implementation we adopted in this paper. In particular, we discuss the kind of performance data we use as a starting point, the distance measures considered, as well as the projection method we selected. The purpose of Sections 3 and 4 is to demonstrate the aggregation properties of our framework. In particular, Section 3 illustrates our approach in the case where a number of classifiers are compared on several domains simultaneously while

section 4 considers the case where these same classifiers are compared on a single multi-class problem. In both sections, we highlight the particular advantages of our technique. Section 5 offers a brief discussion of how our method can be used as a multi-facetted approach to classifier performance evaluation. Section 6 concludes the paper, and discusses potential extensions for future work.

2 The Framework and Its Implementation

As discussed in the introduction, current evaluation methods can be viewed as specialized projections from a high-dimensional to a 1-dimensional space, in the case of Accuracy, F-Measure and AUC, and to a two-dimensional space, in the case of Precision/Recall and Sensitivity/Specificity. In this work we generalize this idea by suggesting that the techniques proposed in the field of visualization can be put to the service of classifier evaluation as well. In particular, we propose to use the projection techniques and distance measures in use in that field for our purpose. We begin by discussing the general methodology we adopted, and then move on to addressing the issue of choosing an appropriate projection method.

2.1 General Methodology

The visualization approach we propose works according to the following steps:

1. All the classifiers involved in the study are run on all the domains considered, and the corresponding performance matrices (be they confusion matrices, performance vectors of the outcome on each testing point, etc...) are saved.
2. The performance matrices associated with one classifier on each domain are organized into a single vector. The process is repeated for each classifier such that there is a pairwise correspondence of each vector component from one classifier to the next one.
3. A distance measure is chosen to represent the distance between two vectors in high-dimensional space.
4. A projection method is chosen to project the vectors into a two-dimensional space.[1]
5. The distance measure and projection method are used on the vectors generated in Step 2.

The traditional approach to classifier performance evaluation is compared to our new approach in Figure 1. As shown in that figure, in the traditional approach, the performance value of a classifier is calculated on each domain, be it binary or multiclass. These values are then aggregated together into an overall performance value, that gets compared from classifier to classifier. In the new approach, the data pertaining to a classifier is preserved into its original form and simply concatenated into a vector. The transformation is delayed until the projection is applied. This means that in our approach, information is lost in a

[1] Three or four dimensions could also be used, if that could be helpful.

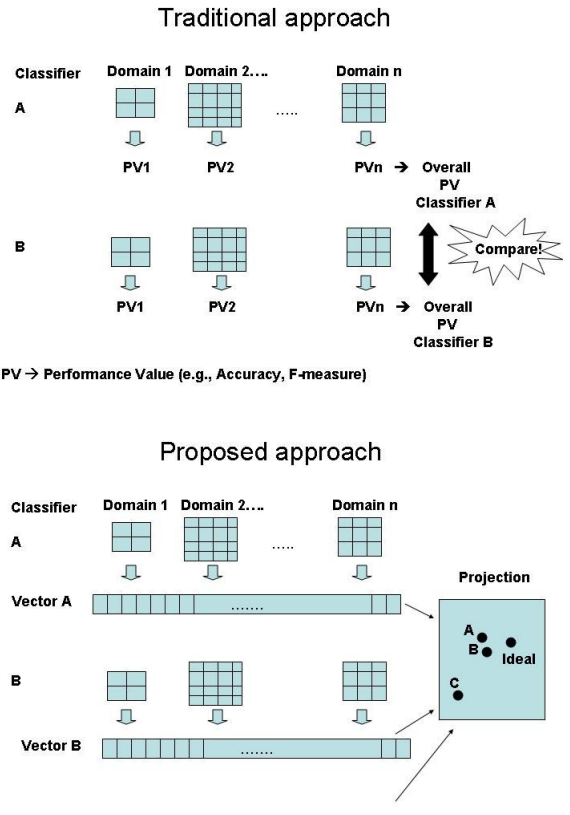


Fig. 1. The Traditional and Proposed Approaches to Classifier Performance Evaluation

single spot: the projection phase. In the traditional approach information is lost whenever any kind of aggregation occurs.[2]

If we consider the performance of several classifiers on a single binary domain, there are two advantages provided by our new framework. First, it decomposes the problem in a principled manner, separating the issue of projection from that of choosing an appropriate distance measure along which to compare the data. Secondly, by going from a projection to a one-dimensional space to a projection to a two-dimensional one[3], it allows for two rather than one relationships to be established. In the traditional approach which, typically, projects the performance data into a single dimension, the classifiers can only get ranked according

[2] Note, however, that since, in both the traditional approaches and our approach, as considered in this paper, we take as a starting point the confusion matrix—an aggregated form of result—, information has been lost even before either performance evaluation approach is used.

[3] Even though the Precision/Recall and Specificity/Sensitivity approaches allow for a two-dimensional projection, the projected data is typically treated as two 1-dimensional projections rather than one 2-dimensional projection.

to their similarity to the ideal classifier. In our evaluation framework, the addition of a dimension allows the classifiers not only to be ranked according to the ideal classifier, but also, to be compared to one another.

A third key advantage over the traditional approach concerns the aggregation of classifier results over different domains. It is common for researchers to simply average the results obtained by a classifier over different domains. This is a mistake when dealing with multi-class classification problems since the same value has different meanings depending on the number of classes. Recognizing this problem, researchers sometimes use a win/tie/loss approach, counting the number of times each classifier won over all the others, tied with the best or lost against one or more. This approach, however, loses any kind of information pertaining to how close classifiers were to winning or tieing. Our approach does not suffer from either of these problems since the entries of each performance vector are compared, in a pairwise fashion, from classifier to classifier.

Please, note that if an unweighted distance measure is used in the projection method, then each matrix entry is given the same importance, but this can be changed by weighting the measure appropriately.

2.2 Implementation Details

Several points considering the implementation of our approach need to be clarified. First, it is important to note that the vectors representing each classifier can take different formats. They can, simply, be 4-dimensional vectors containing all the entries of the confusion matrix on a single binary domain, 9 dimensional vectors containing all the entries of the confusion matrix on a single 3-class domain, and so on. As well, they can be formed by the confusion matrices obtained by a single classifier on several domains, be they multi-class or binary domains. It is also possible, rather than representing the confusion matrices, to represent the classifiers' outcome on each point of the testing set. The graph of Figure 2 in the next subsection is an example where such a representation was used.

Second, we must specify what distance measure and projection approach we selected for implementing the method. The distance measures can take several forms, each with different properties. The Euclidean distance (L2 norm), for example, considers all the performance data equally, though it penalizes more for the presence of a few extreme differences than for the presence of several small differences. The Manhattan distance (L1 norm) attaches less importance to large differences. Other distance measures can weigh different components differently. For example, true positives can be given more importance than true negatives (similarly to Precision). In a multi-class domain, a distance measure can focus on the performance of one class, grouping all the other classes, and so on. In fact all the biases provided by the traditional measures (accuracy, precision, recall, F-measure and so on) can be reproduced in our framework. In our particular study, the main distance function we will consider is the Euclidean distance. However, the Manhattan Distance will be discussed briefly in Section 5.

Third we must discuss our choice of a projection approach. In this work we considered two methods: Principal Component Analysis (PCA)/Multi-Dimensional

Scaling (MDS)[3][4], a linear projection, and the Minimal Cost Spanning Tree Projection (MCST), a non-linear distance-preserving projection approach, recently proposed by [4,5]. The second approach, in addition to being non-linear, has the advantage of guaranteeing *theoretically* that the distance from each point to at least one of its nearest neighbours is preserved. Having plotted a number of graphs using PCA/MDS and MCST and compared their results, we found that in most cases, the two projections yield similar information and that, therefore, this theoretical guarantee does not have much practical bearing on our study. There were a few situations, however, where this was not the case. For example, when plotting the PCA projection of the outcome of the classification by eight different classifiers on *all the data*[5] contained in three UCI [6] domains: `Breast Cancer`, `Labour` and `Liver`, we found a disparity between PCA and MCST. This is shown in Figure 2 where the information provided in the plot produced by PCA, the top plot in Figure 2, is misleading since classifier *IBk*'s closeness to the ideal classifier is not warranted (See Table 2, later on in the paper). This does not happen with the MCST projection, whose graph is shown at the bottom of Figure 2, where the ideal classifier is represented by label "1" and where no classifier in the graph is close to ideal, as demonstrated by the long length of all the graph's broken lines.[6] To sum up, despite the fact that we could probably have used PCA/MDS for our experiments quite safely, we decided to adopt MCST both because of its novelty and greater theoretical soundness and because of the off-chance that we could run into a situation of the kind depicted in Figure 2. The detailed description of the MCST projection method follows in the next subsection. PCA and MDS are not described since they were not adopted and since they are well-known projection approaches.

2.3 A Distance-Preserving Projection Approach

Our approach is a slight variation on an approach by [4,5]. It is described as follows:

Let $d(x, y)$ represent the distance between x and y in the original higher dimensional space; let $P(x)$ and $P(y)$ be the projections of x and y onto the two-dimensional space; and let $d_2(P(x), P(y))$ represent the distance between the projected points in a two-dimensional space. In this case, we are projecting the performance of the classifiers p_i, where $i = 1, 2, \ldots n$. We introduce the ideal classifier as p_0. p_0 is mapped to the origin.

Find the classifier which is closest to ideal, p_1, and put this on the y-axis at $(0, d(p_0, p_1))$.

For the remaining classifiers, at each stage we find the classifier, p_i, which is nearest to the classifier which has just been plotted, p_{i-1}. When plotting p_i we want to preserve two constraints:

[4] PCA is equivalent to Multi-Dimensional Scaling (MDS)[3] in our setting since the use of the Euclidean distance makes the results of the two approaches indistinguishable.

[5] i.e., Not simply the confusion matrix, but the vector containing the outcome of each and every instance contained in the testing set of each domain.

[6] See Sections 2.3 and 3 for a more precise description of the MCST graphs.

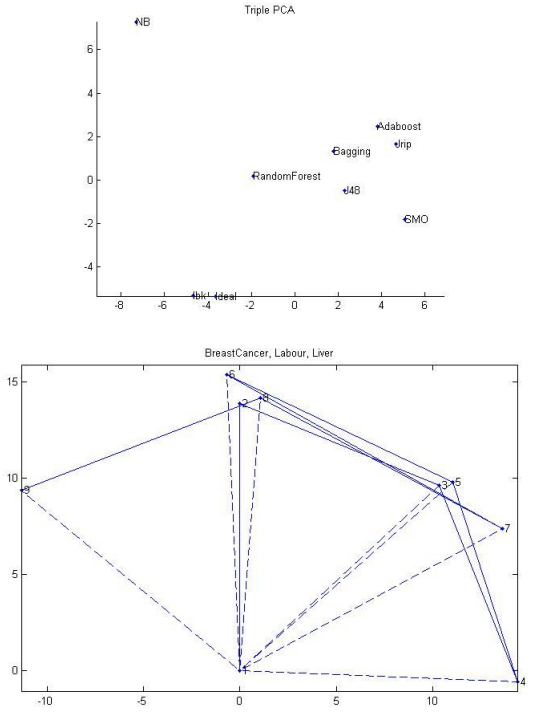


Fig. 2. Top: The PCA/MDS projection of three binary domains represented by the outcome of the classifiers on each data point; Bottom: The MCST projection of the same domains and classifiers

$$d_2(P(p_i), P(p_{i-1})) = d(p_i, p_{i-1}) \tag{1}$$

i.e. we want the projections of p_i and p_{i-1} to be the same distance apart as p_i and p_{i-1}.

We also want to satisfy the second constraint:

$$d_2(P(p_i), P(p_0)) = d(p_i, p_0) \tag{2}$$

i.e. we want the projection of the ideal classifier and the projection of p_i to be the same distance apart as the classifiers are. This means that in the projected space the distance from the origin is a measure of how close the classifier is to ideal. The better the classifier, the closer its projection will be to the origin.

Most times there will be two possible positions for $P(p_i)$ which satisfy both constraints. When there is a choice of solutions, the solution is chosen to satisfy a third constraint as closely as possible:

$$d_2(P(p_i), P(p_{i-2})) = d(p_i, p_{i-2}) \tag{3}$$

Whereas we choose p_i to be the point which has not yet been projected which is closest to the most recently projected point, the original algorithm by [4,5]

chooses p_i to be the point which has not yet been projected and which is closest to *any* of the points which have already been projected. The original approach projects the points in the same order as Prim's algorithm would add the points to a Minimal Cost Spanning Tree. Both approaches were tried, but we preferred the results produced by the modified approach because it seemed to separate clusters more.

Please, note that in our graphs we have found it useful to draw lines between pairs of projected points to show that the distance between the projected points is equal to the distance between the points in the original, higher dimensional space. Dotted lines connect projected points to the original and indicate the exact distance in the higher dimensional space from the classifier to the ideal classifier. Unbroken lines connect a point to the point that was projected immediately before it in the projection order. The distance between these projected points is also identical to the distance between the points in the original space.

When looking at the projected points, it is useful to remember that the triangle formed by $P(p_0)$, $P(p_{i-1})$ and $P(p_i)$ is congruent to the one formed by p_0, p_{i-1}, and p_i.

3 Experiments on Multiple Binary Domains

In this part of the paper, we experiment with the use of our approach on multiple domains. The three domains considered are all from the UCI Repository for Machine Learning and are: `Breast Cancer`, `Labour` and `Liver`. This means that we are projecting vectors of size 12 (3 confusion matrices of 4 entries each) into a two dimensional domain. Eight different classifiers were compared in this study: *Naive Bayes (NB), C4.5 (J48), Nearest Neighbour (Ibk), Ripper (JRip), Support Vector Machines (SMO), Bagging (Bagging), Adaboost (Adaboost)* and *Random Forests (RandFor)*. All our experiments were conducted using Weka [7] and these particular algorithms were chosen because they each represent simple and well-used prototypes of their particular categories. The results we report were obtained using 10-fold stratified cross-validation. It is worth noting that since the purpose of all our experiments was to interpret the results produced by our evaluation method and not to optimize performance, default settings of Weka were used throughout the paper. The significance of this work, thus, does not lie in the results we obtain, which should only be seen as illustrative of the evaluation framework we propose, but rather on the introduction of the evaluation framework, itself.

The results of our approach are presented in Figure 3 and its companion table, Table 1.

The results show that all the methods, except for *SMO* (8) and *NB* (9), fall within the same range. *SMO* and *NB* produce much worse results, since they are further away from *Ideal (1)* than the other approaches; and are shown to behave very differently from one another as well, since they are not clustered together. To better understand the graph, we consider this result in view of the results obtained by the traditional measures of performance that are displayed in Table 2, for the three domains considered.

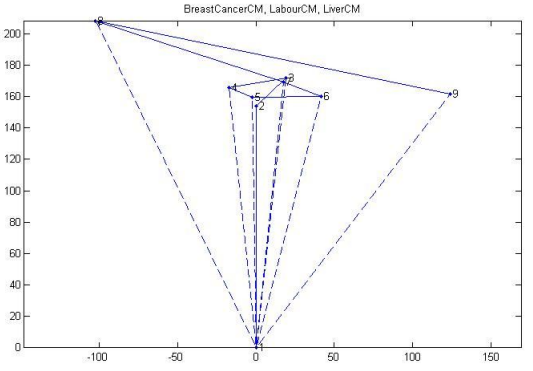


Fig. 3. Projection of Three Binary Domains

Table 1. Legend for Figure 3

Three Binary Domains Projection Legend			
Classifier number	Classifier name	Distance from origin	Distance from previous classifier
1	Ideal	0	
2	RandFor	154	
3	Ibk	173	26
4	JRip	167	37
5	Adaboost	160	16
6	Bagging	166	44
7	J48	170	26
8	SMO	232	126
9	NB	203	230

This comparison tells us something interesting: *SMO* fails quite miserably according to all three measures (Accuracy, F-measure and AUC) on the `Liver` data set. *NB*, on the other hand, only fails badly on this domain when accuracy is considered. The F-Measure and AUC do not signal the presence of a problem. This means that, unless accuracy were considered, we would not have detected a difference in the behaviour of *NB* on the `Liver` data set. In contrast, our method identified both the problems with *NB* and *SMO* and stated that they were of a different nature. Our method seems to warn us that these two classifiers are sometimes unreliable, whereas the other systems are more stable. Of course, if we had used a different distance measure, the results would have been different. The purpose of our discussion is not so much to compare Euclidean distance to accuracy, F-measure and AUC. Instead, we wish to point out how differences between classifiers are clearly and immediately noticeable from our graph.

Please note that *SMO*'s lower performance on the Liver data is something that would not have been picked up (except possibly if the F-measure had been considered) by an averaging of performance on all domains since *SMO* gets

Table 2. Performance by Traditional Measures on the Breast Cancer (BC), Labour (La) and Liver (Li) domains

		Accuracy	F-Measure	AUC
NB	BC:	71.70	0.48	0.70
	La:	89.50	0.92	0.97
	Li:	55.40	0.60	0.64
J48	BC:	77.50	0.40	0.59
	La:	73.70	0.79	0.7
	Li:	68.70	0.59	0.67
Ibk	BC:	72.40	0.41	0.63
	La:	82.50	0.86	0.82
	Li:	62.90	0.56	0.63
JRip	BC:	71	0.43	0.60
	La:	77.20	0.83	0.78
	Li:	64.60	0.53	0.65
SMO	BC:	69.60	0.39	0.59
	La:	89.50	0.92	0.87
	Li:	58.30	0.014	0.50
Bagging	BC:	67.8	.23	.63
	La:	86	0.90	0.88
	Li:	71	0.624	0.73
Adaboost	BC:	70.30	0.46	0.70
	La:	87.70	0.91	0.87
	Li:	66.10	0.534	0.68
RandFor	BC:	69.23	0.39	0.63
	La:	87.70	0.91	0.90
	Li:	69	0.64	0.74

averages of: 72.46% in accuracy, .44 in F-measure and .65 in AUC versus 74.7% accuracy, .64 in F-measure and .75 in AUC, for *Adaboost* (5), quite a good classifier on these domains. Once its performance results averaged, NB would not have exhibited any problem whatsoever, no matter which traditional evaluation method were considered. Indeed, it produced averages of: 72.2% for accuracy, .67 for the F-measure, and .77 for the AUC, three results that are comparable to those obtained by *AdaBoost*, our reference. Once again, what is remarkable about our visualization approach is that the graph of Figure 3 tells us immediately that an abnormal situation has been detected with respect to *SMO* and *NB* and that this problem is of a different nature in each case. It does not tells us what the problem is, but it warns us of that problem in a quite effective way. This is quite useful given how tedious and mistake-bound the reading of large result tables can be. Our approach can be used to filter out problem spots, that can then be carefully analyzed, using only the portion of the result tables that focus on this problem spot.

Though we only used binary domains in this example, we could have, instead, mixed binary and multi-class domains using the same approach, thus finding a

Table 3. Accuracies on the Anneal Data Set

NB	J48	Ibk	JRip	SMO	Bag	AdaBoost	RandFor
86.30	98.40	99.10	98.30	97.40	98.20	83.60	99.30

way to aggregate values that could not, otherwise, be aggregated together. The next section discusses the case of multi-class domains in detail.

4 Experiments on Single MultiClass Domains Using Confusion Matrices

In this section, we consider how our approach fares on multiclass domains. In particular, we consider the `Anneal` domain from UCI. `Anneal` is a 6-class domain (though one of the classes is represented by no data point). The data set is quite imbalanced since the classes contain 684, 99, 67, 40, 8 and 0 instances, respectively. The results obtained on this domain are displayed in Figure 4 along with its companion table, Table 4. This time, the graph encourages us to beware of *NB* (8) and *Adaboost* (9), though it also shows us that *Adaboost* and *NB*'s problems are not related. We compare the results of Figure 4 to the accuracy results obtained on this domain, displayed in Table 3.

While the accuracies (one of the few simple compact measures that can be used in multi-class domains) suggest that *NB* and *Adaboost* do not classify the data as well as the other domains, it does not inform us of the fact that these two classifiers approach the problem differently. Indeed, while it is true that *NB*'s accuracy of 86.3% is different from *AdaBoost*'s accuracy of 83.6%, this 2.7% difference is too small to be deemed significant. This is quite different from the story painted in Figure 4 in which *SMO* and *Adaboost* are exaggeratedly far from one another. On this graph, they are, in fact, slightly further from one another than they are from the other classifiers (that each obtained over 10% accuracy points more than they did).

In order to interpret the results, it is important to remember that the `Anneal` problem is severely imbalanced. The effects of this imbalance are clearly seen in the confusion matrices of *Adaboost* and *NB* in Tables 5 and 6.

As shown in Table 5, *Adaboost* only gets the points from the largest class and the third largest class well-classified, ignoring all the other classes. From Table 6, we see that *NB* classifies all the classes accurately, except for the largest class. We do not have enough space, here, to include the confusion matrices of the other methods, but we can report that they all did quite a good job on all classes. In effect this means that all the classifiers but *NB* and *Adaboost* are able to deal with the class imbalance problem, and that *NB* and *Adaboost* both behave badly on this domain, although they do so in different ways. This is exactly what the graph of Figure 4 tells us. The accuracy results do suggest that *NB* and *Adaboost* have problems, but they do not necessarily warn us of the severity of these problems nor do they differentiate between the two kind of problems.

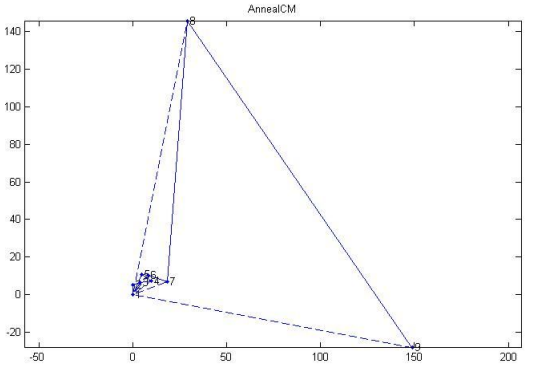


Fig. 4. Projection of the results on a MultiClass domain: Anneal

Table 4. Legend for Figure 4

Anneal Projection Legend			
Classifier number	Classifier name	Distance from origin	Distance from previous classifier
1	Ideal	0	
2	RandFor	5	
3	Ibk	7	4
4	J48	12	6
5	JRip	12	6
6	Bagging	13	3
7	SMO	20	11
8	NB	148	139
9	Adaboost	151	211

Table 5. The confusion Matrix for AdaBoost

Predicted/ True class	a	b	c	d	e	f
a	0	0	8	0	0	0
b	0	0	99	0	0	0
c	0	0	684	0	0	0
d	0	0	0	0	0	0
e	0	0	0	0	67	0
f	0	0	40	0	0	0

5 Multi-facetted Classifier Evaluation

The purpose of this section is to explore the kind of advantages our framework's flexibility can provide. We begin by pointing out that the visualizations we displayed in our previous graphs are only relative assessments. For example, in the

Table 6. The confusion Matrix for NB

Predicted/ True class	a	b	c	d	e	f
a	7	0	1	0	0	0
b	0	99	0	0	0	0
c	3	38	564	0	0	79
d	0	0	0	0	0	0
e	0	0	0	0	67	0
f	0	0	2	0	0	38

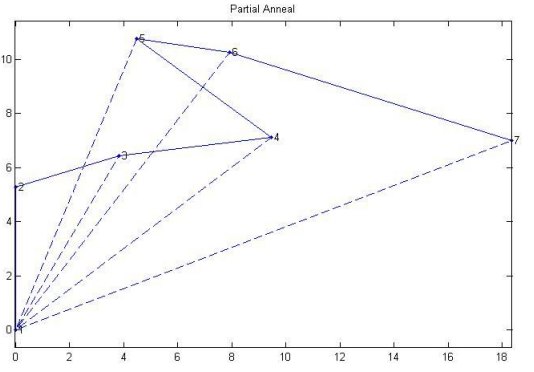


Fig. 5. Projection of the partial results on a MultiClass domain: Anneal

graph of Figure 4, we can see that all the classifiers, aside from *NB* (8) and *AdaBoost* (9) are very close together. After viewing the entire graph, we may want to zoom in on the tight cluster formed of classifiers 2 to 7, included. This is done in Figure 5 (whose legend is the same as that of Figure 4, i.e., the legend can be found in Table 4).

From this figure, we can see that *SMO* (7) does not perform as well as the other classifiers (though a lot better than *NB* and *AdaBoost* in Figure 4), that *RandFor* (2) and *IBk* (3) are the best classifiers on this problem, followed by *J48* (4), *JRip* (5) , which are somewhat equivalent in performance (though somewhat different from one another) and, finally, *Bagging* (6). An implementation that would allow the user to zoom in and out of graphs in that fashion would, thus, be quite a useful analytical tool.

Another issue we wish to investigate is the use of different distance measures. All our experiments, thus far used the Euclidean distance (L2 Norm), we wondered what the outcome would be if we were to use the Manhattan distance (L1 Norm), as well. The results are shown in Figure 6 which comes accompanied by table 7.

There is only one qualitative difference between the graphs produced by the L1 and the L2 norms: *NB* (8) appears closer to ideal than *Adaboost* (9) in Figure 6, than it did in Figure 4. Since the L2 norm penalizes the presence of major

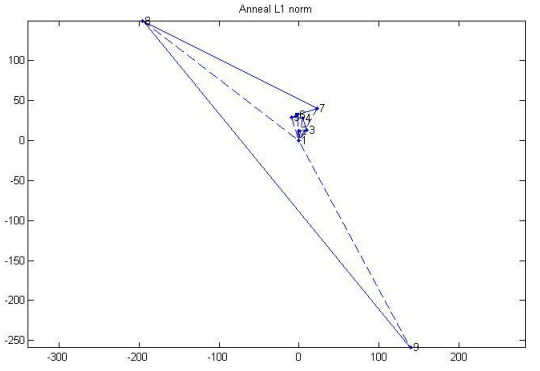


Fig. 6. Projection of the results on a MultiClass domain, using the L1 Norm: Anneal

Table 7. Legend for Figure 6

Anneal L1 Norm Projection Legend			
Classifier number	Classifier name	Distance from origin	Distance from previous classifier
1	Ideal	0	
2	RandFor	12	
3	IBk	16	10
4	J48	28	16
5	JRip	30	14
6	Bagging	32	8
7	SMO	46	26
8	NB	246	244
9	AdaBoost	294	528

concentrated misclassification errors more than the presence of small ones (since each concentration of error gets squared), and the L1 norm simply counts the number of misclassification errors present, we can reason that *NB* makes fewer errors than *Adaboost*, altogether, but that the majority of its errors are concentrated in one or a few large spots. In contrast, we can reason that although *Adaboost* makes more errors than *NB* altogether, its errors are more broadly distributed and appear in large numbers of small clusters. Another look at the confusion matrices of Tables 5 and 6 confirm this hypothesis. Indeed, we see that *Adaboost* makes 147 mistakes versus 123 for *NB*, thus explaining *NB*'s better performance with the L1 norm. In addition, since we see that, inconsiderate of class E, on which the two classifiers behave the same way, *NB* makes its major mistakes on class C, the largest class, whereas *Adaboost* makes no mistake on class C, but, instead, misclassifies all the other, smaller classes (except for class E), we understand where the results obtained with the L2 norm, which equate *Adaboost* and *NB*'s performance, come from. Thus, we can see how, provided

that we understand the meaning of the various distance measures we may use, each of them used simultaneously can quickly give us some important insight into the comparative performance of our classifiers.

6 Conclusion and Future Work

This paper presented a new evaluation method which, rather than aggregating the entries of the confusion matrices pertaining to the performance of a classifier into a single measure, treats all the performance data pertaining to that classifier as a high-dimensional vector. The vectors representing classifiers are then projected into a 2-dimensional space by a distance-preserving projection method. This approach presents several advantages, including the fact that it offers a visualization method that allows data mining practitioners to spot immediately any irregularity in the behaviour of their classifiers. It also indicates whether the detected irregularities are similar to each other or not. This particular method is but one implementation of the general framework we advocate that views the problem of classifier evaluation as one of analyzing high-dimensional data.

As presented, our approach may appear limited to the comparison of single classifier's performance, thus precluding the evaluation of threshold-insensitive classifiers and the computation of statistical guarantees in our results. This is not the case, however, since we could compute the results obtained by the same classifier using different thresholds and project a single point for each classifier at each threshold level. Similarly, in order to establish statistical guarantees, we could plot the results obtained at each fold of cross-validation for each classifier, giving us, a cloud of points for each classifier that would, this time, offer a visualization of the variance of that classifier. More formally, we could then apply a statistical test to the results of this projection.

We are also planning to expand our understanding of our framework by experimenting more thoroughly with different performance data representations (e.g., the outcome of classification on each test data point), different projection methods, as well as different distance measures. We believe that once it is carefully studied, this framework could become an integral part of the classifier evaluation process.

Finally, we are planning more experimentations on larger data sets, as well as on larger numbers of domains and classifiers, to test whether our method can scale up.

Acknowledgement. The first author would like to acknowledge the support of the Natural Science and Engineering Research Council of Canada for this research. Most of the work was completed at the Clayton School of Information Technology of Monash University during the first author's sabbatical leave.

References

1. Fawcett, T.: ROC graphs: Notes and practical considerations for data mining re searchers. Technical Report HPL 2003-4, HP Laboratories, Palo Alto, CA (January 2003)
2. Drummond, C., Holte, R.: Cost curves: An improved method for visualizing classifier performance. Machine Learning 65(1), 95–130 (2006)
3. Jain, A.K., Duin, R.P., Mao, J.: Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1) (2000)
4. Lee, R., Slagle, J., Blum, H.: A Triangulation Method for the Sequential Mapping of Points from N-Space to Two-Space. IEEE Transactions on Computers 26(3), 288–292 (1977)
5. Yang, L.: Distance-preserving projection of high dimensional data. Pattern Recognition Letters 25(2), 259–266 (2004)
6. Blake, C., Merz, C.: Uci repository of machine learning databases (1998)
7. Witten, I.H., Frank, E.: WEKA Software, v3.5.2. University of Waikato

Distortion-Free Nonlinear Dimensionality Reduction

Yangqing Jia, Zheng Wang, and Changshui Zhang

State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Automation, Tsinghua University, Beijing, China
{jiayq06,w-z04}@mails.tsinghua.edu.cn, zcs@mail.tsinghua.edu.cn

Abstract. Nonlinear Dimensionality Reduction is an important issue in many machine learning areas where essentially low-dimensional data is nonlinearly embedded in some high-dimensional space. In this paper, we show that the existing Laplacian Eigenmaps method suffers from the distortion problem, and propose a new distortion-free dimensionality reduction method by adopting a local linear model to encode the local information. We introduce a new loss function that can be seen as a different way to construct the Laplacian matrix, and a new way to impose scaling constraints under the local linear model. Better low-dimensional embeddings are obtained via constrained concave convex procedure. Empirical studies and real-world applications have shown the effectiveness of our method.

1 Introduction

Dimensionality reduction is an important issue in many pattern recognition and machine learning areas, where essentially low dimensional data is often embedded in some high dimensional space. Early works on dimensionality reduction are mostly linear methods such as PCA [1] and MDS [2], and have been widely applied and discussed. However, in recent years, researchers have realized that in many situations the data lies on a low-dimensional manifold embedded in the feature space, and the embedding is often difficult to be captured by the simple linear model. Thus, nonlinear dimensionality reduction (NLDR) is believed to be more powerful to preserve the low dimensional information under such situations.

Several NLDR methods have been proposed in the recent years, such as ISOMAP [3], Locally Linear Embedding (LLE) [4], local tangent space alignment (LTSA) [5], maximum variance unfolding (MVU) [6], and so on. In this paper, we will mainly focus on methods that rely on Laplacian matrices, the most representative work of which is the Laplacian Eigenmaps [7]. It constructs a Laplacian matrix by constructing a graph over the data set, and calculates the low-dimensional coordinates via generalized eigenvalue decomposition. The method has been simplified to a linear version called Locality Preserving Projections [8], which benefits from both the nonlinear graph structure and the simplicity of the linearity.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 564–579, 2008.

One of the key points for Laplacian Eigenmaps is how to construct the Laplacian matrix. Most commonly, it is constructed based on K-nearest neighbor and a Gaussian kernel to calculate the pairwise similarity between data points. Although its theoretical foundations and relationship with the Laplacian-Beltrami operator have been well studied (see *e.g.* [9]), it may not be universally the best option. Specifically, we will discuss in this paper that it has a *distortion problem*: it tends to drive the data points far from the center of the low-dimensional coordinate and may also expand the small "holes" in the intrinsic distribution of the data.

In this paper, we propose a different way to construct the Laplacian matrix by explicitly defining a linear model in each local area of the data. Next, we use the local gradients obtained from the linear model to impose additional scaling constraints to solve the distortion problem. We adopt the *constrained concave convex procedure* (CCCP) to solve the optimization problem, and show the performance of the new dimensionality reduction both visually and quantitatively.

The remainder of this paper is organized as follows: Section 2 introduces the basic notations and the distortion problem of Laplacian Eigenmaps. Section 3 discusses the loss function of our method, leading to the construction of a new Laplacian matrix, and the implementation of the scaling criterions. Section 4 shows experimental results comparing our method and the closely related methods with discussions. Finally, Section 5 concludes the paper.

2 Laplacian Eigenmaps and the Distortion Problem

In this section, after defining necessary notations, we will discuss the distortion problem of the Laplacian Eigenmaps.

2.1 Notations

Formally, in nonlinear dimensionality reduction problems, we are given a set of D-dimensional data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subset \mathbb{R}^D$. The task of NLDR is to find a corresponding set $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\} \subset \mathbb{R}^d$ ($d \ll D$), where $\mathbf{y}_i$ is the low dimensional representation of $\mathbf{x}_i$. The coordinates of each low dimension can also be seen as being calculated from a mapping function $f^i : \mathbb{R}^D \to \mathbb{R}$, where $i = 1, \cdots, d$. Thus, we define the dimension-wise set $\mathcal{F} = \{\mathbf{f}^1, \mathbf{f}^2, \cdots, \mathbf{f}^d\} \subset \mathbb{R}^n$, where the j-th element of $\mathbf{f}^i$ is the i-th dimensional value of $\mathbf{y}_j$.

The Laplacian Eigenmaps [7] method finds the low-dimensional representation by minimizing the weighted sum of the squared distances between neighboring data points:

$$\mathcal{J}(\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_d) = \sum_{k=1}^{d} (\mathbf{f}^k)^\top \mathbf{L} \mathbf{f}^k = \sum_{k=1}^{d} \sum_{i,j=1}^{n} (f_i^k - f_j^k)^2 w_{ij} \ . \tag{1}$$

$\mathbf{L}$ is the graph Laplacian matrix with its (i, j)-th element

$$L_{ij} = \begin{cases} d_i - w_{ii}, & if\ i = j \\ -w_{ij}, & otherwise \ , \end{cases} \tag{2}$$

where $d_i = \sum_j w_{ij}$, and w_{ij} is the similarity between data point $\mathbf{x}_i$ and $\mathbf{x}_j$. There are several different ways to calculate the similarity, among which the weighted exponential similarity is most often used: let d_{ij} be the distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, the weight is calculated as

$$w_{ij} = \exp\left[-d_{ij}^2/(2\sigma^2)\right] \quad , \tag{3}$$

where σ is a parameter that controls the width of the Gaussian function. Usually, a precedent K-nearest neighborhood is calculated to force $w_{ij} = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are not neighbors.

To recover the low-dimensional coordinates, Belkin *et al.* [7] proposed to solve the following optimization problem:

$$\begin{aligned} \min \ & \sum_{k=1}^{d} (\mathbf{f}^k)^\top \mathbf{L}\mathbf{f}^k \\ s.t. \ & (\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^j = \delta_{ij}, \ (\mathbf{f}^i)^\top \mathbf{D}\mathbf{1} = 0, \quad 1 \le i, j \le d \ , \end{aligned} \tag{4}$$

where δ_{ij} is the *kronecker delta* function that takes value 1 if $i = j$ and 0 otherwise. $\mathbf{D}$ is the diagonal degree matrix with the i-th diagonal element be d_i. This is further solved by the following generalized eigenvalue decomposition problem:

$$\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f} \ , \tag{5}$$

and the d eigenvectors corresponding to the smallest non-zero eigenvalue are the low-dimensional coordinates (The eigenvector corresponding to eigenvalue 0 is constant vector $\mathbf{1}$ and is discarded).

We note here that solving the eigenvalue decomposition problem $\mathbf{L}\mathbf{f} = \lambda\mathbf{f}$ is also applicable. From a spectral clustering view [10], the former is equivalent to minimizing the Normalized Cut [11] criterion on the corresponding graph defined by the Laplacian matrix, and the latter is equivalent to minimizing the Ratio Cut [12] criterion.

2.2 The Distortion Problem

To show the distortion problem, we construct a toy *clover* data shown in Fig. 1(a) and consider using Laplacian Eigenmaps to recover the 2-dimensional coordinates[1]. From the recovered coordinates under different σ values shown in Fig. 1(b), we can see that Laplacian Eigenmaps may not recover the low-dimensional data satisfactorily: the circlar structure of the data is expanded and the data area with higher density (the blue dots) is driven far from the center of the low-dimensional embedding. This is essentially different from the ground truth. Another example is the recovered embedding of the Swiss roll, see Fig. 3. Laplacian Eigenmaps tends to expand the "holes" in the intrinsic data distribution, and squeeze the data points into lines of high-density areas, which produces distorted results.

[1] For details about constructing the toy data, see Section 4.

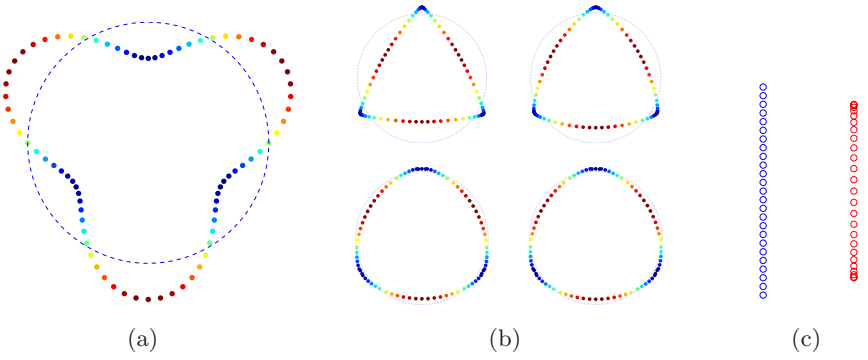


Fig. 1. The distortion problem of Laplacian Eigenmaps. (a) The original toy clover data; (b) recovered coordinates using Laplacian Eigenmaps with $\sigma = 0.05, 0.1, 1$, and $+\infty$ respectively; (c) example of a one-dimensional uniformly spaced data (blue points on the left) showing that Laplacian Eigenmaps tend to drive the recovered coordinates (red points on the right) to the two ends, very similar to performing clustering.

The distortion problem roots largely in two aspects. First, the loss function of Laplacian Eigenmaps considers minimizing the pairwise distance between neighboring data points only and does not consider any isometric information, thus it may essentially neglect the intrinsic shape of the data. Second, Laplacian Eigenmaps can be seen as minimizing $\mathbf{f}^\top \mathbf{L}\,\mathbf{f}$ and maximizing $\mathbf{f}^\top \mathbf{D}\mathbf{f}$ at the same time. Although maximizing $\mathbf{f}^\top \mathbf{D}\mathbf{f}$ successfully removes the scaling factor and avoids trivial solution $\mathbf{f} = \mathbf{0}$, it also tends to give large function values for data points in the dense area because they have large degree values d_i. Even for uniformly distributed data, Laplacian Eigenmaps still tends to drive the data to the two ends of the low-dimensional embedding, leaving a very low density area in the middle, see Fig. 1(c) for example.

In other words, Laplacian Eigenmaps works more similar to clustering than dimensionality reduction if we see the low-dimensional coordinates as class labels: it tries to recover the labels as determinate as possible, and avoids the central ambiguous area. Admittedly, this is in some cases desired and natural, such as in spectral clustering (actually, Normalized Cuts [11] is equivalent to Laplacian Eigenmaps plus discretization), it may not be an optimal choice for dimensionality reduction since it produces unnecessary distortion that especially affects data visualization.

In the next section, we aim to solve the distortion problem by both changing the loss function and the constraints.

3 The Proposed Method

In this section, we will first propose a new approach to construct the Laplacian matrix for dimensionality reduction, and then impose additional constraints to solve the distortion problem.

Table 1. A collection of the notations involving f, for better clarification

$f : \mathbb{R}^D \to \mathbb{R}$, and f^k	the mapping function, the superscript is used if we specify the function for the k-th dimension.
f_i, and f_i^k	the (k-th dimensional) function value for the i-th data $\mathbf{x}_i$.
$\mathbf{f}$, and $\mathbf{f}^k$	the (k-th dimensional) vector for all the data points.
$\mathbf{f}_i$, and $\mathbf{f}_i^k$	a $(K+1) \times 1$ local column vector whose first K elements are $[f_j]$ for all $\mathbf{x}_j \in \mathcal{N}_i$ and the last element is f_i. See Section 3.2.
$\tilde{\mathbf{f}}$	an $n(K+1) \times 1$ concatenated label vector, $\tilde{\mathbf{f}} = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \cdots, \mathbf{f}_n^\top]^\top$.

3.1 The Local Linear Model and the Loss

To simplify the representation of symbols, we consider reducing the dimensionality to $d = 1$: let $\mathbf{f} = [f_1, f_2, \cdots, f_n] \in \mathbb{R}^n$ be such a mapping that each f_i is the low-dimensional coordinate of $\mathbf{x}_i$. This can also be seen from a functional-view as $f_i = f(\mathbf{x}_i)$ where f is the mapping functional. In most real-world applications, we can assume f to be differentiable, *i.e.*, if there are enough data points, then in the small neighborhood $\mathcal{N}_i$ of $\mathbf{x}_i$ (we use K-nearest neighbor to define the neighborhood), we may use the first-order Taylor expansion to approximate the function as:

$$f_j \approx f_i + \nabla f(\mathbf{x}_i)^\top (\mathbf{x}_j - \mathbf{x}_i), \quad \forall \mathbf{x}_j \in \mathcal{N}_i \ . \tag{6}$$

In other words, if we find a mapping $\mathbf{f}$ that represents the local coordinates well, we expect the function values f_j in the neighborhood of $\mathbf{x}_i$ to behave linearly with respect to $\mathbf{x}_j - \mathbf{x}_i$ and f_i. Thus, a direct criterion to evaluate the fitness of a mapping $\mathbf{f}$ is to minimize the sum of the least-square loss in each neighborhood of the data points:

$$\mathcal{J}(\mathbf{f}) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} [f_j - f_i - \nabla f(\mathbf{x}_i)^\top (\mathbf{x}_j - \mathbf{x}_i)]^2 \ . \tag{7}$$

The thought of the loss function is inspired by the two most related works, namely Laplacian Eigenmaps [7] and Locally Linear Embedding (LLE) [4]. We discuss the relationship and difference here.

LLE. One may think that our derivation is similar to LLE at first glance, which assumes that each data point can be linearly reconstructed from its neighboring data points. In other words, for LLE, a reconstruction weight matrix W is calculated from the data so that $\sum_j W_{ij}\mathbf{x}_j$ gives the least-square estimation of $\mathbf{x}_i$ under the constraint $\sum_j W_{ij} = 1$. However, LLE focuses on the linear relationship from the neighborhood points and considers little about the direct relationship between the high and low coordinates, while in our model we explicitly adopt a function that generates the embedding from the high-dimensional coordinates. Thus the two methods are conceptually different. Further, when

the neighbor number K is larger than the dimensionality d, such reconstruction may not be unique. Even when unique solution exists, eliminating the suboptimal reconstruction weights may still lose information, as has been pointed out in [13]. Thus LLE may only be able to preserve comparatively weak information. Instead, by defining a local linear model $f_j = \mathbf{w}_i^T(\mathbf{x}_j - \mathbf{x}_i) + f_i$, we may preserve richer information than LLE: assume that the least-square estimations are precise, then any weight $\{W_{ij}\}$ that satisfies $\mathbf{x}_i = \sum_j W_{ij}\mathbf{x}_j$ will automatically satisfy $f_i = \sum_j W_{ij} f_j$ under the local linear model. Thus, we also expect our method to perform better than LLE in utilizing the local information.

Laplacian Eigenmaps. The Laplacian Eigenmaps method uses a slightly different criterion, which attempts to minimize the distance between nearby points with weight factor W_{ij}, *i.e.*, to minimize $\sum_j W_{ij}(f_i - f_j)^2$ in each local area. However, this introduces an additional parameter σ in the heat kernel that needs careful tuning, and may dilate the small holes in the intrinsic structure of the data as we have indicated in Section 2, which can further be seen experimentally in Fig. 3. We will experimentally show that our loss criterion empirically works better in Section 4. Theoretically, our loss criterion and the one of Laplacian Eigenmaps are essentially different: since the low-dimensional coordinates of nearby data points are essentially different (otherwise they will collapse to a constant value), we argue that it may not be optimal to impose penalty on any changes of the low-dimensional coordinates in the local area. Instead, we adopt a local linear model to allow reasonable changes, and only impose penalty if the low-dimensional embedding does not follow the model. This may be more reasonable than simply minimizing the distance of the neighbors.

3.2 Constructing the Laplacian Matrix

The above idea faces some difficulty to implement: there does not exist an explicit analytical representation of the gradient ∇f, since usually we only find the mapping value on the n data points. Thus, we denote the gradient vector at each data point $\mathbf{x}_i$ by an additional hidden parameter $\mathbf{w}_i \in \mathbb{R}^D$, and equivalently rewrite the loss function as:

$$\mathcal{J}(\mathbf{f}, \mathbf{w}_1, \cdots, \mathbf{w}_n) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} [f_j - f_i - \mathbf{w}_i^\top (\mathbf{x}_j - \mathbf{x}_i)]^2 \ . \tag{8}$$

Then, the remaining problem is how to estimate each $\mathbf{w}_i$. Since $\mathbf{w}_i$ is an ancillary parameter and we want to find $\mathbf{f}$ ultimately, we consider using $\mathbf{f}$ to analytically represent the gradient $\mathbf{w}_i$. That is, if given the function value $\mathbf{f}$, how to estimate the gradients $\{\mathbf{w}_i\}$ at each data point $\mathbf{x}_i$?

A straightforward thought may be to adopt the least-square estimation of the linear model at each neighborhood as $\mathbf{w}_i$. However, in most real-world applications, we often face a high dimensionality D (which may be significantly larger than the neighbor number K) and a comparatively small number of data points in the local area. This may cause the simple least-square estimation to be

unstable especially when there is noise in the data. Thus, we add an additional regularizer $\|\mathbf{w}_i\|^2$ in a SVM-like way to perform structural risk minimization. Mathematically, we aim to estimate $\mathbf{w}_i$ to be the optimum solution to the following regularized least-square problem:

$$\arg\min_{\mathbf{w}_i} \quad [\lambda\|\mathbf{w}_i\|^2 + \sum_{j\in\mathcal{N}_i}(f_j - f_i - \mathbf{w}_i^\top(\mathbf{x}_j - \mathbf{x}_i))^2] \ , \tag{9}$$

where λ is a constant weight parameter. The thought is inspired by the supervised local learning algorithms [14] and semi-supervised Local Learning Regularization [15]. However, instead of interpreting the local model as a "classifier" and using the classification error as the loss function, we use the local model to do linear regression, which is more reasonable for dimensionality reduction, where the low-dimensional coordinates take continuous values.

Such estimation can further be written in a closed form. Define matrix X_i to be the matrix $[\mathbf{x}_j - \mathbf{x}_i]$ for all $\mathbf{x}_j \in \mathcal{N}_i$, local vector $\mathbf{f}_i$ to be a $(K+1)\times 1$ column vector whose first K elements are $[f_j]$ for all $\mathbf{x}_j \in \mathcal{N}_i$ and the last element is f_i, and a $K\times(K+1)$ ancillary matrix $H = [I_{K\times K}, -\mathbf{1}_{K\times 1}]$. Then, the optimization problem Eqn. (9) can be written as:

$$\arg\min_{\mathbf{w}_i} \quad \lambda\|\mathbf{w}_i\|^2 + \|H\mathbf{f}_i - X_i^\top\mathbf{w}_i\|^2 \ . \tag{10}$$

Take the derivative with respect to $\mathbf{w}_i$ and set it to be zero, we get

$$\mathbf{w}_i = A_i\mathbf{f}_i, \quad \text{where } A_i = (\lambda I + X_i X_i^\top)^{-1} X_i H \ . \tag{11}$$

Substitute (11) into (8), the loss function $\mathcal{J}$ can then be written in a closed form of $\mathbf{f}$ as

$$\mathcal{J}(\mathbf{f}) = \sum_{i=1}^{n} \mathbf{f}_i^\top M_i \mathbf{f}_i \ , \tag{12}$$

where the matrix M_i is

$$M_i = H^\top(I - X_i^\top A_i)^\top(I - X_i^\top A_i)H \ . \tag{13}$$

Next, we define an $n(K+1)\times 1$ concatenated label vector $\tilde{\mathbf{f}} = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \cdots, \mathbf{f}_n^\top]^\top$, an $n(K+1)\times n(K+1)$ concatenated block-diagonal matrix

$$M = \begin{bmatrix} M_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & M_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & M_n \end{bmatrix} \ , \tag{14}$$

and an $n(K+1)\times n$ selection matrix S whose elements take 0-1 values and there is only one 1 in each row so that $\tilde{\mathbf{f}} = S\mathbf{f}$. Then, the loss function is written as

$$\mathcal{J}(\mathbf{f}) = \mathbf{f}^\top\mathbf{L}\,\mathbf{f}, \quad \text{where } \mathbf{L} = S^\top M S \ . \tag{15}$$

It is not difficult to prove the following property:

Theorem 1. *The matrix* $\mathbf{L}$ *is a Laplacian matrix, i.e., it is positive semi-definite and satisfies* $\mathbf{L1} = \mathbf{0}$ *where* $\mathbf{1}$ *and* $\mathbf{0}$ *are* $n \times 1$ *constant-value column vectors.*

Proof. See the Appendix.

3.3 Trivial Solution Elimination

Similar to Laplacian Eigenmaps, simply minimizing the loss function $\mathbf{f}^\top \mathbf{L}\mathbf{f}$ leads to a trivial solution $\mathbf{f} = 0$. Thus, we add scaling constraints to eliminate such trivial solutions. To eliminate the scaling factor, we make an isometric assumption that the norm of the local gradient vector at each data point $\mathbf{x}_i$ is 1, *i.e.*, $\|\mathbf{w}_i\|^2 = 1$, inspired by the idea of PCA. This leads to n quadratic constraints $\mathbf{f}_i^\top A_i^\top A_i \mathbf{f}_i = 1$, where $i = 1, \cdots, n$. As a relaxation and also for faster optimization, we may only place constraint on the sum of all the local gradients' norms as $\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{w}_i\|^2 = 1$, which further leads to $\mathbf{f}^\top \mathbf{D}\mathbf{f} = 1$, where the scalar matrix $\mathbf{D}$ is calculated as

$$\mathbf{D} = S^\top \begin{bmatrix} A_1^\top A_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A_2^\top A_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_n^\top A_n \end{bmatrix} S \ . \tag{16}$$

When the reduced dimensionality is larger than 1, we constraint each two low-dimensional mappings $\mathbf{f}^{k_1}$ and $\mathbf{f}^{k_2}$ to be such that their local gradient vectors at each point $\mathbf{x}_i$ to be orthogonal to each other, *i.e.*, $\mathbf{w}_i^{k_1} \perp \mathbf{w}_i^{k_2}$, inspired by the idea of PCA. This leads to the constraints $(\mathbf{f}_i^{k_1})^\top A_i^\top A_i \mathbf{f}_i^{k_2} = 0$, where $i = 1, \cdots, n$. Similar to the discussion above, we consider relaxing these constraints to the sum over the n points as $\sum_{i=1}^{n} (\mathbf{f}_i^{k_1})^\top A_i^\top A_i \mathbf{f}_i^{k_2} = 0$. It is easy to see that this leads to the constraint

$$(\mathbf{f}^{k_1})^\top \mathbf{D}\mathbf{f}^{k_2} = 0, \quad \text{for } k_1 \neq k_2 \ . \tag{17}$$

Further, to remove the degree of translation freedom, we require the coordinates to be centered in the origin, *i.e.*, to require $(\mathbf{f}^k)^\top \mathbf{1} = 0$.

Note that although looking similar, our constraint is different from Laplacian Eigenmaps, as the definitions of the matrix $\mathbf{D}$ in the two methods are different. It is worth pointing out the following theorem:

Theorem 2. *The matrix* $\mathbf{D}$ *is a Laplacian matrix.*

Proof. See the Appendix.

This leads to a different optimization approach, which we will observe in the next subsection.

3.4 Optimization

We use the following way to recover the d low-dimensional coordinates sequentially: for the i-th dimension ($1 \leq i \leq d$), we summarize the optimization task as follows:

$$\begin{aligned} \min_{\mathbf{f}^i} \ & (\mathbf{f}^i)^\top \mathbf{L}\mathbf{f}^i \\ s.t. \ & (\mathbf{f}^i)^\top \mathbf{1} = 0, \ (\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i = 1 \\ & (\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^j = 0, \quad \forall 1 \leq j < i \ . \end{aligned} \tag{18}$$

This is a non-convex problem because of the constraint $(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i = 1$. Note that generalized eigenvalue decomposition $\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}$ is not applicable to solve this problem. We explain this in detail. In Laplacian Eigenmaps, the constraint $(\mathbf{f}^i)^\top \mathbf{1} = 0$ will be automatically satisfied by eliminating the eigenvector $\mathbf{1}$ corresponding to generalized eigenvalue 0 (note that the degree matrix in Laplacian Eigenmaps is positive definite). However, in our method, the matrix $\mathbf{D}$ is defined differently and is also a Laplacian matrix. This leads to $\mathbf{L}\mathbf{1} \equiv \mathbf{D}\mathbf{1} \equiv 0$, which means that $\mathbf{1}$ is a generalized eigenvector corresponding to any value in $\mathbb{R}$. Most GEVD solvers may suffer from severe numerical problems performing such tasks.

Instead, to get a stable and satisfying solution, we directly solve the problem using Constrained Concave Convex Procedure (CCCP) [16]. CCCP works in an iterative way: at each iteration, the 1-st order Taylor expansion is used to approximate the non-convex constraints, and the problem is thus approximated by a convex optimization subproblem. The optimum solution to the subproblem is then used as the initial value of the next iteration. This procedure is repeated until convergence, which has been theoretically guaranteed in [16]. Specifically, we rewrite the constraint $(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i = 1$ to a convex one $(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i \leq 1$ and a concave one $(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i \geq 1$. At each iteration, assume that the initial value is $\mathbf{f}^{i,\text{old}}$, the concave constraint is approximated as

$$2(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^{i,\text{old}} - (\mathbf{f}^{i,\text{old}})^\top \mathbf{D}\mathbf{f}^{i,\text{old}} \geq 1 \ . \tag{19}$$

Thus the subproblem given initial value $\mathbf{f}^{i,\text{old}}$ is

$$\begin{aligned} \min_{\mathbf{f}^i} \ & (\mathbf{f}^i)^\top \mathbf{L}\mathbf{f}^i \\ s.t. \ & (\mathbf{f}^i)^\top \mathbf{1} = 0, \ (\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^i \leq 1 \\ & 2(\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^{i,\text{old}} - (\mathbf{f}^{i,\text{old}})^\top \mathbf{D}\mathbf{f}^{i,\text{old}} \geq 1 \\ & (\mathbf{f}^i)^\top \mathbf{D}\mathbf{f}^j = 0, \quad \forall 1 \leq j < i \ . \end{aligned} \tag{20}$$

It is a standard *quadratic programming* (QP) problem that can be solved efficiently.

Further, it is worth indicating that although CCCP only guarantees local optimum, in our experiments we have reached identical solutions from multiple runs with different initial values, implying that the optimization procedure may already be stable and good enough for real-world applications. Still, a theoretical analysis on the optimization method is one of our further interests. One

shortcoming using CCCP is that it may require some time to converge. In our experiments, the method runs about half a minute for a 2000-point data set using Matlab code and Mosek to carry out the optimization on a P4 2G computer, which is a time a bit longer than that spent by LE.

3.5 Extension for Out-of-sample Data

To predict the low-dimensional coordinates of a new data point $\mathbf{x}$ that does not belong to the original data set $\mathcal{X}$, we propose the following induction approach: first find its neighborhood $\mathcal{N}_{\mathbf{x}}$ in the original data set, and then find the low-dimensional embedding $f(\mathbf{x})$ that simultaneously fits the local linear models on all its neighbor points well by minimizing the following criterion:

$$f(\mathbf{x}) = \arg\min_{f_{\mathbf{x}}} \sum_{\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}} [(f_{\mathbf{x}} - f_i) - \mathbf{w}_i^\top (\mathbf{x} - \mathbf{x}_i)]^2 \ . \tag{21}$$

Take the derivative with respect to $f_{\mathbf{x}}$ and set it to be zero, the result can be written in a closed form as

$$f(\mathbf{x}) = \frac{1}{K} \sum_{\mathbf{x}_i \in \mathcal{N}_{\mathbf{x}}} [f_i + \mathbf{w}_i^\top (\mathbf{x} - \mathbf{x}_i)] \ , \tag{22}$$

where K is the number of neighbor points.

4 Experiments

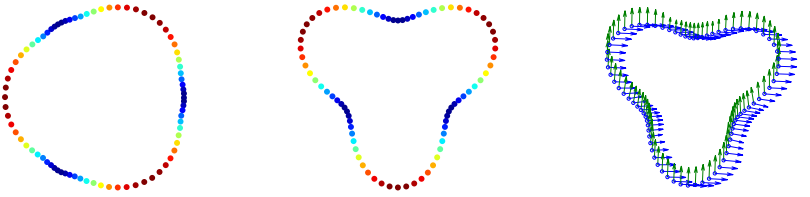
In this section, we perform dimensionality reduction for data visualization on toy and real-world data sets. For all the experiments, the parameter σ in Laplacian Eigenmaps is chosen via a line search from $\{1/16, 1/4, 1, 4, 16, \infty\} \times \sigma_0$ where σ_0 is the squared root of the mean pairwise distance between data points; the parameter λ in our method is chosen from $\{0.01, 0.1, 1, 10\}$.

4.1 Toy Data Sets

We generate a toy *clover* data set with two essential dimensions generated by $\{x = (1+0.3\sin(3t))\cos(t), y = (1+0.3\sin(3t))sin(t)\}$ where t takes 80 uniformly spaced values from $[0, 2\pi)$. The data is then mapped to a 20 dimensional space with a randomly generated linear transform. We test our method against two most related algorithms namely Laplacian Eigenmaps and LLE[2] to recover the true 2-dimensional embedding using neighbor number $K = 10$. The result of Laplacian Eigenmaps has been shown in Fig. 1(b). The result of LLE and our method are shown in Fig. 2. Although all three methods keep the neighboring information well, our method clearly preserves better shape information, while the other two methods show poorer reconstructions.

Next we experiment on four kinds of Swiss roll data sets that have appeared in the previous literature. *Swiss Roll* is the classical Swiss roll data set;

[2] The LLE code used here is from http://www.cs.toronto.edu/~roweis/lle/code.html.



(a) LLE (b) Our method (c) Gradient

Fig. 2. The result of LLE and our method on the toy clover data. Figure (c) shows the local gradient vectors estimated by our method.

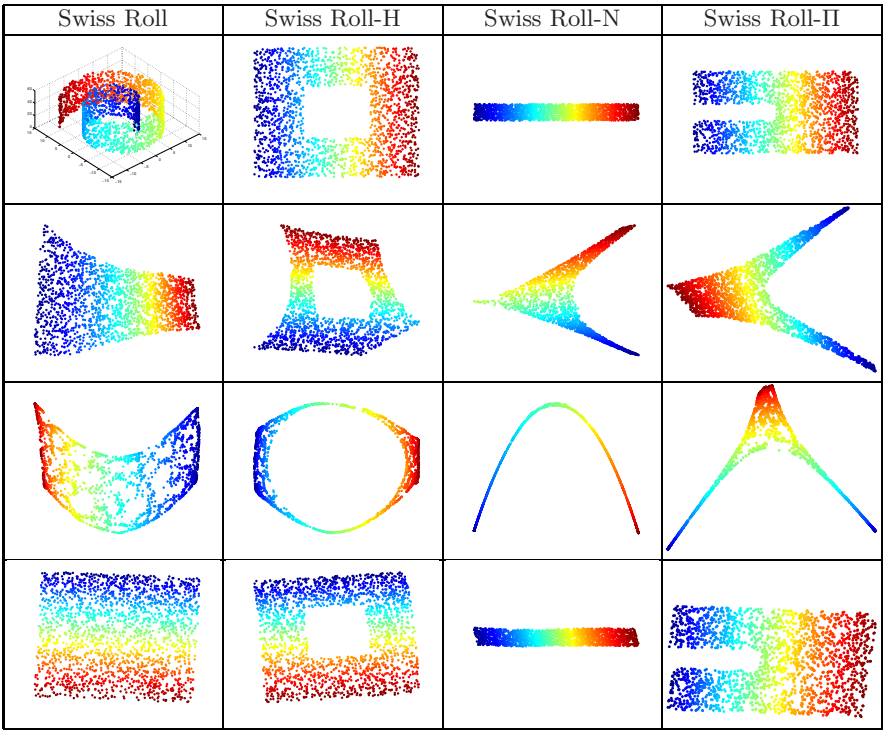


Fig. 3. Experimental results on the Swiss roll data sets. In the first row, the first figure shows a 3-dimensional roll structure and the other two shows the true 2-dimensional coordinates of the corresponding data. The 2nd to the 4th rows are the results of LLE, Laplacian Eigenmaps, and our method respectively.

Swiss Roll-H cuts a squared hole on the intrinsic data structure; *Swiss Roll-N* has a narrow width and a comparatively long roll; *Swiss Roll-Π* embeds a π-shaped set of data into the 3-dimensional space. The data and results are shown in Fig. 3. We can observe visually that our method better preserves the

low-dimensional structure under all conditions, while LLE may give globally distorted recovery and Laplacian Eigenmaps tend to "squeeze" the data into lines of high-density areas as we have analyzed in the previous part of the paper.

4.2 Quantitative Comparison

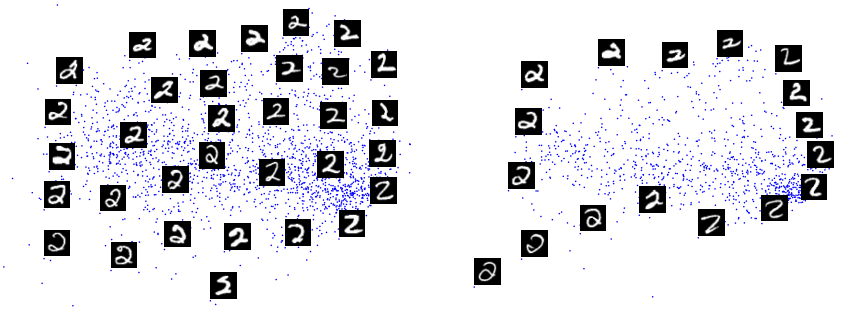
For the two data sets, since we have the true low-dimensional coordinates of the data, we are able to calculate the *procrustes measure* (PM) [17] between the true coordinates and the recovered ones for each method for quantitative comparison. Procrustes measure between two sets of points determines a linear transformation (including translation, reflection, orthogonal rotation, and scaling) and uses the sum of squared errors as a goodness-of-fit measure, which in our case shows the accuracy of the dimensionality reduction algorithms. A smaller PM value indicates more accurate recovery. We also calculate the local procrustes measure, which is the mean of the procrustes measures in each local neighborhood, to compare the accuracy of the dimensionality reduction algorithms in the local area. PCA is adopted as a baseline. Table 2 shows the results of different methods on the toy data sets (if one algorithm needs parameter tuning, the best result is reported here). The result verifies the superiority of our method to preserve the intrinsic data structure both locally and globally.

Table 2. The procrustes measure (PM) and local procrustes measure (LPM) on the toy data sets. LE denotes Laplacian Eigenmaps and DFDR is our distortion-free dimensionality reduction method.

Data set	PM				LPM			
	PCA	LLE	LE	DFDR	PCA	LLE	LE	DFDR
clover	0.1464	0.0303	0.0637	1.006×10^{-4}	0.0433	0.0167	0.0396	2.188×10^{-4}
Swiss Roll	0.6891	0.1973	0.2090	0.0032	0.1078	0.4525	0.2726	0.0051
Swiss Roll-H	0.6791	0.1232	0.1678	0.0047	0.1136	0.1022	0.3519	0.0077
Swiss Roll-N	0.8988	0.4738	0.5116	0.0016	0.4255	0.2917	0.4434	0.0187
Swiss Roll-II	0.6634	0.2066	0.2550	0.0054	0.1201	0.1316	0.3316	0.0068

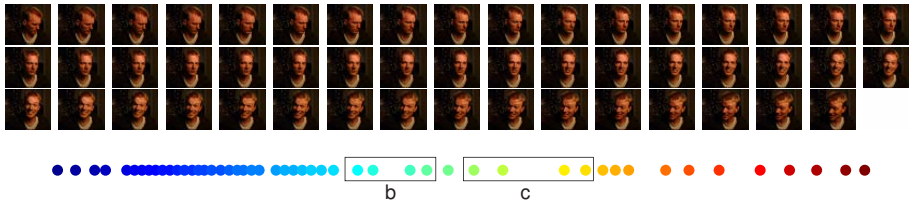
4.3 Real-World Data

In this part, we perform dimensionality reduction on real-world data sets. We randomly select 2000 digital TWOs from the MNIST database [18] for dimensionality reduction and 1000 for out-of-sample test, and show the results in Fig. 4. For computation speed consideration, a precedent PCA is adopted to save 95% of the energy before NLDR is applied. It can be seen that both results reveal the intrinsic structure of the low-dimensional embedding. Next, we adopt the sequential smiling face data from [19] to test our method. The one-dimensional embedding and the original images are shown in Fig. 5. Our method recovers the low-dimensional coordinates well and is consistent with the original data. This can be seen visually: in the recovered coordinates, a larger gap between

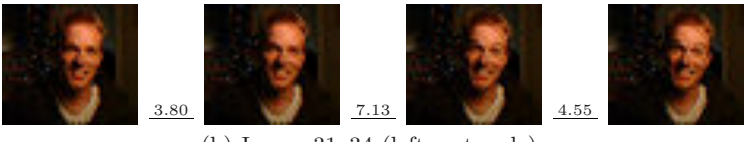


(a) Dimensionality reduction result (b) Out-of-sample result

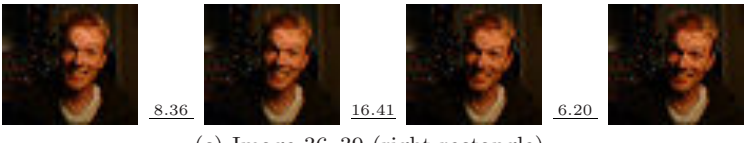
Fig. 4. The result of dimensionality reduction on the digital TWO images



(a) The sequential images and the recovered coordinates



(b) Image 31–34 (left rectangle)



(c) Image 36–39 (right rectangle)

Fig. 5. Recovered coordinates from a sequential smiling face images. The images are borrowed from [19]. The numbers in (b) and (c) shows the Euclidean distance between related images.

data points such as in the rectangle marked “b” and “c” corresponds to a larger distance between the corresponding images. It is comparatively difficult for us to provide a quantitative evaluation (and comparison against other NLDR algorithms) like we did on the toy data, as we do not know what low-dimensional coordinates are “optimal”. However, the experiments have revealed that out method does obtain good dimensionality reduction results visually and preserves the local information well.

5 Conclusion

Dimensionality reduction has been an important issue in the machine learning field to reveal the low-dimensional structure of the data. The main contribution of our paper lies in two aspects: (1) we discussed the distortion problem of the classical Laplacian Eigenmaps method and proposed an alternative way to construct the Laplacian matrix via a local linear model; (2) we proposed a new way to reduce the dimensionality to obtain a better result both visually and numerically, and avoid the distortion problem successfully. In the future, we will explore the theoretical nature of the proposed method as well as developing more efficient algorithms to solve the optimization task.

Acknowledgement

This work is supported by National 863 Project (No. 2006AA01Z121) and NSFC (Grant No. 60675009) of China.

References

1. Jolliffe, I.: Principal Component Analysis. Springer, Heidelberg (2002)
2. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall, Boca Raton (2001)
3. Tenenbaum, J., Silva, V., Langford, J.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500), 2319–2323 (2000)
4. Roweis, S., Saul, L.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290(5500), 2323–2326 (2000)
5. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM J. Sci. Comp. 26(1), 313–338 (2005)
6. Weinberger, K., Saul, L.: Unsupervised Learning of Image Manifolds by Semidefinite Programming. Intl. J. Computer Vision 70(1), 77–90 (2006)
7. Belkin, M., Niyogi, P.: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation 15(6), 1373–1396 (2003)
8. He, X., Niyogi, P.: Locality preserving projections. Advances in Neural Information Processing Systems (2003)
9. Belkin, M., Niyogi, P.: Towards a theoretical foundation for laplacian-based manifold methods. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 486–500. Springer, Heidelberg (2005)
10. Ding, C., Simon, H., Jin, R., Li, T.: A learning framework using Green's function and kernel regularization with application to recommender system. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 260–269 (2007)
11. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
12. Hagen, L., Kahng, A.: New spectral methods for ratio cut partitioning and clustering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11(9), 1074–1085 (1992)
13. Zhang, Z., Wang, J.: MLLE: Modified Locally Linear Embedding Using Multiple Weights. Advances in Neural Information Processing Systems (2007)

14. Bottou, L., Vapnik, V.: Local learning algorithms. Neural Computation 4(6), 888–900 (1992)
15. Wu, M., Schölkopf, B.: Transductive classification via local learning regularization. In: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 624–631 (2007)
16. Smola, A., Vishwanathan, S., Hofmann, T.: Kernel methods for missing variables. In: Proc. International Workshop on Artificial Intelligence and Statistics (2005)
17. Seber, G.: Multivariate Observations. Wiley, Chichester (1984)
18. LeCun, Y.: The MNIST database of handwritten digits. NEC Research Institute, `http://yann.lecun.com/exdb/mnist/index.html`
19. Gashler, M., Ventura, D., Martinez, T.: Iterative non-linear dimensionality reduction with manifold sculpting. In: Advances in Neural Information Processing Systems 20, pp. 513–520 (2008)

Appendix: Proof to the Theorems

Proof to Theorem 1: To prove that $\mathbf{L}$ is positive semi-definite, we first write $\mathbf{L}$ as $\mathbf{L} = P^\top P$, where we define the ancillary matrix P as

$$P = \begin{bmatrix} (I - X_1^\top A_1)H & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & (I - X_2^\top A_2)H & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & (I - X_n^\top A_n)H \end{bmatrix} S \ , \tag{23}$$

where the selection matrix S is defined in Section 3.2. For any arbitrary vector $\mathbf{f}$, we have

$$\mathbf{f}^\top \mathbf{L} \mathbf{f} = \|P\,\mathbf{f}\|^2 \geq 0 \ , \tag{24}$$

thus $\mathbf{L}$ is positive semi-definite.

To prove $\mathbf{L}\mathbf{1} = \mathbf{0}$, notice that the row sum of S is 1, *i.e.*, $S\mathbf{1} = \mathbf{1}$ (with a slight abuse of the notation, we use $\mathbf{1}$ to denote all the column vectors with proper dimensions), so we have

$$\mathbf{L}\mathbf{1} = S^\top M S \mathbf{1} = S^\top M \mathbf{1} \ . \tag{25}$$

Thus we can prove $\mathbf{L}\mathbf{1} = \mathbf{0}$ if we can prove $M\mathbf{1} = \mathbf{0}$. This is further equivalent to proving $M_i\mathbf{1} = \mathbf{0}$, $\forall i = 1, \cdots, n$. Define the symmetric matrix $B_i = (I - X_i^\top A_i)^\top (I - X_i^\top A_i)$, from Eqn. 13, we get

$$\begin{aligned} M_i\mathbf{1} &= H^\top (I - X_i^\top A_i)^\top (I - X_i^\top A_i) H\mathbf{1} \\ &= \begin{bmatrix} I \\ -\mathbf{1}^\top \end{bmatrix} B_i \begin{bmatrix} I & -\mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \begin{bmatrix} B_i & -B_i\mathbf{1} \\ -\mathbf{1}^\top B_i & \mathbf{1}^\top B_i \mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \mathbf{0} \ . \end{aligned} \tag{26}$$

Thus, we have

$$\mathbf{L}\mathbf{1} = S^\top M \mathbf{1} = S^\top \mathbf{0} = \mathbf{0} \ , \tag{27}$$

and the theorem is proved.

Proof to Theorem 2: the proof is similar to the proof to Theorem 1. To prove that D is positive semi-definite, we use $D = \tilde{P}^\top \tilde{P}$ where P is defined as:

$$P = \begin{bmatrix} A_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & A_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & A_n \end{bmatrix} S \ . \tag{28}$$

To prove $D\mathbf{1} = \mathbf{0}$, it is equivalent to prove $A_i^\top A_i \mathbf{1} = \mathbf{0}$, $\forall i = 1, \cdots, n$. To see this, we have

$$\begin{aligned} A_i^\top A_i \mathbf{1} &= H^\top [(\lambda I + X_i X_i^\top)^{-1} X_i]^\top (\lambda I + X_i X_i^\top)^{-1} X_i H \\ &= \begin{bmatrix} I \\ -\mathbf{1}^\top \end{bmatrix} \hat{B}_i \begin{bmatrix} I & -\mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \begin{bmatrix} \hat{B}_i & -\hat{B}_i \mathbf{1} \\ -\mathbf{1}^\top \hat{B}_i & \mathbf{1}^\top \hat{B}_i \mathbf{1} \end{bmatrix} \mathbf{1} \\ &= \mathbf{0} \ , \end{aligned} \tag{29}$$

where we define $\hat{B}_i$ as $\hat{B}_i = [(\lambda I + X_i X_i^\top)^{-1} X_i]^\top (\lambda I + X_i X_i^\top)^{-1} X_i$. Thus, the theorem is proved.

Learning with $L_{q<1}$ vs L_1-Norm Regularisation with Exponentially Many Irrelevant Features

Ata Kabán and Robert J. Durrant

School of Computer Science, The University of Birmingham,
Birmingham, B15 2TT, UK
A.Kaban@cs.bham.ac.uk

Abstract. We study the use of fractional norms for regularisation in supervised learning from high dimensional data, in conditions of a large number of irrelevant features, focusing on logistic regression. We develop a variational method for parameter estimation, and show an equivalence between two approximations recently proposed in the statistics literature. Building on previous work by A.Ng, we show the fractional norm regularised logistic regression enjoys a sample complexity that grows logarithmically with the data dimensions and polynomially with the number of relevant dimensions. In addition, extensive empirical testing indicates that fractional-norm regularisation is more suitable than L1 in cases when the number of relevant features is very small, and works very well despite a large number of irrelevant features.

1 $L_{q<1}$-Regularised Logistic Regression

Consider a training set of pairs $z = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^n$ drawn i.i.d. from some unknown distribution P. $\boldsymbol{x}_j \in \mathcal{R}^m$ are m-dimensional input points and $y_j \in \{-1, 1\}$ are the associated target labels for these points. Given z, the aim in supervised learning is to learn a mapping from inputs to targets that is then able to predict the target values for previously unseen points that follow the same distribution as the training data.

We are interested in problems with large number m of input features, of which only a few $r << m$ are relevant to the target. In particular, we focus on a form of regularised logistic regression for this purpose:

$$\max_{\boldsymbol{w}} \sum_{j=1}^{n} \log p(y_j|\boldsymbol{x}_j, \boldsymbol{w}) \tag{1}$$

$$\text{subject to} ||\boldsymbol{w}||_q \leq A \tag{2}$$

or, in the Lagrangian formulation:

$$\max_{\boldsymbol{w}} \sum_{j=1}^{n} \log p(y_j|\boldsymbol{x}_j, \boldsymbol{w}) - \alpha||\boldsymbol{w}||_q^q \tag{3}$$

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 580–596, 2008.

where α is the Lagrange multiplier that balances between fitting the data well and having small parameter values.

In the above, the likelihood of predicting y for some input $\boldsymbol{x}$ in logistic regression is

$$p(y|\boldsymbol{w}^T\boldsymbol{x}) = 1/(1+\exp(-y\boldsymbol{w}^T\boldsymbol{x})),$$

parameterised by $\boldsymbol{w} \in \mathcal{R}^{1\times m}$. The norm that forms the regularisation term is defined as

$$||\boldsymbol{w}||_q = (\sum_{i=1}^{m} w_i^q)^{1/q}$$

Note, with $q = 2$ or $q = 1$, this is L2- or L1-regularised logistic regression respectively. The generalisation ability and sample complexity of L2- vs. L1-regularised logistic regression have been comprehensively studied in [10] — showing the impressive superiority of the latter in problems with large m and small r. Here we seek to extend their study to the case of $q \in (0, 1)$, which we refer to as $L_{q<1}$-regularisation or 'fractional norm'-regularisation.

The fractional norm is not strictly a norm in the mathematical sense, since it does not satisfy the triangle inequality. In addition it is non-differentiable at zero and non-convex, which make its use technically more challenging than that of the more common L1 or L2 norms. Nevertheless, work in a number of disjoint areas independently indicate added value to this norm, in terms of certain specific criteria. It may be interesting to note, the fractional norms were previously studied in the databases and data engineering literature [5], for mitigating the dimensionality curse. Work in statistics [3,16] have established the oracle properties of such and related [14] non-convex regularisation, despite the existence of several local optima. Good empirical results were also reported in signal reconstruction [2] and in SVM classification [8,13]. Furthermore, using fractional norm regularisation, consistently superior empirical results were reported on real genomic data sets [7]. Related ideas of using non-convex ('zero-norm') regularisation [13] were found useful in many application settings, though this appeared to be data dependent.

It is therefore of interest to know more exactly in what conditions fractional norm regularisation would be superior to other commonly used norms for machine learning, in terms of its generalisation ability and sample complexity. The work of [10] elucidated such issues for L1 vs. L2 regularisation, but to our best of knowledge, there is no such systematic study assessing fractional norm regularisation, and this is what we attempt in this paper.

2 Analysis

The analysis presented in this section follows standard techniques in statistical learning theory, and in particular the techniques previously employed in [10] for analysing the L_1-regularised logistic regression. We give more details to show that, their results extend straightforwardly to the $L_{q<1}$-regularised case, namely that $L_{q<1}$-regularised logistic regression enjoys a logarithmic sample complexity

w.r.t. the data dimensionality, and polynomial in the number of relevant features. Thus, it can learn to generalise from data with exponentially many irrelevant features. Recall that, a logarithmic sample complexity corresponds to the best known bounds for feature selection (see e.g. [10] and references therein).

Denote by $G = \{g : g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{x} \in \mathcal{R}^m\}$ the linear function class, and $H = \{h_g(\boldsymbol{x}, y) = -\log p(y|g(\boldsymbol{x})) : g \in G, \boldsymbol{x} \in \mathcal{R}^m, y \in \{-1, 1\}\}$ is the function class under study, i.e. the parameterised negative log likelihood of the logistic regression model.

The learning algorithm we consider is the following. Divide the available labelled set z in two disjoint sets z_1 and z_2, with sizes $|z_1| = n_1, |z_2| = n - n_1$ respectively, where z_1 (the training set) is used for learning $\boldsymbol{w}$ by optimising (1)-(2) or (3) with a fixed A (or α) – this outputs the function $L(z_1) = \min_{h\in H} \hat{er}_{z_1}(h)$ – and z_2 (the validation set) is used to select the optimal A (or equivalently α).

We start by considering the probability of error[1] at the training stage of the above learning algorithm. For a given $h \in H$, denote $er_P(h) = E_{(\boldsymbol{x},y)\sim \text{iid}P}[h(\boldsymbol{x}, y)]$ the true error of h w.r.t. the unknown data distribution P under the i.i.d. sampling assumption. Further, $\hat{er}_{z_1}(h) = \frac{1}{n_1}\sum_{i=1}^{n_1} h(\boldsymbol{x}_i, y_i)$ is the sample error achieved by h on the training set z_1, and $opt_P(H) = \inf_{h\in H} er_P(h)$ is the approximation error of our function class H. $h^* \in H$ will denote the function in H that is closest to the one where this infimum is attained.

Theorem1. $\forall \epsilon > 0, \forall \delta > 0, \forall m, n_1 \geq 1$ and $\forall A \geq 0$ fixed, in order to ensure that

$$er_P(L(z_1)) \leq opt_P(H) + \epsilon \tag{4}$$

with probability $1 - \delta$, it is enough to have the following training set size:

$$n_1(L, \epsilon, \delta) = \frac{2048(A+1)^2}{\epsilon^2}[\log\frac{8(2m+1)}{\delta} + \frac{256A^2}{\epsilon^2} + 1] \tag{5}$$

Hence the sample complexity $n_1 = \Omega((\log m) \times poly(A, 1/\epsilon, \log(1/\delta))$ is logarithmic in the data dimensionality m and polynomial in A and other quantities of interest.

The proof is given in the Appendix.

In the above, the regularisation parameter A was fixed. Now the error from the validation procedure for selecting A should be considered. We employ the same hold-out validation scheme as [10], the implementation of which is to select A from the pre-defined set of possible values $\{0, 1, 2, 4, 8, ..., C\}$ such that the logloss is minimised on the hold-out set. If r denotes the number of relevant features, we have that $|w_{i_j}| \leq K, j = 1, ..., r$ for some constant K and for all others the entry in $\boldsymbol{w}$ is zero. So this gives us the following:

$$|w_{i_j}|^q \leq K^q \Rightarrow ||\boldsymbol{w}||_q^q \leq rK^q \Rightarrow ||\boldsymbol{w}||_q \leq r^{1/q}K \tag{6}$$

Therefore, from the above set of possible values, the optimal choice of A is $r^{1/q}K \leq A \leq 2r^{1/q}K$ (which recovers in the $q = 1$ case the relationship between

[1] Although the theory concerns the logloss error, an upper bound on the logloss also implies an upper bound on the 0-1 misclassification error [10].

A and r given in [10]). From (6), we see A grows polynomially with r, so with this optimal choice, the previous result also implies the sample complexity is polynomial in the number of relevant features r. Now, for the same standard arguments as in [10], the hold-out validation procedure ensures that with probability $1-\delta$ the selected parameter will have performance at most ϵ worse than that with the best parameter. Adding this together with the previous result (eq (42) in Appendix), we have that:

$$er_P(L(z_1)) \leq opt_P(H) + 3\epsilon \tag{7}$$

with probability $1-2\delta$. We can replace δ with $\delta/2$ and ϵ with $\epsilon/3$, and the sample complexity remains in the same complexity class.

Comments. It may be interesting to note, from the expression of the sample complexity we can see this model is advantageous as long as the $\log m$ term is dominant – i.e. when A is small, or equivalently, when the number of relevant features is small. The sample complexity grows with the 4-th power of A. So when the number of relevant features is large enough for this to become the dominant term, we might expect to lose the benefits of a sparsity-inducing regularisation. Also, from (6) we can see that, the smaller the $q < 1$, the faster A grows in r. Hence we might expect a small exponent q to work the best in a setting where the number of relevant features is very small.

3 Implementation

The likelihood function is not convex and is also non-differentiable at zero, which makes the implementation non-trivial.

3.1 Method 1: Using a Smooth Approximation

The following smooth approximation to the regularisation term has been proposed in [7]:

$$\sum_{i=1}^{m} |w_i|^q \approx \sum_{i=1}^{m} (w_i^2 + \gamma)^{q/2} \tag{8}$$

where γ is set to a small value. The approximate log likelihood is then differentiable and any nonlinear optimisation method applies.

Although this approach seems practically convenient and easy to implement, it has several drawbacks. The main difficulty is that there is no obvious or principled way to set the smoothing parameter γ. Ideally, one would like it to be as small as it can be without causing numerical instability problems — which is not easy to determine. As one would expect, we observed in our experiments that, the smaller the q is, the more likely the danger of running into numerical instability when γ is chosen to be too small. On the other hand, a larger γ tends to over-smooth the regularisation term, specially when q is further from zero, and this causes it to lose its beneficial sparsity-inducing effect. In addition, iterative optimisers applied to this approximation are not guaranteed to produce

an increase in the likelihood of the model at each iteration. The next section presents a more principled alternative that bypasses all of these limitations.

3.2 Method 2: Local Quadratic Variational Approximation

A local quadratic approximation was proposed in [3], which, as we shall see, is actually a strict lower bound on the model likelihood. This becomes evident by deriving it from convex duality [6].

With $q < 1$, the function $|w_i|^q$ is concave, so we can write:

$$f(w_i) = |w_i|^q = \min_{\lambda_i} \left\{ \lambda_i w_i^2 - f^*(\lambda_i) \right\} \tag{9}$$

$$f^*(\lambda_i) = \min_{\eta_i} \left\{ \lambda_i \eta_i^2 - f(\eta_i) \right\} \tag{10}$$

In convex analysis, the function $f^*(.)$ is termed the conjugate (or dual) function of $f(.)$. Geometrically, $f^*(\lambda_i)$ represents the amount of vertical shift applied to λw_i^2 to obtain the quadratic upper bound with precision parameter λ, that touches $f(w_i)$.

Denoting $g(\eta_i) = \lambda_i \eta_i^2 - f(\eta_i)$, the maximum occurs either at $\eta_i = 0, g(\eta_i = 0) = 0$ or at a solution of the stationary equation when $\eta_i \neq 0$:

$$g'(\eta_i) = 2\lambda_i \eta_i - f'(\eta_i) = 0 \quad \Rightarrow \lambda_i = \frac{f'(\eta_i)}{2\eta_i} \tag{11}$$

and $f'(\eta_i) = q|\eta_i|^{q-1} sign(\eta_i)$. Replacing in (9) yields the variational bound:

$$|w_i|^q \leq \frac{f'(\eta_i)}{2\eta_i}(w_i^2 - \eta_i^2) + f(\eta_i) = \frac{1}{2} \;\; q|\eta_i|^{q-2} w_i^2 + (2-q)|\eta_i|^q \tag{12}$$

The new parameters η_i are called variational parameters, and the resulting upper bound is tangent to $|w_i|^q$ in $\eta_i = \pm|w_i|$ — see Figure 1.

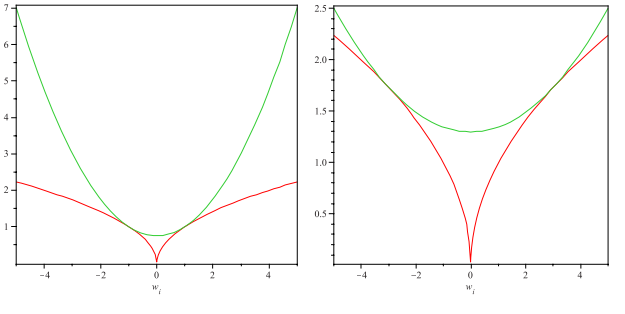


Fig. 1. Examples: Left: $q = 0.5$, $\eta_i = 1$, so the quadratic upper bound is tangent in ± 1. Right: $q = 0.5$, $\eta_i = 3$, so the quadratic upper bound is tangent in ± 3.

It is interesting to note that in the case of $q = 1$, the bound (12) recovers exactly the bound proposed in [4] for deriving an exact estimation algorithm for L1-regularised logistic regression.

Using the above local bounds, the log likelihood is lower-bounded:

$$\begin{aligned}\mathcal{L} &= -\sum_{j=1}^{n} \log\left\{1+\exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} - \alpha \sum_{i=1}^{m} |w_i|^q \\ &\geq -\sum_{j=1}^{n} \log\left\{1+\exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} - \alpha \sum_{i=1}^{m} \frac{1}{2}\left\{q|\eta_i|^{q-2} w_i^2 + (2-q)|\eta_i|^q\right\} \\ &= \tilde{\mathcal{L}}^{quad}(\boldsymbol{w}, \boldsymbol{\eta}) \end{aligned} \tag{13}$$

Now, maximising the lower bound to the log likelihood can be done iteratively. Each iteration will alternate between maximising w.r.t. $\boldsymbol{w}$ while keeping $\boldsymbol{\eta}$ fixed and maximising w.r.t. the variational parameters $\boldsymbol{\eta}$, i.e. tightening the variational bound while keeping $\boldsymbol{w}$ fixed. Convergence to a local optimum is guaranteed, convergence proofs for this kind of algorithms are detailed in e.g. [6] and [3], and are essentially based on the fact that the sequence of log likelihood estimates is non-decreasing and bounded from above (analogous to E-M algorithms).

Consider first the maximisation w.r.t. the variational parameters $\boldsymbol{\eta} = (\eta_1, \eta_2, ..., \eta_n)$, with $\boldsymbol{w}$ being fixed to their current value. Solving the stationary equations w.r.t. η_i, i.e. $\frac{\partial \tilde{\mathcal{L}}^{quad}}{\partial \eta_i} = 0$ yields:

$$\boldsymbol{\eta} = |\boldsymbol{w}| \tag{14}$$

This is indeed where we have seen the bound touches the function.

Now, the maximisation w.r.t. w, with fixed $\boldsymbol{\eta}$ is technically an L2-regularised logistic regression problem, since (13) depends quadratically on $\boldsymbol{w}$, which may be carried out using existing methods. One convenient option, which we briefly reproduce here for completeness, is to employ the local quadratic lower bound to the log likelihood term as proposed in [6], based on the fact that the logistic function is convex as a function of the square root of its argument. This is the following,

$$\begin{aligned}-\log\left\{1+\exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} \geq -\frac{1}{4\xi_j}\tanh(\frac{\xi_j}{2})\left\{(\boldsymbol{w}^T \boldsymbol{x}_j)^2 - \xi_j^2\right\} \\ + (y_j \boldsymbol{w}^T \boldsymbol{x}_j - \xi_j)/2 - \log\left\{1+\exp(\xi_j)\right\}\end{aligned} \tag{15}$$

where $\xi_j, j = 1, ..., n$ are new variational parameters that control the tightness of the locally quadratic bound on the log likelihood. Replacing this into (13), we obtain a lower bound expression on the log likelihood (parameterised by both $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$) which we can maximise iteratively by solving for $\boldsymbol{w}$ and $\boldsymbol{\xi}$ in turn. These optimisation problems now have closed form solutions, and we get:

$$\begin{aligned}\boldsymbol{w} &= [\boldsymbol{X}\boldsymbol{\Xi}\boldsymbol{X}^T + \boldsymbol{\Lambda}]^{-1}\boldsymbol{X}\boldsymbol{y}/2 \\ &= \left\{\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{\Lambda}\boldsymbol{X} + \boldsymbol{\Xi}]^{-1}\boldsymbol{X}^T\boldsymbol{\Lambda}\right\}\boldsymbol{X}\boldsymbol{y}/2 \end{aligned} \tag{16}$$

$$\boldsymbol{\xi} = |\boldsymbol{X}^T\boldsymbol{w}| \tag{17}$$

where $\boldsymbol{\Lambda} = \text{diag}(\frac{|\eta_i|^{2-q}}{q\alpha})$ and $\boldsymbol{\Xi} = \text{diag}(\frac{2\xi_j}{\tanh(\xi_j/2)})$, and the form (16) is more convenient in the $m >> n$ case. This inner loop can then be interleaved with the re-estimation of $\boldsymbol{\eta}$ and so the overall algorithm consists of performing (16)-(17)-(14) in turn until convergence to a local optimum.

It should be noted that, although each step of the algorithm yields a unique solution for the parameter being re-estimated, the overall solution is not unique since the likelihood (and the bound on it) is not convex. There are multiple local optima, and so the initialisation may be important. In the reported experiments, we initialised all variational parameters to one. Then, cf. (16), $\boldsymbol{w}$ is a deterministic function of these values and the training set. While this is just one of several possible reasonable choices, it worked well, and as we shall see, even the local optima obtained is able to produce more accurate classification (and feature recovery) than the alternative convex approach of L_1-regularised logistic regression.

3.3 Method 3: Local Linear Variational Approximation

Now, convex duality [6] will be used to create a linear upper bound on the regularisation term. The idea of creating local linear (rather than quadratic) bounds for this problem was proposed in a recent statistics paper [16], and our use of convex duality is just a convenient framework for deriving variational bounds in a more systematic manner.

With $q < 1$, the function $|w_i|^q$ is concave, so we can write:

$$f(w_i) = |w_i|^q = \min_{\lambda_i} \{\lambda_i |w_i| - f^*(\lambda_i)\} \tag{18}$$

$$f^*(\lambda_i) = \min_{\eta_i} \{\lambda_i |\eta_i| - f(\eta_i)\} \tag{19}$$

Again, the function $f^*(.)$ is the conjugate (or dual) function of $f(.)$, and $f^*(\lambda_i)$ represents the amount of vertical shift to be applied to $\lambda|w_i|$ in order to obtain the linear upper bound with slope λ, that touches $f(w_i)$. For every $\pm|w_i|$, there is an optimal slope λ_i from the family of upper bounds.

Denoting $g(\eta_i) = \lambda_i|\eta_i| - f(\eta_i)$, the maximum occurs either at $\eta_i = 0, g(\eta_i = 0) = 0$ or at a solution of the stationary equation when $\eta_i \neq 0$:

$$g'(\eta_i) = \lambda_i \text{sign}(\eta_i) - f'(\eta_i) = 0 \quad \Rightarrow \tag{20}$$

$$\lambda_i = \frac{f'(\eta_i)}{\text{sign}(\eta_i)} = q|\eta_i|^{q-1} \quad \text{and} \tag{21}$$

$$f^*(\lambda_i) \leq (q-1)|\eta_i|^q \tag{22}$$

Replacing in (18) yields the variational bound:

$$|w_i|^q \leq q|\eta_i|^{q-1}|w_i| + (1-q)|\eta_i|^q \tag{23}$$

The new parameters η_i are called variational parameters, and the resulting upper bound is tangent to $|w_i|^q$ in $\eta_i = \pm|w_i|$ — see Figure 2.

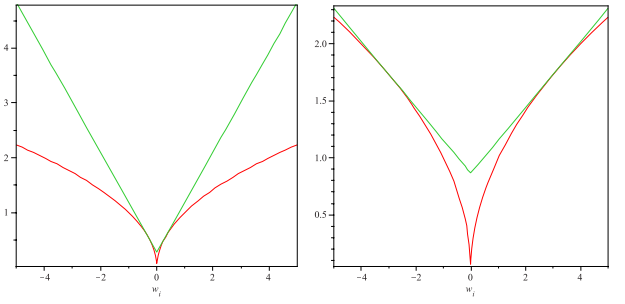


Fig. 2. Examples - Left: $q = 0.5$, $\eta_i = 0.3$, so the linear upper bound is tangent in ± 0.3. Right: $q = 0.5$, $\eta_i = 3$, so the linear upper bound is tangent in ± 3.

This variational approximation casts back the initial problem of fractional regularisation to solving a number of L1-regularised problems instead.

$$\begin{aligned}\mathcal{L} &= -\sum_{j=1}^{n} \log\left\{1 + \exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} - \alpha \sum_{i=1}^{m} |w_i|^q \\ &\geq -\sum_{j=1}^{n} \log\left\{1 + \exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} - \alpha \sum_{i=1}^{m} \left\{q|\eta_i|^{q-1}|w_i| + (1-q)|\eta_i|^q\right\} \\ &= \tilde{\mathcal{L}}^{lin}(\boldsymbol{w}, \boldsymbol{\eta}) \end{aligned} \tag{24}$$

As before, maximising the lower bound to the log likelihood can be done iteratively. Each iteration will alternate between maximising w.r.t. $\boldsymbol{w}$ — which is now an L1-regularised version of the problem — and maximising w.r.t. the variational parameters $\boldsymbol{\eta}$, i.e. tightening the bound. Convergence to a local optimum is guaranteed, proofs are detailed in e.g. [6] and [16].

Maximising w.r.t. the variational parameters is straightforward, and after some algebra we get the stationary equation,

$$\frac{\partial \tilde{\mathcal{L}}^{lin}}{\partial \eta_i} = q(q-1)|\eta_i|^{q-2} sign(\eta_i)(|w_i| - |\eta_i|) = 0 \tag{25}$$

the solution of which is, as one would expect, $|\eta_i| = |w_i|$. Due to symmetry, since the objective is a function of $|\eta_i|$, it is enough to take:

$$\boldsymbol{\eta} = |\boldsymbol{w}| \tag{26}$$

For maximising w.r.t. $\boldsymbol{w}$, we can use any of the several existing efficient methods for solving an L_1-regularised logistic regression. The algorithm is then to iterate between these two steps till convergence, and we see this requires us to solve an L1-regularised regression problem in each iteration. However, in the sequel we shall show instead, that — perhaps surprisingly — the method described in this section is actually equivalent to the previous one.

Equivalence of Local Linear and Local Quadratic Approximation. Of course, the local linear approximation appears to be a tighter bound to the $L_{q<1}$-term than the local quadratic one, which has been the main motivation for [16] proposing it. However, we will show in fact they are both finding a local optimum of the same objective. To see this, we develop a generalised E-M [9] estimation algorithm for the local linear bound, which will turn out to be identical to the iterative estimation algorithm we have developed for the local quadratic bound.

Let us start by rewriting the $|w_i|$ term as follows:

$$|w_i| = -\log \int_0^\infty \frac{1}{\sqrt{2\pi\tau_i}} \exp\left\{-\frac{w_i^2+\tau_i^2}{2\tau_i}\right\} d\tau_i \tag{27}$$

$$= -\log\left\{2\int_0^\infty N(w_i|0,\tau_i)Exp(\tau_i)d\tau_i\right\} \tag{28}$$

where $N(w_i|0,\tau_i) = \frac{1}{\sqrt{2\pi\tau_i}}\exp(-w_i^2/(2\tau_i))$ is the Gaussian density with zero men and variance τ_i and $Exp(\tau_i) = \frac{1}{2}\exp(-\tau_i/2)$ is the exponential density with parameter 1.

In this rewriting, τ may be seen as a latent variable, so we could use the E-M methodology for iterative estimation of $\boldsymbol{w}$ from (24) (in an inner-loop) while keeping $\boldsymbol{\eta}$ fixed at its currently estimated value (from the outer loop). We should emphasise, this is not a proposal for a practical algorithm, but serves to show the equivalence relationship with the local quadratic approximation approach — which is obviously more convenient to implement.

It is well known from the theory of E-M [9] that the expectation of the log complete likelihood forms a so-called auxiliary function, meaning that an increase in this function corresponds to an increase in the initial objective (in our case $\tilde{\mathcal{L}}^{lin}(\boldsymbol{w},\boldsymbol{\eta})$), and a local optimum of the auxiliary function is also a local optimum of the initial objective. For (24), the expected log complete likelihood is the following:

$$Q(\boldsymbol{w}, p(\boldsymbol{\tau}|\boldsymbol{w}^{old}), \boldsymbol{\eta}) = -\sum_{j=1}^{n}\log\{1+\exp(-y_j\boldsymbol{w}^T\boldsymbol{x}_j)\} + \alpha q\sum_{i=1}^{m}|\eta_i|^{q-1}\int_0^\infty p(\boldsymbol{\tau}|\boldsymbol{w}^{old})\times$$
$$\times\{\log N(w_i|0,\tau_i) + \log Exp(\tau_i)\} + const_{\boldsymbol{w},\boldsymbol{\tau}} \tag{29}$$

where $const_{\boldsymbol{w},\boldsymbol{\tau}}$ is independent of both $\boldsymbol{w}$ and $\boldsymbol{\tau}$.

The E-step of this inner loop estimation procedure would then be to compute the posterior $p(\tau_i|w_i^{old})$ and the M-step would maximise Q with respect to $\boldsymbol{w}$. Observe, however, that the only posterior statistic required for the estimation of $\boldsymbol{w}$ is the expectation $E[1/\tau_i|w_i^{old}]$ w.r.t. $p(\tau_i|w_i^{old})$. Thus, computing this completes the E-step:

$$E[1/\tau_i|w_i^{old}] = \frac{\int_0^\infty 1/\tau_i N(w_i^{old}|0,\tau_i)Exp(\tau_i)d\tau_i}{\int_0^\infty N(w_i^{old}|0,\tau_i)Exp(\tau_i)d\tau_i} = \frac{1}{|w_i^{old}|} \tag{30}$$

Hence we have, in more compact notation:

$$E[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{old}] = |\boldsymbol{w}^{old}|^{-1} \tag{31}$$

which, after one E-step, is identical to the inverse of the estimate of $\boldsymbol{\eta}$ that we obtained previously (26).

The M-step is to compute:

$$\boldsymbol{w} = \operatorname*{argmax}_{\boldsymbol{w}} Q(\boldsymbol{w}, E[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{old}], \boldsymbol{\eta}) \tag{32}$$

$$= \operatorname*{argmax}_{\boldsymbol{w}} -\sum_{j=1}^{n} \log\left\{1 + \exp(-y_j \boldsymbol{w}^T \boldsymbol{x}_j)\right\} - \alpha q \sum_{i=1}^{m} \frac{1}{2} |\eta_i|^{q-1} \mathrm{E}[1/\tau_i | w_i^{old}] w_i^2 \tag{33}$$

where we replaced (31) into the terms of Q that depend on $\boldsymbol{w}$. Observe, in the first M-step this is exactly the same as $\operatorname*{argmax}_{\boldsymbol{w}} \tilde{\mathcal{L}}^{quad}(\boldsymbol{w}, \boldsymbol{\eta})$.

Now, rather than *maximising* $\tilde{\mathcal{L}}^{lin}(\boldsymbol{w}, \boldsymbol{\eta})$ w.r.t. $\boldsymbol{w}$ with $\boldsymbol{\eta}$ fixed, by iterating the E and M steps to convergence in an inner-loop (and reestimate $\boldsymbol{\eta}$ in the outer loop), it is sufficient, cf. the generalised E-M [9], to make an *increase* in $\tilde{\mathcal{L}}^{lin}(\boldsymbol{w}, \boldsymbol{\eta})$ w.r.t. $\boldsymbol{w}$ before reestimating $\boldsymbol{\eta}$. This leads to merging the inner and outer loops in a single loop, while still having the guarantee of a monotonic convergence to a local optimum of $\tilde{\mathcal{L}}^{lin}(\boldsymbol{w}, \boldsymbol{\eta})$ w.r.t. $\boldsymbol{w}$. In particular, we have the following convergent sequence, where t is the iteration index:

$$\begin{aligned}
\tilde{\mathcal{L}}^{lin}(\boldsymbol{w}^{t-1}, \boldsymbol{\eta}^{t-1}) &\leq \tilde{\mathcal{L}}^{lin}(\boldsymbol{w}^{t-1}, \boldsymbol{\eta}^{t}) = Q(\boldsymbol{w}^{t-1}, \mathrm{E}[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{t-1}], \boldsymbol{\eta}^{t}) \\
&\leq Q(\boldsymbol{w}^{t}, \mathrm{E}[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{t-1}], \boldsymbol{\eta}^{t}) = \tilde{\mathcal{L}}^{quad}(\boldsymbol{w}^{t}, \boldsymbol{\eta}^{t}) \\
&\leq Q(\boldsymbol{w}^{t}, \mathrm{E}[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{t}], \boldsymbol{\eta}^{t}) \\
&= \tilde{\mathcal{L}}^{lin}(\boldsymbol{w}^{t}, \boldsymbol{\eta}^{t}) \leq \tilde{\mathcal{L}}^{lin}(\boldsymbol{w}^{t}, \boldsymbol{\eta}^{t+1}) = Q(\boldsymbol{w}^{t}, \mathrm{E}[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{t}], \boldsymbol{\eta}^{t+1}) \\
&\leq Q(\boldsymbol{w}^{t+1}, \mathrm{E}[\boldsymbol{\tau}^{-1}|\boldsymbol{w}^{t}], \boldsymbol{\eta}^{t+1}) = \tilde{\mathcal{L}}^{quad}(\boldsymbol{w}^{t+1}, \boldsymbol{\eta}^{t+1}) \leq ...
\end{aligned}$$

Now, observe, this is in fact identical to the monotonic sequence produced when optimising the local quadratic bound. Hence there is no advantage to the extra sophistication brought by the local linear approximation. Because of this, we implemented and used the former in the reported experiments.

4 Experiments

From the theoretical analysis we found they both enjoy a logarithmic sample complexity w.r.t. the data dimension and so they can both learn with exponentially many irrelevant features. We may also expect in the light of that analysis that a smaller q should work best in a setting where the number of relevant features is very small. However, to find out how $L_{q<1}$-regularisation compares with L_1 regularisation in such a setting, we conduct an empirical comparison in this section.

We generated synthetic data sets as in [10] and followed the same experimental protocol in the first instance. In each experiment, 30% of the data was used as a validation set to select the regularisation parameter. The training + validation set size was 70+30 and the performance was measured on an independent test set of size 100. The number of features was varied between 100 and 1000, out

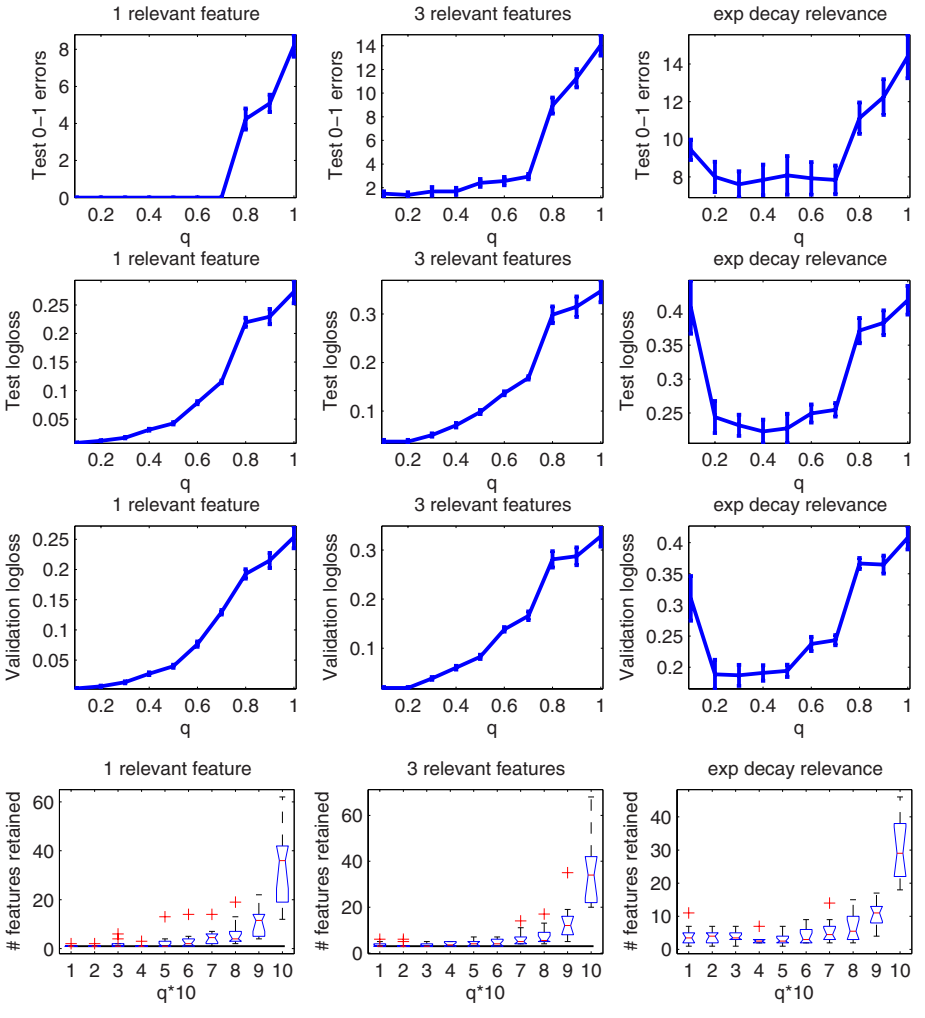


Fig. 3. Results on synthetic data sets when varying q. The training set size was $n_1 = 70$, the validation set size=30, and the out-of-sample test set size=100. The statistics are over 10 independent runs with overall feature size (dimensionality) ranging from 100 to 1000. The upper figures represent average ± standard error.

of which just a few are relevant, as follows. We created and studied the same 3 types of data sets as in [10]: (1) a single feature is relevant; (2) 3 features are relevant; and (3) exponentially decaying relevance of the features. See [10] for more details on the data generation procedure. Figure 3 summarises the results, measured by three different criteria: the number of 0-1 errors on the test set (out of 100), the logloss achieved on the test set and the number of features retained vs. the true number of relevant features. We also plotted the loglosses on validation set, to ascertain these are in good agreement with the

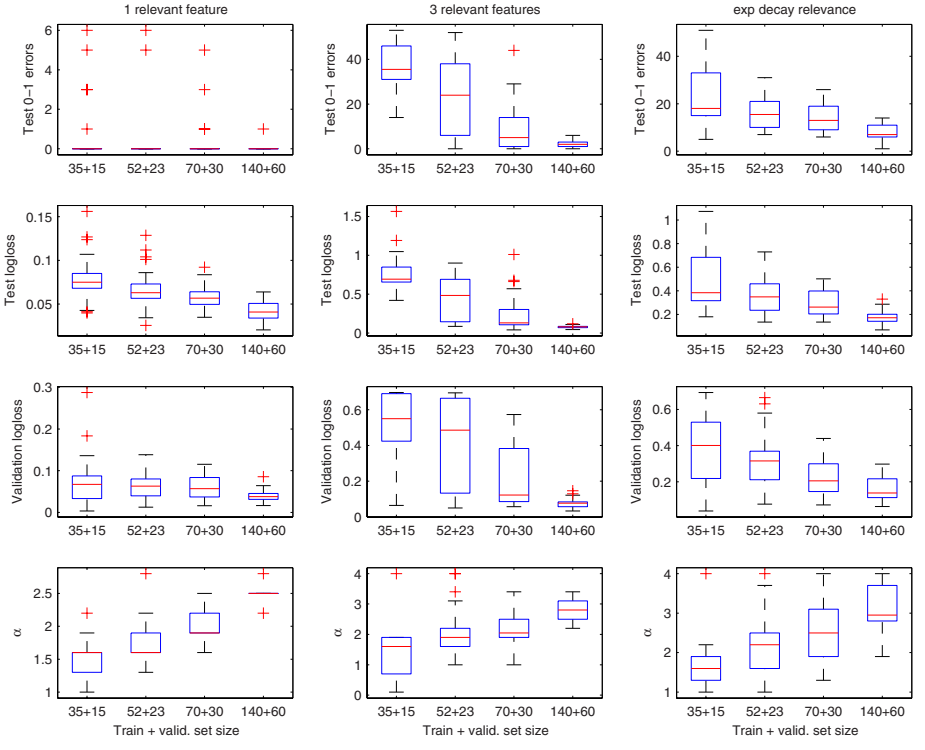


Fig. 4. Varying the training + validation set size. q=0.5, m=3000. The box-plots summarise 20 independent trials for each experiment.

other measures. For each q, the error bars represent averages and standard errors of results obtained for all data dimensions tested. As we can see, the variation w.r.t. this factor turns out to be much smaller than that w.r.t. varying q. It is clear from the figure that small values of q work extremely well when only one feature is relevant (column 1), despite of the large number of relevant dimensions. The improvement achieved over the L_1-regularised logistic regression is both statistically and practically significant, w.r.t. all measures. The picture is very similar when 3 features are relevant (column 2). Moreover, we notice that for certain values of q, the fractional-norm regularisation is also able to improve over L_1 in the case of exponentially decaying feature relevance (column 3). However, unsurprisingly, small values are not the best in this latter case — since in this case all features have some degree of information about the target, a relatively small but not too small number of them is needed for good prediction.

Next, noting that the theoretical guarantees assume a large number of examples, and although the previous set of experiments considered a small but fixed sample size throughout, we conducted an additional set of experiments designed to test the variability of L_q-regularised logistic regression when varying

the number of samples in both directions. Here $q = 0.5$ is fixed, as from the previous experiments we observe this value tends to win over on average. Figure 4 presents these results for all three types of data sets previously considered. The overall dimensionality is 3000 this time, so these data sets must be harder than the previous ones. The size of the independent test set is 100 in all cases, so the misclassifications reported are again out of 100.We see the results remain reasonably good as long as the ratio of sample size vs. relevant dimensions is not too small. The results with a single relevant feature in 3000 dimensions (column 1) are in fact excellent — even with very small sample sizes the median of the 0-1 missclassification error rate is still zero. In turn, as the number of relevant features increases to 3 and the training + validation size is as small as 35+15, we see in the leftmost box-plot (middle column) this is where the method reaches its limits and ceases to work.

5 Conclusions

We studied fractional-norm regularisation for logistic regression both theoretically and empirically, for high dimensional data with many irrelevant features. We developed a variational method for parameter estimation, and have shown an equivalence between local quadratic and local linear approximations to the regularisation term. Based on our results, fractional-norm regularisation is more suitable than L1 in cases when the number of relevant features is very small, and works very well despite a large number of irrelevant features.

A Word about a Bayesian Interpretation. $L_{q<1}$ regularisation may be interpreted as the MAP estimate of a Generalised Laplacian Distribution prior. Our variational approach makes it tractable to compute variational posterior distributions and hence to obtain uncertainty estimates. Whether those will keep having such favourable sample complexity properties or not, remains to be elucidated in the future.

Acknowledgments. RJD was supported by an EPSRC CTA studentship for MSc in Natural Computation.

References

1. Anthony, M., Bartlett, P.L.: Neural Network Learning: Theoretical Foundations. Cambridge University Press, Cambridge (2001)
2. Chartrand, R.: Exact reconstructions of sparse signals via non-convex minimization. IEEE Signal Process. Lett. 14, 707–710 (2007)
3. Fan, J., Li, R.: Variable Selection via Non-concave Penalized Likelihood and its Oracle Properties. Journal of the American Statistical Association, Theory and Methods 96(456) (December 2001)
4. Krishnapuram, B., Carin, L., Figueiredo, M., Hartemink, A.: Learning sparse Bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(6), 957–968 (2005)

5. François, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE Trans. on Knowledge and Data Engineering 19(7) (July 2007)
6. Jordan, M., Ghahramani, Z., Jaakkola, T., Saul, L.: An Introduction to Variational Methods for Graphical models. In: Jordan, M. (ed.) Learning in Graphical Models. The MIT Press, Cambridge (1998)
7. Liu, Z., Jiang, F., Tian, G., Wang, S., Sato, F., Meltzer, S.J., Tan, M.: Sparse Logistic Regression with Lp Penalty for Biomarker Identification. Statistical Applications in Genetics and molecular Biology 6(1) (2007)
8. Bradley, P.S., Mangasarian, O.L.: Feature Selection via Concave Minimization and Support Vector Machines. In: ICML 1998, pp. 82–90 (1998)
9. McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions. JohnWiley and Sons, New York (1997)
10. Ng, A.: Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In: ICML 2004 (2004)
11. Pollard, D.: Empirical Processes:Theory and Applications. Springer, Heidelberg (1984)
12. Tipping, M.: Sparse Bayesian Learning and the Relevance Vector Machine. Journal of Machine Learning Research 1, 211–244
13. Weston, J., Elisseeff, A., Schölkopf, B., Tipping, M.: Use of the Zero-Norm with Linear Models and Kernel Methods. Journal of Machine Learning Research 3, 1439–1461 (2003)
14. Wipf, D.P., Rao, B.D.: ℓ_0-Norm Minimization for Basis Selection. In: Saul, L., Weiss, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems, vol. 17. MIT Press, Cambridge (2005)
15. Zhang, T.: Covering number bounds of certain regularized linear function classes. Journal of Machine Learning Research 2, 527–550 (2002)
16. Zou, H., Li, R.: One-step sparse estimates in non-concave penalized likelihood models. The Annals of Statistics (2008)

Appendix

Proof of Theorem 1

First, it is useful to notice that

$$\forall \boldsymbol{w}, ||\boldsymbol{w}||_{q<1} \geq ||\boldsymbol{w}||_1 \tag{34}$$

The plan is then the following. We show that $h \in H$ is bounded, so we can apply standard results to bound its error probability by a uniform covering number. This will then be bounded further using (34) and an existing result of [15] for regularised function classes, in the same manner as previously employed in [10]. Finally, the sufficient sample complexity is computed by requiring the error bound to be smaller or equal than the user-defined confidence parameter δ.

To see (34), note first that the inequality $||\boldsymbol{w}||_q \geq ||\boldsymbol{w}||_p, \forall \boldsymbol{w}$ is well known for $1 < q < p$ from measure theory. Extending it to $0 < q < p < 1$ is straightforward, rewrite the required inequality as follows:

$$\left(\sum_{i=1}^{m}|w_i|^q\right)^{1/q} \geq \left(\sum_{i=1}^{m}|w_i|^p\right)^{1/p}$$

$$\left(\sum_{i=1}^{m}|\frac{w_i}{(\sum_{i=1}^{m}|w_i|^p)^{1/p}}|^q\right)^{1/q} \geq \left(\sum_{i=1}^{m}|\frac{w_i}{(\sum_{i=1}^{m}|w_i|^p)^{1/p}}|^p\right)^{1/p} = 1$$

$$\left(\sum_{i=1}^{m}|u_i|^q\right)^{1/q} \geq \left(\sum_{i=1}^{m}|u_i|^p\right)^{1/p} = 1$$

where we denoted $|u_i| \equiv |\frac{w_i}{(\sum_{i=1}^{m}|w_i|^p)^{1/p}}|$. Now, since $\sum_{i=1}^{m}|u_i|^p = 1 \Rightarrow |u_i|^p \leq 1 \Rightarrow |u_i| \leq 1$. In consequence, and since $q \leq p$, we have $|u_i|^q \geq |u_i|^p$. Summing both sides w.r.t. i yields $\sum_{i=1}^{m}|u_i|^q \geq \sum_{i=1}^{m}|u_i|^p$. But we know the r.h.s. is one, so $\sum_{i=1}^{m}|u_i|^q \geq 1$. Finally, raising both sides to $1/q$, the required result follows (for any $q > 0$), i.e. $(\sum_{i=1}^{m}|u_i|^q)^{1/q} \geq 1$.

To see that $h \in H$ is bounded, we write:

$$M = |-\log p(y|\boldsymbol{w}^T\boldsymbol{x})| = |\log(1+\exp(-y\boldsymbol{w}^T\boldsymbol{x}))| \leq 1 + |\boldsymbol{w}^T\boldsymbol{x}| \quad (35)$$

$$\leq 1 + ||\boldsymbol{w}||_1||\boldsymbol{x}||_\infty \quad \text{(by Hölder's inequality)} \quad (36)$$

$$\leq 1 + ||\boldsymbol{w}||_{q<1}||\boldsymbol{x}||_\infty \quad \text{cf. (34)} \quad (37)$$

$$\leq 1 + A||\boldsymbol{x}||_\infty \quad (38)$$

Therefore, the classical result due to Pollard [11] (pp. 492), combined with standard results [10,1] applies, and we have the error probability bounded by a uniform covering number of the linear function class G in the L2-norm:

$$P\left\{\exists h \in H : |er_P(h) - \hat{er}_{z_1}(h)| > \epsilon\right\} \leq 8\mathcal{N}_2(G, \frac{\epsilon}{8L}, n_1)\exp\left\{-\frac{n_1\epsilon^2}{512M^2}\right\} \quad (39)$$

where M is the output bound that we previously computed and L is the Lipschitz constant of $h \in H$ as a function of $g \in G$. Since h is continuous on $[0,1]$ and differentiable on $(0,1)$, and $|h'(t)| = |\frac{d}{dt}\log\frac{1}{1+e^{-t}}| = |\frac{e^{-t}}{1+e^{-t}}| \leq 1$, therefore h satisfies the first order Lipschitz condition with Lipschitz constant L=1.

Now, in the above, it remains to approximate the uniform covering number[2] $\mathcal{N}_2(G, \epsilon/8, n_1)$. This is a combinatorial quantity that expresses the complexity of a function class at the given scale, and as such, it is affected by the regularisation constraints imposed. Similarly to the approach in [10], the following result by

[2] The definition of uniform covering numbers is as follows (see e.g. [1]). B is called an ε-cover for a real valued function class F of infinite cardinality (e.g. a parameterised function class) w.r.t. a particular training set z of size n and a distance d, if B is a finite set of functions, and $\forall f \in F, \exists b \in B$, such that $d(f(\boldsymbol{x}) - b(\boldsymbol{x})) < \varepsilon$. The size of the smallest such cover set B is the covering number of F w.r.t. z. The uniform covering number is then the maximum of these w.r.t. all training sets of size n and is denoted $\mathcal{N}_d(F, \varepsilon, n)$.

Zhang [15], developed for regularised linear function classes, can be used to bound this uniform covering number.

Lemma [15]. In a linear function class $G = \{g : g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{x} \in \mathcal{R}^m\}$, if $||\boldsymbol{x}||_p \leq b$ and $||\boldsymbol{w}||_q \leq a$, where $1/p + 1/q = 1$ and $p \geq 2$, then

$$\log_2 \mathcal{N}(G, \epsilon, n) \leq \text{ceil}(\frac{a^2b^2}{\epsilon^2}) \log_2(2m+1)$$

where n is the sample size and m is the data dimension.

Clearly, the lemma does not apply directly to our case, since $1/q > 1$ already when $q < 1$. However using again (34), it follows by transitivity that $||\boldsymbol{w}||_1 \leq A$. Now applying the Lemma, we get:

$$\log_2 \mathcal{N}(G^{(q<1)}, \epsilon, n) \leq \text{ceil}(\frac{A^2b^2}{\epsilon^2}) \log_2(2m+1)$$

where the $L_{q<1}$-regularised linear function class is denoted by $G^{(q<1)}$, and $b = \max_i ||\boldsymbol{x}_{i\geq n}||_\infty$ and $||\boldsymbol{x}||_\infty \equiv \max_{j\geq m} |x_j|$.

Replacing these results into the generalisation bound (39) and assuming the data is normalised such that $||\boldsymbol{x}_i||_\infty \leq 1$, we get

$$P\{\exists h \in H : |er_P(h) - \hat{er}_{z_1}(h)| > \epsilon\} \leq 8 \times 2^{64A^2/\epsilon^2+1}(2m+1) \exp \quad -\frac{n_1\epsilon^2}{512(1+A)^2} \tag{40}$$

Since we want this to be small, we set the r.h.s. of (40) to be smaller or equal to a desired (user-prescribed) confidence parameter δ, we seek to ensure that h is uniformly ϵ-good with probability at least $1-\delta$. Then, with high probability, i.e. with probability $1-\delta$, we have, from (40), that:

$$\forall h \in H, \ |er_P(h) - \hat{er}_{z_1}(h)| \leq \epsilon \tag{41}$$

Now, it is a standard result to show that from (41) it follows that $er_P(L(z_1))$ is close to $opt_P(H)$, which follows below for completeness. Indeed, if (41) holds then it must hold also for our learning algorithm $h = L(z_1)$. Hence, applying (41), the definition of $L(z_1)$, and the definition of $opt_P(H)$, we get:

$$\begin{aligned} er_P(L(z_1)) &\leq \hat{er}(L(z_1)) + \epsilon = \min_{h\in H} \hat{er}_{z_1}(h) + \epsilon \\ &\leq \hat{er}_{z1}(h^*) + \epsilon \leq er_P(h^*) + 2\epsilon = \inf_{h\in H} er_P(h) + 2\epsilon = opt_P(H) + 2\epsilon \end{aligned}$$

Thus,

$$er_P(L(z_1)) \leq opt_P(H) + 2\epsilon \tag{42}$$

which is indeed of the form (4) if we replace ϵ by $\epsilon/2$.

Finally, solving for n_1 the r.h.s. of (40)$= \delta$ when ϵ is replaced by $\epsilon/2$, yields the sample complexity of $L(z_1)$, i.e. the amount of data required for good generalisation:

$$n_1(L, \epsilon, \delta) = \frac{2048(A+1)^2}{\epsilon^2}[\log \frac{8(2m+1)}{\delta} + \frac{256A^2}{\epsilon^2} + 1] \tag{43}$$

We can now conclude that the sample complexity of fractional norm regularised logistic regression with a given regularisation parameter A is $n_1 = \Omega(\log(m) \times \text{poly}(A, 1/\epsilon, \log(1/\delta)))$. Moreover, following the argument in [10] to express n_1 as a linear function of n, i.e. $n_1 = (1 - \nu)n$ where $\nu < 1$ is a constant, the sample complexity result obtained for n_1 also extends to n, i.e. we have $n = \Omega(\log(m) \times \text{poly}(A, 1/\epsilon, \log(1/\delta)))$. Q.E.D.

Catenary Support Vector Machines

Kin Fai Kan and Christian R. Shelton

Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521, USA
{kkan,cshelton}@cs.ucr.edu

Abstract. Many problems require making sequential decisions. For these problems, the benefit of acquiring further information must be weighed against the costs. In this paper, we describe the *catenary support vector machine* (catSVM), a margin-based method to solve sequential stopping problems. We provide theoretical guarantees for catSVM on future testing examples. We evaluated the performance of catSVM on UCI benchmark data and also applied it to the task of face detection. The experimental results show that catSVM can achieve a better cost tradeoff than single-stage SVM and chained boosting.

1 Introduction

Many problems require making sequential decisions. In product testing, parts are inspected throughout the manufacturing process. Humans or computers must decide whether to continue manufacturing or whether to stop (in case the piece is not salvageable). In medical diagnosis, doctors, patients, and insurers must decide whether the current information is sufficient to make a decision or whether to conduct the next of a bank of tests. In both of these cases, the benefit of further processing must be weighed against the costs. The acquisition of new information is costly.

In object detection in images, a similar problem is faced. Scanning an image for an object of interest takes processing time. If the image can be scanned more quickly or at a lower resolution (reducing the number of pixels to be examined), the detection can be sped up. In doing so, the speed of detection must be weighed against the accuracy of detection.

Most classification methods assume full information about testing examples and are thus not suitable for sequential decision making scenarios. Recently, Shelton et al. [1] proposed chained boosting to solve sequential stopping problem. They assume that the relative costs of stopping at each stage are known and can be made explicit. Given the stopping costs for each training example, the goal is to minimize the cost of the decision rules applied to future examples. The difficulty of the problem lies in the fact that the decisions in later stages depend on what happens in early stages. Motivated by the success of support vector machines (SVMs) in many classification problems, this paper presents the *catenary support vector machine* (catSVM), a margin-based method to solve sequential stopping problems.

2 Related Work

We are interested in direct estimation of a sequence of decision rules. For this reason we are not considering density estimation (like a hidden Markov model) followed by a cost

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 597–610, 2008.

analysis to derive the decision rules. This rules out approaches like influence diagrams [2] as we would like to skip the density estimation step.

Our formulation (see the next section) appears similar to cascade classification [3,4,5] in that there are stages of classification. For applications like face detection, negative examples are far more frequent than positive examples. Rejecting negative examples as quickly as possible is crucial to the speed of the classification process. Viola and Jones [3] propose an iterative approach to train the cascade. In each iteration, a new stage is added to the cascade and a new stage classifier is trained to achieve a very low false negative rate and an approximately 50% false positive rate using a modified AdaBoost algorithm. Stages are added to the cascade until the number of false positives is reduced below a small number on a validation set. Bi et al. [6] propose using 1-norm SVM as the stage classifiers in the cascade. Like Viola and Jones, their approach trains the stage classifiers sequentially from the first stage to the last stage. In every stage, an 1-norm SVM is trained to minimize the sum of the weighted errors and the regularization term.

There are two major differences between our problem formulation and cascade classification. First, although classification speed is important, we are mainly concerned about the costs of gathering information (i.e., the feature costs). Also, while cascade classification requires the user to choose the desired false negative rate and false positive rate at every stage, we assume that the feature costs and the misclassification costs are explicitly specified by the user and our algorithm automatically determines the best tradeoff between the feature cost and the two types of errors.

Second, we optimize the stage classifiers as a group to maximize the overall performance of the processing pipeline. The problem formulation allows the information available to change at each stage. Thus, false-positive and false-negative rates at each stage are not sufficient. It matters *which* positive examples are incorrectly rejected at a stage, not just *how many*. In particular, examples for whom further processing would still result in the incorrect classification should be rejected, while those for whom further information would clarify their classification should be saved.

Cascade classification and catSVM are both "staged" classifiers, but they are more complementary than competitive. The former attempts to speed up the computation of a single classification task (fixed information) by exploiting the asymmetric distribution of examples while the latter attempts to speed up a decision task by exploiting the correlation between data sources gathered at different times. One could well imagine using cascade classifiers at each stage within the framework shown here.

The prior algorithms that are closest to our work are two recent papers, [7] and [1]. Dundar and Bi [7] consider the problem of jointly optimizing cascaded SVM classifiers. However, they ignore the difference of rejecting an example at different stages. They formulate a non-convex and non-linear objective function and propose a cyclic optimization algorithm to optimize it. Shelton et al. [1] consider using boosting to build a classifier pipeline. They formulate a loss function in terms regular costs and propose a upper bound of the conjunction of indicator functions using a product of exponential functions. The resulting upper bound of the loss function is convex and easy to optimize, but it may be too loose to approximate the loss function well. We demonstrate evidence to this effect in our experimental results.

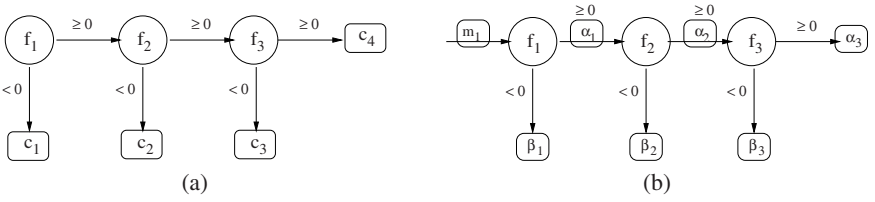


Fig. 1. An example of a three-stage processing pipeline

3 Sequential Stopping Problem

We follow the problem formulation in Shelton et al. [1]. There is no assumption about the structure of the costs. Rather, we assume that each training example carries a cost vector indicating the costs of stopping after each stage. These costs may increase, decrease, or have any other arbitrary relationship with the stage index. The costs might be function of a "label" or might be different for each example.

Let S be the number of stages in the processing pipeline. Denote the feature vector and the costs of an example by x and c, respectively. Let x_j be the components of x that are available at the j-th stage. Let c_j be the total cost of rejecting the example at the j-th stage and c_{S+1} be the total cost of accepting it (allowing it to "pass" at each decision). We assume that c is drawn from a known set $\mathcal{C}$. In the case of binary classification, $\mathcal{C}$ might be of cardinality 2: one sequence of costs for positive examples, and one sequence for negative examples. In general, $\mathcal{C}$ can be of any size. The only requirement is that the maximum magnitude of the members of $\mathcal{C}$ be bounded. Figure 1(a) shows an example of a three-stage processing pipeline.

Denote the classifier at the j-th stage by f_j and the entire processing pipeline by f. A positive value for f_j indicates that processing should continue, while a negative value indicates processing should stop. The loss for an example is therefore

$$L(f(x), c) = \sum_{j=1}^{S} \left(c_j I\left[f_j(x_j) < 0\right] \prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) + c_{S+1} \prod_{k=1}^{S} I\left[f_k(x_k) \geq 0\right] \ . \tag{1}$$

The goal is to find S classifiers, one for each stage, which together minimize $\mathbf{E}[L(f(x), c)]$. Although we do not know the true distribution of (x, c), we can use the empirical loss as a surrogate. We let $\{(X_1, C_1), \ldots, (X_N, C_N)\}$ denote the training set and X_{ij} denote the features of X_i that are available at the j-th stage. Analogously, we let C_{ij} denote the cost associated with X_i at the j-th stage.

4 Catenary Support Vector Machines

We return to the formulation in Section 3 and derive an optimization procedure based on a loss bound.

4.1 Loss Bound

We start by re-writing the loss function (1) in terms of incremental costs.

$$L(f(x), c) = m_1 + \sum_{j=1}^{S} \left(\prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) \Big(\alpha_j I\left[f_j(x_j) \geq 0\right] + \beta_j I\left[f_j(x_j) < 0\right] \Big) \quad (2)$$

where for $j = 1, \ldots, S$,

$$m_j = \begin{cases} \min\left(m_{j+1}, c_j\right) & \text{if } j < S, \\ \min\left(c_{j+1}, c_j\right) & \text{if } j = S; \end{cases}$$

$$\alpha_j = \begin{cases} m_{j+1} - m_j & \text{if } j < S, \\ c_{j+1} - m_j & \text{if } j = S; \end{cases}$$

$$\beta_j = c_j - m_j \quad .$$

In words, m_j is the minimal cost at stage j or later. α_j is the incremental increase in the minimal cost by continuing processing and β_j is the incremental cost of stopping processing. Note that either α_j or β_j is positive but not both. We denote the incremental costs associated with X_i at the j-th stage by m_{ij}, α_{ij}, and β_{ij}. Figure 1(b) shows a three-stage processing pipeline with the incremental costs.

It is hard to minimize (2) directly. We define a upper bound for $L(f(x), c)$ and minimize the upper bound instead.

$$\hat{L}(f(x), c) = m_1 + \sum_{j=1}^{S} \left[\alpha_j \Big(U_j^{\alpha}(x) - V_j^{\alpha}(x) \Big) + \beta_j \Big(U_j^{\beta}(x) - V_j^{\beta}(x) \Big) \right] \quad (3)$$

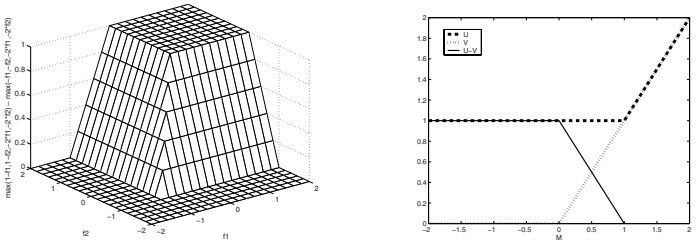
where

$$\begin{aligned} U_j^*(x_j) &= \max\left(1, M_j^*(x)\right) \\ V_j^*(x_j) &= \max\left(0, M_j^*(x)\right) \\ M_j^{\alpha}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), -f_j(x_j)\right) \\ M_j^{\beta}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), f_j(x_j)\right) \quad . \end{aligned}$$

The wildcard '*' represents either α or β. The key idea of deriving Equation (3) is to use the difference of two $\max$ functions to upper bound the conjunction of indicator functions. Figure 2(a) shows an example when the conjunction consists of two indicator functions. Note that we simply bound the 0-1 function by the ramp function. Similar ramp loss functions have been used before to approximate the classification error more closely [8,9] and to improve the scalability of SVMs [10]. Figure 2(b) shows U_j^* and V_j^* as a function M_j^*.

We formulate the following optimization problem.

$$\min \sum_{i=1}^{N} \hat{L}(f(X_i), C_i) + \lambda \Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) \quad (4)$$



(a) (b)

Fig. 2. (a) Upper bound of $I[f_1 > 0] \cdot I[f_2 > 0]$, (b) U_j^* and V_j^* as a function M_j^*

where Ω is some monotonically increasing function. The first term measures the empirical loss and the second term is the regularization term, measured with respect to a set of reproducing kernel Hilbert spaces $\{\mathcal{H}_j\}$.

4.2 Catenary Support Vector Optimization

We begin with linear classifiers $f_j(x_j) = w_j \cdot x_j + b_j$ and an ℓ_2 regularization term $\Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) = \sum_{j=1}^{S} \|w_j\|_2^2$. Note that U_j^* and V_j^* are all convex functions in $\{(w_1, b_1), \ldots, (w_S, b_S)\}$. But the difference of two convex functions, $U_j^*(x_j) - V_j^*(x_j)$, is non-convex. Thus, Problem (4) is not a convex optimization problem.

We re-formulate Problem (4) as the following constrained optimization problem.

$$\begin{array}{ll} \min & \sum_{i=1}^{N} \sum_{j=1}^{S} \left(\alpha_{ij} \xi_{ij}^{\alpha} + \beta_{ij} \xi_{ij}^{\beta} \right) + \lambda \sum_{j=1}^{S} \|w_j\|_2^2 \\ & \xi_{ij}^{\alpha} \geq -w_k \cdot X_{ij} - b_k - V_j^{\alpha}(X_i) \ \forall i, k \leq j \\ & \xi_{ij}^{\beta} \geq -w_k \cdot X_{ij} - b_k - V_j^{\beta}(X_i) \ \forall i, k < j \\ \text{s.t.} & \xi_{ij}^{\beta} \geq w_j \cdot X_{ij} + b_j - V_j^{\beta}(X_i) \quad \forall i, j \\ & \xi_{ij}^{\alpha}, \xi_{ij}^{\beta} \geq 1 \qquad\qquad\qquad\quad \forall i, j \end{array} \tag{5}$$

Note that we have dropped the constant term, $\sum_{i=1}^{N} m_{i,1}$, from the objective. In principle, we can leave V_j^* in the objective. But moving V_j^* to the constraints appears to give better empirical results. Also, it is possible to have different tradeoff parameters for each classifier stage.

The Concave-Convex Procedure. The constraints of Program (5) can all be viewed as the difference of two convex functions. We employ the concave-convex procedure (CCCP), first introduced by Yuille and Rangarajan [11] to solve minimization problems whose objective function can be expressed as the sum of a convex part and a concave part. While Yuille and Rangarajan [11] considered only linear constraints, Smola et al. [12] generalized the CCCP to handle concave-convex constraints. The CCCP is an iterative procedure. In each iteration, it replaces the concave parts in the objective function and the constraints by their first-order Taylor approximation. The resulting problem is convex and can be solved using efficient convex minimization algorithms.

Consider the following optimization problem:

$$\begin{aligned} &\min\ f_0(x) - g_0(x) \\ &\text{s.t.}\quad f_i(x) - g_i(x) \leq c_i \quad \forall i \end{aligned}$$

where f_i and g_i are real-valued convex and differentiable functions on $\Re^n$ for $i \in \{0, \ldots, m\}$, and $c_i \in \Re$ for $i \in \{1, \ldots, m\}$. The CCCP computes $x^{(t+1)}$ from $x^{(t)}$ by solving the following convex optimization problem.

$$\begin{aligned} &\min f_0(x) - \left(g_0(x^{(t)}) + \nabla g_0(x^{(t)})^T (x - x^{(t)})\right) \\ &\text{s.t.}\ \ f_i(x) - \left(g_i(x^{(t)}) + \nabla g_i(x^{(t)})^T (x - x^{(t)})\right) \leq c_i \quad \forall i \end{aligned}$$

It can be shown that the CCCP converges to a local minimum [12]. In case of non-global minimum, one may restart the CCCP with a different $x^{(0)}$. However, the CCCP can be considered as a special case of difference of convex functions (D.C.) programming. Tao and An [13] state that the D.C. minimization algorithm (DCA) often converges to a global minimum.

catSVM Program. To formulate Program (5) as a CCCP problem, let $\mathbf{w} = (w_1, \ldots, w_S)$ and $\mathbf{b} = (b_1, \ldots, b_S)$. In each iteration, we need to replace V_j^* in the constraints by its first-order Taylor expansion at the current estimates of $\mathbf{w}$ and $\mathbf{b}$. Notice that V_j^* are non-smooth functions. When we calculate its Taylor expansion, we use its subgradient. For the pointwise maximum function $h(x) = \max_{1 \leq i \leq m} h_i(x)$, its subdifferential at x, $\partial h(x)$, is the convex hull of the subdifferentials of the "active" functions at x, i.e., $\partial h(x) = \mathbf{H}_{\mathbf{Convex}}\{\partial h_i(x) | h_i(x) = h(x)\}$. Thus, by simple calculus, we obtain that, for $j = 1, \ldots, S$,

$$\frac{\partial V_j^*(x; \mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = \begin{cases} \{\mathbf{0}\} & \text{if } M_j^*(x) < 0, \\ \left\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma \tau_j x_j, \mathbf{0}) \;\middle|\; \tau_k \geq 0,\ \sum_{k=1}^{j} \tau_k \leq 1\right\} & \text{if } M_j^*(x) = 0, \\ \left\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma \tau_j x_j, \mathbf{0}) \;\middle|\; \tau_k \geq 0,\ \sum_{k=1}^{j} \tau_k = 1\right\} & \text{if } M_j^*(x) > 0; \end{cases} \tag{6}$$

where

$$\tau_k = 0 \quad \begin{array}{c} \text{if } k < j \text{ and } M_j^*(x) \neq -w_k \cdot x_k - b_k \\ \text{or if } k = j \text{ and } M_j^*(x) \neq \sigma(w_k \cdot x_k + b_k) \end{array}$$

$$\sigma = \begin{cases} -1 & \text{if } *= \alpha, \\ +1 & \text{if } *= \beta. \end{cases}$$

and $\mathbf{0}$ denotes padding zeroes of appropriate length.

Similarly, we can obtain $\frac{\partial V_j*(x; \mathbf{w}, \mathbf{b})}{\partial \mathbf{b}}$ by replacing x_k's by 1 in Equation (6). In the experiments, we pick the subgradient with

$$\tau_k = \begin{cases} c & \text{if } k \text{ is the largest index s.t.} \\ & \quad \text{either } k < j \text{ and } M_j^*(x) = -w_k \cdot x_k - b_k, \\ & \quad \text{or } k = j \text{ and } M_j^*(x) = \sigma(w_k \cdot x_k + b_k), \\ 0 & \text{otherwise,} \end{cases}$$

where $c = \frac{\rho}{\rho+1}$ and ρ is the number of active functions if $M_j^*(x) = 0$, and $c = 1$ if $M_j^*(x) > 0$.

Since only one of α_{ij} or β_{ij} is nonzero, we need only consider the constraints associated with one of ξ_{ij}^α or ξ_{ij}^β. The number of constraints in Program (5) is quadratic in the number of stages. We can re-write it so that the number of constraints depends linearly on the number of stages.

$$\begin{array}{ll} \min & \sum_{i=1}^N \sum_{j=1}^S \left(\alpha_{ij}\xi_{ij}^\alpha + \beta_{ij}\xi_{ij}^\beta\right) + \lambda \sum_{j=1}^S \|w_j\|_2^2 \\ & \xi_{ij}^\alpha \geq \eta_{ij} - V_j^\alpha(X_i) \qquad \forall i,j \\ & \xi_{ij}^\beta \geq \eta_{i,j-1} - V_j^\beta(X_i) \qquad \forall i,j \\ \text{s.t.} & \xi_{ij}^\beta \geq w_j \cdot X_{ij} + b_j - V_j^\beta(X_i) \; \forall i,j \\ & \xi_{ij}^\alpha, \xi_{ij}^\beta \geq 1 \qquad \forall i,j \\ & \eta_{ij} \geq -w_j \cdot X_{ij} - b_j \qquad \forall i,j \\ & \eta_{ij} \geq \eta_{i,j-1} \qquad \forall i,j \end{array} \tag{7}$$

4.3 Extensions

One important advantage of SVM is that it can use kernels to handle data that are not linearly separable. By the Representer Theorem [14], we can also kernelize the stage classifiers. Denote the kernel matrix for the j-th stage by K_j and its i-th column by $K_j(\cdot, i)$. The only changes to Program (7) are (i) replacing the regularization term in the objective by $\lambda \sum_j^S w_j' K_j w_j$, and (ii) replacing the feature vector X_{ij} by $K_j(\cdot, i)$. We are free to use different kernels for different stages. Also, instead of using a ℓ_2 regularization term, we may also use ℓ_1 regularization term to promote sparsity. If we do so, in each iteration of the CCCP, we need to solve a linear program instead of a quadratic program.

4.4 An Alternative Loss Bound

In the above derivation, we view the loss of an example as the sum of losses incurred in each stage and then derive a upper bound using ramp functions. This is not the only way to do it. Alternatively, we can view the loss of an example as the max of losses incurred in each stage and obtain the following loss bound:

$$\tilde{L}(f(x), c) = d_0 + \max\left(\left\{d_j(U_j^\alpha(x) - V_j^\alpha(x))\right\}_{j=1}^S, d_{S+1}(U_{S+1}^\beta(x) - V_{S+1}^\beta(x))\right) \tag{8}$$

where $d_0 = \min(c_1, \ldots, c_{S+1})$ and $d_j = c_j - d_0$.

Note that $\tilde{L}$ is no greater than $\hat{L}$. It is not difficult to see that we can formulate a constrained optimization problem with $\hat{L}$ and employ the CCCP to solve it. Unfortunately, our preliminary experimental results showed that the CCCP is not effective in optimizing $\tilde{L}$. We leave the problem of optimizing $\tilde{L}$ as an open problem.

5 Performance Bounds

We are able to provide theoretical bounds on how well catSVM will perform on new testing data. In fact, Shelton et al. [1] gave a risk bound for chained boosting and a similar bound holds for catSVM. We will state the theorem below. The proof is similar to the one in [1] and is omitted. We need the following definition.

Definition 1. *Let μ be a probability distribution on a set $\mathcal{X}$ and suppose that $X_1, \ldots, X_n$ are independent samples selected according to μ. Let F be a class of functions mapping from $\mathcal{X}$ to $\Re$. Define the random variable*

$$\hat{G}_n(F) = \mathbf{E}\left[\sup_{f \in F} \left| \frac{2}{n} \sum_{1}^{n} g_i f(X_i) \right| \;\middle|\; X_1, \ldots, X_n \right] ,$$

where $g_1, \ldots, g_n$ are independent Gaussian $N(0,1)$ random variables. The Gaussian complexity of F is $G_n(F) = \mathbf{E}\hat{G}_n(F)$.

We can now state the theorem, bounding the true risk by the empirical risk and the Gaussian complexity of the classes of the stage classifiers:

Theorem 1. *Let L and $\hat{L}$ be as in Equations (1) and (3). Let $\gamma_j = \max_{c \in \mathcal{C}}(\alpha_j + \beta_j)$ and $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. Let $F_1, \ldots, F_S$ be the sequence of the classes of the stage classifiers. Let $(X_i, C_i)_{i=1}^N$ be independently selected according to some fixed probability measure P. Then, for any integer N and any $0 < \delta < 1$, with probability at least $1 - \delta$ over samples of size N, every sequence $f_1, \ldots, f_S$ in $F_1 \times \ldots \times F_S$ satisfies*

$$\mathbf{E}[L] \leq \hat{\mathbf{E}}_N[\hat{L}] + \kappa \sum_{j=1}^{S} \left(\sum_{\ell=j}^{S} \gamma_\ell \right) G_N(F_j) + \Lambda \sqrt{\frac{8 \ln \frac{2}{\delta}}{N}}$$

for some constant κ.

Note that the second term in the bound does not depend on the regular costs c_j's directly, and it does not depend on m_1 at all. Quite often, the incremental costs α_j's and β_j's are smaller than the c_j's. Additionally, for $j < j'$, the complexity of F_j has a larger weight than that of $F_{j'}$. This may suggest that it is advantageous to use simple stage classifiers in early stages and use complex stage classifiers in later stages. We can further bound the true risk in terms of kernel functions of the stage classifiers. We need the following lemma which follows from McDiarmid's inequality [15].

Lemma 1. *Let F be a class of functions mapping to $[-1, 1]$. For any integer n,*

$$P\left\{ |G_n(F) - \hat{G}_n(F)| \geq \epsilon \right\} \leq 2 \exp\left(\frac{-n\pi\epsilon^2}{4} \right) .$$

Theorem 1 and Lemma 1, combined with Lemma 22 in Bartlett and Mendelson [16], imply the following theorem.

Theorem 2. *Let L and $\hat{L}$ as in Equations (1) and (3). Let $\gamma_j = \max_{c\in\mathcal{C}}(\alpha_j + \beta_j)$ and $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. Let $F_1, \ldots, F_S$ be the sequence of the classes of stage classifiers. Let $\mathcal{X}_j$ be the feature space in the j-th stage. For $j = 1, \ldots, S$, fix B_j, and let $K_j : \mathcal{X}_j \times \mathcal{X}_j \to \Re$ be a kernel with $\sup_{x\in\mathcal{X}_j} |K_j(x, x)| < \infty$. Let $\mathcal{X}$ be the full feature space, i.e., $\mathcal{X} = \bigcup_{j=1}^{S} \mathcal{X}_j$. Suppose that $\{X_i, C_i\}_{i=1}^{N}$ are selected at random and independently according to some probability distribution P on $\mathcal{X} \times \mathcal{C}$. Then with probability at least $1 - \delta$, every function sequence $f_1, \ldots, f_S$ of the form*

$$f_j(x) = \sum_{i=1}^{N} \alpha_i K_j(x_{ij}, x)$$

with $\sum_{i_1,i_2} \alpha_{i_1}\alpha_{i_2} K_j(x_{i_1,j}, x_{i_2,j}) \le B_j^2$ satisfies

$$\mathbf{E}[L] \le \hat{\mathbf{E}}_N[\hat{L}] + \frac{\kappa}{N} \sum_{j=1}^{S} \left(\sum_{\ell=j}^{S} \gamma_\ell \right) B_j \sqrt{\sum_{i=1}^{N} K_j(x_{ij}, x_{ij})} + \left(\Lambda + \frac{1}{\sqrt{2\pi}} \sum_{j=1}^{S} j\gamma_j \right) \sqrt{\frac{8 \ln \frac{2(S+1)}{\delta}}{N}}$$

for some constant κ.

6 Experimental Results

We tested catSVM on UCI benchmark data and the MIT face database. We compared the performance of catSVM to chained boosting[1] and single-stage SVM. For chained boosting, we used decision stumps as weak classifiers and set the number of rounds before the algorithm stops to 2000. For single-stage SVM and catSVM, we used the RBF kernel and set the kernel width to the median of the pairwise distance in the training set. We set the regularization parameter λ to be 1 and did not adjust it. Furthermore, for catSVM, we initialized the SVM in every stage to be zero (i.e., for $j = 1, \ldots, S$, $w_j = \mathbf{0}$ and $b_j = 0$). We used Mosek to solve the quadratic programs generated by catSVM.

The single-stage SVM and our catSVM algorithms run on the same hypothesis space of RBF kernels. We ran chained boosting on a different feature space. It was not possible to use the same feature space. The boosting algorithm constructs a linear surface in the space of features that are decision stumps. That does not correspond to any easily constructed kernel. The feature space dimensions of the RBF kernel could be defined as the set of all kernel functions with one point as a training point. However, those dimensions have real values and the boosting algorithm is designed for thresholded features (or weak learners). However, our experience with these datasets suggests that boosting decision stumps and RBF kernels for SVMs have roughly the same performance on single-stage problems.

We did not compare to constructing three independent SVM classifiers and then connecting them in a chain; this would have required selecting the false-positive and

[1] The original chained boosting algorithm uses regular costs. We modified it to use incremental costs as well, which improved its performance.

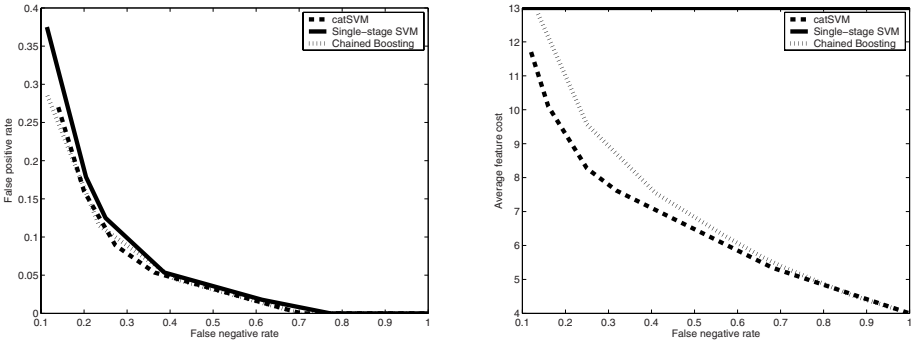


Fig. 3. UCI heart tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

Table 1. For four different cost settings for the heart dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples

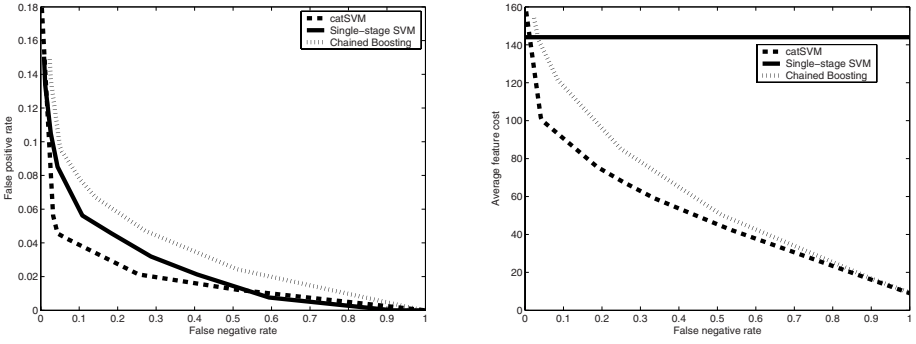
		fn: 9, fp: 18				fn: 18, fp: 18				fn: 36, fp: 18				fn: 72, fp: 18			
boosting	train +	76	0	0	0	70	0	0	6	38	0	0	38	16	0	0	60
	train –	94	0	0	0	94	0	0	0	82	8	4	0	74	4	16	0
	test +	43	0	0	1	39	0	0	5	24	2	0	18	18	2	1	23
	test –	56	0	0	0	52	1	2	1	45	5	4	2	42	7	4	3
catSVM	train +	76	0	0	0	6	2	6	62	2	1	1	72	0	0	2	74
	train –	94	0	0	0	68	3	19	4	54	8	20	12	25	20	34	15
	test +	44	0	0	0	9	0	5	30	7	1	3	33	7	0	4	33
	test –	56	0	0	0	49	1	5	6	35	5	9	7	20	9	19	8

false-negative costs for each classifier. One of the main purposes of our approach is to automatically adjust the classifier to achieve the desired cost results without having to manually search over such trade-off parameters.

We construct the vectors of stage costs as follows. We assign a constant feature cost to each stage that is the same for all examples (as specified in the problem set up below). It represents the cost of collecting the features, regardless of the final outcome. If the example is positive, we add an extra cost to c_i for $i \leq s$, representing an extra penalty if a positive example is rejected at any stage. If the example is negative, we add an extra cost to c_{s+1}, that is we penalize the classifier if it allows the example to pass through every stage (and therefore wrongly accepts it as a positive example).

6.1 UCI Data

We report the results on the heart disease dataset from the UCI machine learning repository. We assigned the 13 attributes to three stages in descending order according to



(a) (b)

Fig. 4. Face detection tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

Table 2. For four different cost settings for the face detection dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples

		fn: 150, fp: 250				fn: 250, fp: 250				fn: 500, fp: 250				fn: 1000, fp: 250			
boosting	train +	224	0	0	0	224	0	0	0	220	0	0	4	33	0	0	191
	train –	376	0	0	0	376	0	0	0	375	1	0	0	342	34	0	0
	test +	341	0	0	1	341	0	0	1	342	4	0	36	143	32	1	166
	test –	654	1	3	0	653	1	3	1	646	4	4	4	533	88	19	18
catSVM	train +	224	0	0	0	35	8	0	181	0	0	1	223	0	0	0	224
	train –	376	0	0	0	278	86	10	2	104	233	36	3	96	240	37	3
	test +	342	0	0	0	42	16	6	278	1	10	7	324	1	10	7	324
	test –	658	0	0	0	457	143	28	30	170	392	63	33	152	410	62	34

their correlation with the predicted output: four attributes to each of the first two stages and five to the last stage. The single-stage SVM was trained and tested on all the 13 attributes. We set the feature cost of each stage to the number of attributes assigned to that stage and all the preceding stages. Early rejections are treated as "normal" whereas an example that passes all stages is treated as "disease." Figure 3(a) shows the false negatives and false positives as the misclassification penalties vary. The three methods give very similar tradeoff between the two types of errors. Figure 3(b) shows the false negative and the average feature cost as the misclassification penalties vary. The average feature cost is the average number of features that must be examined before the classifier makes a decision. The feature cost of single-stage SVM is fixed at 13. We observe that catSVM does a better job in trading false negative rate for feature cost than chained boosting.

Table 1 shows at which stage the examples are rejected or accepted. The feature costs are 4, 8, 13, and 13. 'fn: 9, fp: 18' means the penalties for false negative and false positive are 9 and 18, respectively; therefore the cost vectors would be $[13, 17, 22, 13]$

and $[4, 8, 13, 31]$ for positive and negative examples, respectively. Note that for every penalty setting, the first three columns are the number of examples rejected in the three stages and the last column is the number of examples accepted. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples.

6.2 Face Detection

We also validated the catenary SVM by applying it to face detection. We tested on the MIT face database [17] which contains 19-by-19 gray-scale images of faces and non-faces. For face detection, the non-face is usually the majority. Therefore, our goal is to produce a classifier that can identify non-face images by examining as low a resolution patch as possible. We built a multi-stage detection system where any early rejection is labeled as a non-face. The first stage looks at down-sampled versions of the images at a resolution of 3-by-3. The next stages do the same, at resolutions of 6-by-6 and 12-by-12. We did not examine the full 19-by-19 resolution as it did not provide significant improvement over the 12-by-12 resolution.

We assign a feature cost to each stage proportional to the total number of pixels at that stage and all the preceding stages. There are three free parameters in the problem formulation: the per pixel cost, the penalty for an incorrect face classification, and the penalty for an incorrect non-face classification. Changing these quantities will control the tradeoff between false negatives and false positives, and between classification error and feature cost. In the experiments, we fix the per pixel cost and vary the other two quantities.

We used 600 images as the training set and 1000 images as the testing set. The single-stage SVM was trained and tested on image patches at the highest resolution, 12-by-12. Figure 4(a) shows the false negatives and false positives as the misclassification penalties vary. Note that catSVM can achieve better tradeoff than single-stage SVM and chained boosting. The processing pipeline successfully improves the ability of SVM to tradeoff between the two types of errors. Figure 4(b) shows the false negative and the average feature cost as the misclassification penalties vary. The feature cost of single-stage SVM is fixed at 144. Chained boosting and catSVM give higher average feature costs for lower false negative rates. Note that catSVM requires a lower average feature cost than chained boosting for most false negative rates. The advantage of catSVM becomes more obvious when the false negative rate is small.

Table 2 shows at which stage the examples are rejected or accepted. The feature costs are 9, 45, 189, and 189. 'fn: 150, fp: 250' means the penalties for false negative and false positive are 150 and 250, respectively. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples. It is clear that catSVM is more effective in pushing the positive examples forward.

It is interesting to note that the performance of catSVM is superior to that of a single-stage SVM (which is a regular SVM trained on the full set of features, varying the false-positive and false-negative costs) in terms of testing error. We believe this is because the hypothesis class of the earlier stages are simpler. Therefore, those decision rules have less variance for a fixed number of samples. Our algorithm then has a natural bias that

helps reduce the variance with few numbers of samples. Our generalization bounds also point to this advantage.

7 Conclusion

We believe that for some decision-making problems, it is important to weigh the benefit against the cost of acquiring more information. We present the catenary SVM to solve one-sided early detection for binary classification. We formulate the problem as a constrained concave-convex optimization problem and solve it using CCCP. In addition, we are able to provide data-dependent theoretical guarantee for catSVM. The experimental results show that catSVM can tradeoff misclassification error and feature cost more effectively than single-stage SVM and chained boosting.

The main drawback of catSVM is its scalability. Currently, we use a generic solver to solve the linear or quadratic programs generated by CCCP. Although the number of constraints is only linear in the number of stages, the generic solver runs out of memory even for medium-size datasets. We are planning to explore other more scalable algorithms (e.g., cutting-plane methods). Moreover, it would be interesting to extend catSVM to two-sided early detection.

Acknowledgments

This research was supported through the grant "Adaptive Decision Making for Silicon Wafer Testing" from Intel Research and UC MICRO. We thank the anonymous reviewers for their careful and thoughtful comments.

References

1. Shelton, C.R., Huie, W., Kan, K.F.: Chained boosting. In: NIPS 2007, pp. 1281–1288 (2007)
2. Howard, R.A., Matheson, J.E.: Influence diagrams. In: Howard, R.A., Matheson, J.E. (eds.) Readings on the Principles and Applications of Decision Analysis. Strategic Decision Group, vol. 2, pp. 719–762 (1984) article dated 1981
3. Viola, P., Jones, M.: Fast and robust classification using asymmetric adaboost and a detector cascade. In: NIPS, pp. 1311–1318 (2002)
4. Wu, J., Mullin, M.D., Rehg, J.M.: Linear asymmetric classifier for cascade detectors. In: ICML, pp. 988–995 (2005)
5. Šochman, J., Matas, J.: WaldBoost — learning for time constrained sequential detection. In: CVPR, pp. 150–156 (2005)
6. Bi, J., Periaswamy, S., Okada, K., Kubota, T., Fung, G., Salganicoff, M., Rao, B.: Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. In: SIGKDD, pp. 837–844 (2006)
7. Dundar, M., Bi, J.: Joint optimization of cascaded classifiers for computer aided detection. In: CVPR, pp. 1–8 (2007)
8. Shen, X., Tseng, G.C., Zhang, X., Wong, W.H.: On ψ-learning. Journal of the American Statistical Association 98(463), 724–734 (2003)
9. Liu, Y., Shen, X., Doss, H.: Multicategory ψ-learning and support vector machine: computational tools. J. of Comp. and Graphical Statistics 14(1), 219–236 (2005)

10. Collobert, R., Weston, J., Bottou, L.: Trading convexity for scalability. In: ICML, pp. 201–208 (2006)
11. Yuille, A.L., Rangarajan, A.: The concave-convex procedure (CCCP). In: NIPS, pp. 1033–1040 (2002)
12. Smola, A.J., Vishwanathan, S., Hofmann, T.: Kernel methods for missing variables. In: AI-STATS, pp. 325–332 (2005)
13. Tao, P.D., An, L.T.: A D.C. optimization algorithm for solving the trust-region subproblem. SIAM Journal on Optimization 8(2), 476–505 (1998)
14. Schölkopf, B., Smola, A.J.: Learning with kernels. The MIT Press, Cambridge (2002)
15. McDiarmid, C.: On the method of bounded differences. Surveys in Combinatorics, 148–188 (1989)
16. Bartlett, P.L., Mendelson, S.: Rademacher and Gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research 2, 463–482 (2002)
17. MIT: CBCL face database #1 (2000), `http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html`

Exact and Approximate Inference for Annotating Graphs with Structural SVMs

Thoralf Klein[1], Ulf Brefeld[2], and Tobias Scheffer[1]

[1] Max Planck Institute for Computer Science, Saarbrücken, Germany
{tklein,scheffer}@mpi-inf.mpg.de
[2] Machine Learning Group, Technische Universität Berlin, Germany
brefeld@cs.tu-berlin.de

Abstract. Training processes of structured prediction models such as structural SVMs involve frequent computations of the *maximum-a-posteriori* (MAP) prediction given a parameterized model. For specific output structures such as sequences or trees, MAP estimates can be computed efficiently by dynamic programming algorithms such as the Viterbi algorithm and the CKY parser. However, when the output structures can be arbitrary graphs, exact calculation of the MAP estimate is an NP-complete problem. In this paper, we compare exact inference and approximate inference for labeling graphs. We study the exact junction tree and the approximate loopy belief propagation and sampling algorithms in terms of performance and ressource requirements.

1 Introduction

Many problem settings which require the prediction of multiple dependent variables arise naturally. For instance, sequential input and output variables occur in protein secondary structure prediction and tree-structured output is produced in natural language parsing. Examples for general graphical output structures include classification of linked documents or webpages [11,8,10], and simultaneous prediction of multiple dependent class labels [3].

Many classical learning algorithms have been lifted to deal with structured variables. Generally, the learning task is phrased as finding a function f such that

$$\hat{y} = \operatorname*{argmax}_y f(x, y) \tag{1}$$

is the desired output for a given input x. Conditional random fields [6] and structural support vector machines [13,12] are parameter estimation techniques that are wrapped around the collective inference machinery. However, inferring the actual prediction acts as a bottleneck in the training process. Dynamic programming approaches such as the Viterbi algorithm and the CKY algorithm efficiently solve the inference problem for sequential and tree-structured output variables.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 611–623, 2008.

When the structures involved can be arbitrary graphs, exact inference is not tractable: For instance, there is no efficient analytic solution for discrete variables and dynamic programming for exact inference scales exponentially in the size of the largest clique of the graph in terms of memory and computation time requirements. Hence, practitioners usually resort to approximate inference techniques. For instance, loopy belief propagation and Gibbs sampling are appealing and intuitive approaches that lead to efficient algorithms which alleviate excessive computational requirements. Although these approximate variants are not guaranteed to converge – let alone to find good optima – they seem to be surprisingly well-suited for many practical applications [7,10,3].

In this paper, we present a substantial comparison of exact and approximate inference techniques for their use with structural support vector machines. We address their implications on the SVM algorithm in terms of convergence, resource requirements, and performance. Our experiments reveal that exact inference is indispensable for the reliable learning of accurate prediction models. According to our findings, a remedy to the computational costs of the junction tree algorithm is to decompose large graphs into smaller subgraphs.

Our paper is structured as follows. We introduce the learning task formally in Section 2 and review structural support vector machines in Section 3. Section 4 addresses exact and approximate inference algorithms and Section 5 reports on our empirical results. We discuss our findings in Section 6 and Section 7 concludes.

2 Learning with Graphs

Let $G = (V, E)$ be a graph such that the set of nodes decomposes into a set of observed nodes X and a set of latent nodes Y the labeling of which remains to be conjectured. The set of edges $E \subseteq (X \times Y) \cup (Y \times Y)$ introduces a dependency structure between nodes $V = X \cup Y$. That is, two variables are connected if they directly depend on each other. A *structured input* x is a manifestation of a graph together with a labeling of its observed nodes. A *structured output* is a labeling $y \in \Sigma^{|Y|}$, where Σ denotes the label alphabet.

The random variables $X \cup Y$ form a Markov random field with dependency structure E. The conditional probability of Y given a partial realization (observation) x can be written as

$$\hat{p}(y|x) = \exp\{\langle \lambda, \Phi(x, y)\rangle - \log g(\lambda|x)\}, \tag{2}$$

where $g(\lambda|x) = \sum_{\bar{y}} \exp\{\langle \lambda, \Phi(x, \bar{y})\} < \infty$ is called the partition function, Φ is the sufficient statistics, and λ denotes the natural parameter. The sufficient statistics factorizes according to the dependency structure into terms of the maximal cliques $C \in \mathcal{C}$ of the graph,

$$\Phi(x, y) = \sum_{C \in \mathcal{C}} \phi_C(x_C, y_C).$$

Functions ϕ_C may be interpreted as joint feature vectors, extracted from clique C and therefore encode the dependency structure to allow the model to learn about dependencies between input and output variables.

When dealing with arbitrary graphs, cliques may grow arbitrarily large and employing feature functions for all possible cliques can become intractable. As a remedy, one usually resorts to factorizing over all cliques that contain a pair of nodes, rather than over all maximally large cliques. In this case, G is sometimes called a Markov network and every edge in G is associated with a feature function ϕ. That is, we have edges between label and observation pairs,

$$\phi_1(x_i, y_i) = (\delta_{k_1,y_i}, \ldots, \delta_{k_{|\Sigma|},y_i})^{\mathsf{T}} \otimes \psi(x_i),$$

and between neighboring labels,

$$\phi_2(y_i, y_j) = \begin{pmatrix} \delta_{k_1,y_i} \\ \vdots \\ \delta_{k_{|\Sigma|},y_i} \end{pmatrix} \otimes \begin{pmatrix} \delta_{k_1,y_j} \\ \vdots \\ \delta_{k_{|\Sigma|},y_j} \end{pmatrix},$$

where ψ is a feature vector solely drawn from the observation, $k \in \Sigma$ enumerates all possible labels, $\delta_{i,j}$ is the Kronecker product and $\otimes$ denotes the tensor product. Equation 2 says that given parameters λ, the most likely labeling $\hat{y}$ can be computed using the generalized linear model

$$\hat{y} = \operatorname{argmax}_{\bar{y}} \hat{p}(\bar{y}|x) = \operatorname{argmax}_{\bar{y}} f(x, \bar{y}),$$

with $f(x, y) = \langle \lambda, \Phi(x, y) \rangle$. We thus seek to find a parameterization λ, such that the model f generalizes well on new and unseen graphs.

The quality of f is measured by a task-dependent loss function $\Delta : (y, y') \mapsto r \in \mathbb{R}_0^+$. We require that Δ is decomposable in terms of the nodes of graph, for instance, Δ may be a Hamming-like loss given by

$$\Delta(y, y') = \sum_{j=1}^{|y|} [[y_j \neq y'_j]]. \tag{3}$$

The ultimate goal is to find f such as to minimize the true loss $\sum_y \int \Delta(y, \operatorname{argmax}_{\bar{y}} f(x, \bar{y})) p(x, y) dx$. Being ignorant of the true distribution $p(x, y)$ of instances and labelings of the unobserved variables, one resorts to declaring minimization of the regularized empirical loss on an iid sample $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{\ell}$ to be the operational goal

$$\hat{R}[f] = \sum_{i=1}^{\ell} \Delta(y^{(i)}, \operatorname*{argmax}_{\bar{y}} f(x^{(i)}, \bar{y})) + \frac{1}{\eta} \|f\|^2.$$

3 Structural Support Vector Machines

Structural SVMs [12,13] adapt the parameters of ranking functions that are defined over joint attributes of input and output:

$$f(x, y) = \langle \lambda, \Phi(x, y) \rangle. \tag{4}$$

Given training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{\ell}$, SVMs aim at finding λ that minimizes $|\lambda|^2 + \sum_i \xi_i$ subject to the constraint that, for all $(x^{(i)}, y^{(i)})$, the desired output $y^{(i)}$ exceeds any other output $\bar{y}$ in its decision function values by at least $1 - \xi_i$:

$$\forall_{i=1}^{\ell} \quad f(x^{(i)}, y^{(i)}) - \max_{\bar{y} \neq y^{(i)}} f(x^{(i)}, \bar{y}) \geq 1 - \xi_i.$$

Optimization Problem 1 is solved iteratively by column generation: If at least one output violates the margin constraint for a given x, then the output $\bar{y}$ that violates it most strongly is inferred and added to a working set of explicitly represented constraints. Column generation is interleaved with optimization steps in which the current hypothesis λ is refined according to the current working set. Typically only a small fraction of conceivable constraints are represented in the working set and structural SVMs provide sparse solutions.

Optimization Problem 1. *Given data $\mathcal{D}$, loss function Δ, and $\eta > 0$, the structural SVM optimization problem is defined as:*

$$\min_{\lambda, \xi \geq \mathbf{0}} \frac{|\lambda|^2}{2} + \eta \langle \xi, \ \rangle$$

subject to the constraints

$$\langle \lambda, \Phi(x^{(i)}, y^{(i)}) \rangle \geq \max_{\bar{y} \neq y^{(i)}} [\Delta(y^{(i)}, \bar{y}) + \langle \lambda, \Phi(x^{(i)}, \bar{y}) \rangle] - \xi_i$$

for all $i = 1, \ldots, \ell$.

Provided that exact inference of large margin violators runs in polynomial time, the solution to Optimization Problem 1 converges to the optimum in polynomial time [13].

4 Inference Strategies

Intuitively, Equation 1 can be solved by explicitly computing the score for all possible assignments of the output variables and choosing the output $\hat{y}$ that realizes the highest score. However, there are exponentially many different assignments in the size of the graph and explicit enumeration is prohibitive. In this section, we briefly review the exact junction tree algorithm, the approximate loopy belief propagation and an inference strategy based on Gibbs sampling.

4.1 Junction Tree Algorithm

Belief propagation [1] sends messages across edges of the graph that are used to update actual beliefs about labelings. Belief propagation terminates and produces the exact joint probability of all unobserved variables when the graph is free of cycles. Therefore, one transforms the graph into a junction tree $J = (\mathcal{C}, E_J)$; nodes in the junction tree correspond to cliques in the underlying graph.

Message propagation consists of two phases. In the *distribute evidence* phase, every node $A \in \mathcal{C}$ in the junction tree that receives a message from its parent, sends a messages m_{AB} to its children.

$$m_{AB}(y_{A\cap B}) = \underset{y_{A \setminus B}}{\operatorname{argmax}} \langle \lambda, \phi(x_A, y_A) \rangle.$$

In the *collect evidence* phase, every node in the junction tree that received messages from all children, sends the messages m_{BA} back to its parent.

After the two phases the junction tree is in equilibrium and further iterations will not affect the actual beliefs. The Viterbi (for sequences) and inside-outside algorithms (for trees) are special cases of message propagation. The most likely labeling can be computed by dynamic programming [9] according to

$$\hat{y} = \underset{y}{\operatorname{argmax}} \sum_{C \in \mathcal{C}} \langle \lambda, \phi_C(x_C, y_C) \rangle - \sum_{AB \in E_J} m_{BA}(y_{A\cap B})$$

The overall complexity is $\mathcal{O}(\exp |V|)$ for building the junction tree and $\mathcal{O}(|\Sigma|^{max|C|})$ for the message passing.

4.2 Loopy Belief Propagation

Similar to Section 4.1, loopy belief propagation propagates messages across the graph. However, instead of using the computationally expensive junction tree, messages are sent in the original graph. Messages encode beliefs about the labeling of direct neighbors in the graph and are passed simultaneously between all nodes. The message from node i to j is computed according to

$$m_{ij}(y_j) = \max_{y_i} \left\{ \langle \lambda, \phi_1(x_i, y_i) \rangle + \langle \lambda, \phi_2(y_i, y_j) \rangle + \sum_{j : y_{ij} \in E} m_{ji}(y_i) \right\}.$$

Although iteratively updating the marginals does not necessarily result in convergence – let alone convergence to the global optimum – loopy belief propagation has been found to work well in practice [7]. After termination, the highest scoring labeling can be computed by

$$\hat{y}_i = \underset{y_i}{\operatorname{argmax}} \left[\langle \lambda, \phi_1(x_i, y_i) \rangle + \sum_{j : y_{ij} \in E} m_{ji}(y_i) \right].$$

The computational cost for each iteration is $\mathcal{O}(|E||Y|^2)$.

4.3 Gibbs Sampling

Repeatedly iterating over the latent variables and drawing a value for each variable y_i according to the conditional probability $\hat{p}(y_i|x, \{y_j : j \neq i\})$ creates a Markov chain of observations that converges to the joint probability $\hat{p}(y|x)$.

This Gibbs sampling rule [4] can be used to approach the assignment of values to y that maximizes the decision function $f(x, y)$. The Gibbs decoder starts with an initial random guess at y. In each iteration, a latent variable Y_i is drawn uniformly and labeled according to $\hat{p}(y_i|x, \{y_j : j \neq i\})$. The new labeling is kept if it is more likely than the previous one. The conditional probability can be derived from the score which the SVM assigns to the pair of input x and output y; we will now describe this process. Parameters λ induce the joint probability $p(y)$ given by

$$\hat{p}(y) \propto \exp\left\{\sum_{i \in V} \langle \lambda, \phi_1(x_i, y_i)\rangle + \sum_{ij \in E} \langle \lambda, \phi_2(y_i, y_j)\rangle\right\},$$

which follows from the definition of the conditional probability. Inserting $p(y)$ in Equation 5 gives us Equation 6. Please note, that nominator and denominator have many factors in common, that is, all factors that do not contain the variable y_i. It is easy to see that cancelling these factors gives Equation 7. This final equation also shows that the full conditionals of y_i only depend on parameters λ and the given feature functions. Furthermore, we only need to consider the neighborhood of the i-th node, i.e. it does not depend on the size of the graph. Let $\bar{y}^\sigma$ defined as $\bar{y}^\sigma_i = y_i$ for $j \neq i$ and $\bar{y}^\sigma_j = \sigma$ in case $i = j$, we have,

$$\hat{p}(y_i|x, \{y_j : j \neq i\}) = \frac{p(Y_i = y_i, x, \{y_j : j \neq i\})}{\sum_{\sigma \in \Sigma} p(Y_i = \sigma, x, \{y_j : j \neq i\})} \tag{5}$$

$$= \frac{\exp\left\{\sum_{k \in V} \langle \lambda, \phi_1(x_k, y_k)\rangle + \sum_{kl \in E} \langle \lambda, \phi_2(y_k, y_l)\rangle\right\}}{\sum_{\sigma \in \Sigma} \exp\left\{\sum_{k \in V} \langle \lambda, \phi_1(x_k, \bar{y}^\sigma_k)\rangle + \sum_{kl \in E} \langle \lambda, \phi_2(\bar{y}^\sigma_k, \bar{y}^\sigma_l)\rangle\right\}} \tag{6}$$

$$= \frac{\exp\left\{\langle \lambda, \phi_1(x_i, y_i) + \sum_{k \in \mathcal{N}(i)} \langle \lambda, \phi_2(y_k, y_i)\rangle\right\}}{\sum_{\sigma \in \Sigma} \exp\left\{\langle \lambda, \phi_1(x_i, \sigma) + \sum_{k \in \mathcal{N}(i)} \langle \lambda, \phi_2(y_k, \sigma)\rangle\right\}} \tag{7}$$

A heuristic test for convergence can be implemented by testing whether the highest-scoring values of the latent variables in several parallel sampling chains remain constant over many iterations. Updating all $|Y|$ variables once is in $\mathcal{O}(|E||Y|^2)$.

5 Empirical Evaluation

In this section, we evaluate SVMs with the exact and approximate inference strategies described in Section 4 in terms of performance, convergence, and execution time. We experiment on the WebKB, Cora, and the Reuters21578 data sets.

In order to evaluate the impact of the size of the graphs on execution time for WebKB and Cora, we generate training instances as follows. Given a maximal graph size m and a data set, we draw a node randomly without replacement from the data. We iteratively add nodes from the neighborhood of the actual graph until the maximal number of nodes is reached or there are no more nodes that can be added. By doing so, we generate graphs of sizes between one and m. We discard singleton graphs and draw training, parameter tuning, and holdout sets randomly from the remaining instances. We employ $m = 2, 4, \ldots, 20$. To guarantee a fair comparison, we make sure that for every data set, all training, tuning, and holdout sets contain on average the same number of documents.

As additional baselines, we include structural multi-class SVMs (mc) [2,13] and a structural SVM that is deprived of all link information (naïve). The main difference between multi-class and naïve SVM is that the former provides slack variables for all documents while the latter relates documents within a graph to the same slack variable [5]. The optimal SVM parameter C is determined for all methods within the interval $[10^{-4}, 10^4]$. When the optimal parameter is at the border of the interval we extend the search appropriately. We report on averages of 100 repetitions with randomly drawn training, tuning, and holdout sets; errorbars indicate standard error.

Loopy belief propagation converges in all our experiments to a stationary distribution. For Gibbs sampling, we employ three chains of length 10000 and use a variance-based criterion to detect convergence.

5.1 WebKB

The WebKB data consist of 8282 web pages from five universities. Every page is labeled with one out of 7 different labels, including *course*, *department*, *student*, and others. We remove tokens with less than four occurrences and employ a bag-of-words representation.

Figure 1 details the error rates for the WebKB data set, where training, tuning, and holdout sets consist on average of 4500 documents, respectively. The two baseline methods, multiclass and naïve, perform worst and almost constantly in the size of the graphs. The exact junction tree together with loopy belief propagation lead to the most accurate prediction models. However, the former could not be computed for graph sizes larger than 8. For loopy belief propagation, we further observe a negative correlation of graph size and error, that is, the larger the graph, the more accurate is the prediction model with loopy belief propagation. Gibbs sampling deteriorates with increasing graph sizes since the number of iterations becomes too small. We will address this issue again in the discussion.

5.2 Cora

The Cora data set consists of 25891 computer science articles of 11 areas including *artificial intelligence*, *machine learning*, *information retrieval*, *databases*, and

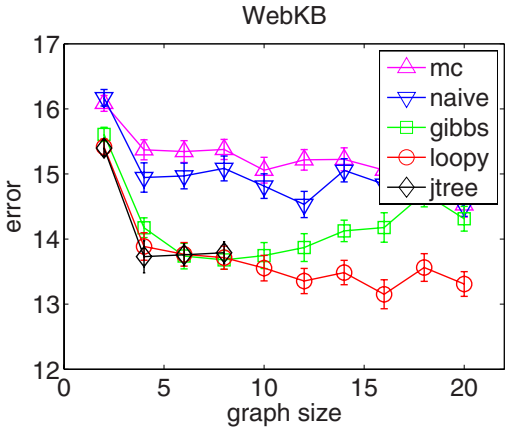


Fig. 1. Results for the WebKB data set

others. We remove stop words and tokens that occurred less than four times and apply a bag-of-words representation of the input data.

Figure 2 details error rates for different graph sizes. The average number of documents in the training, tuning, and holdout sets is 17300, respectively. SVMs utilizing exact junction tree and approximate loopy belief progagation inference perform nearly identical and lead to the most accurate prediction models. However, for the large Cora data set, junction tree inference is only computable up to graph sizes of six. As expected, the multi-class and naïve baselines that do not utilize link information are unaffected by varying sizes of the underlying graphs and perform worst. The Gibbs sampling performs significantly worse than the junction tree algorithm or loopy belief propagation.

5.3 Reuters21578

The Reuters data set consist of 21578 documents from the Reuters news archive. Documents are classified according to their topic into 120 classes, including *grain*, *oil*, and *trade*. Since the topics interdepend and co-occur frequently in a single document, many documents are assigned with multiple labels. We employ a bag-of-words representation for the documents. We represent the multi-class multiple-label problem by a fully connected graph, consisting of a single observation node (the document) and a binary latent variable for every possible label. A value of one indicates the presence of the corresponding topic in the observed document and a value of zero its absence.

In order to quantify the performance of the methods in terms of the number of labels, we organize the data as follows. We begin with the largest two classes and increment the number of labels until we arrive at the twelve largest classes. Since the underlying graphs are completely connected, the number of considered classes is identical to the graph-size. For each graph size we utilize all 21578

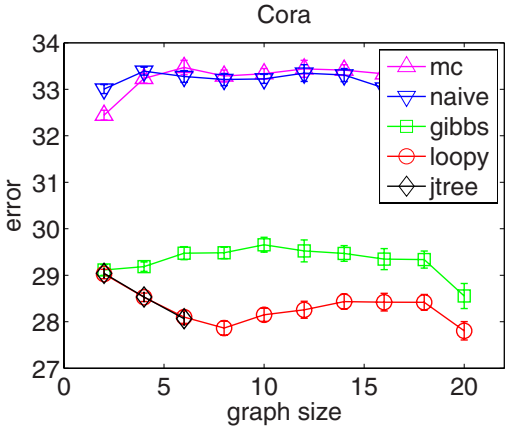


Fig. 2. Results for the Cora data set

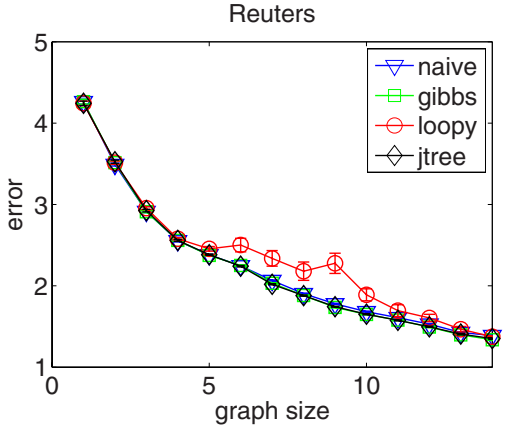


Fig. 3. Results for the Reuters data set

documents that are randomly divided into training, tuning, and holdout sets. All sets are equally sized. We report on averages over 100 repetitions.

Figure 3 shows the results for inference based on naïve, Gibbs sampling, loopy belief propagation, and the exact junction tree algorithm. First of all, every tested method leads to more accurate prediction models when the size of the graph is increased. Moreover, except for loopy belief propagation, all methods perform equally well. As for the junction tree algorithm this is not surprising. The task is also well suited for Gibbs sampling that copes with the small sized graphs and leads to an almost identical performance.

Surprisingly, the naïve approach that does not contain any edge information performs similarly for all numbers of labels. The reason lies in the reduced

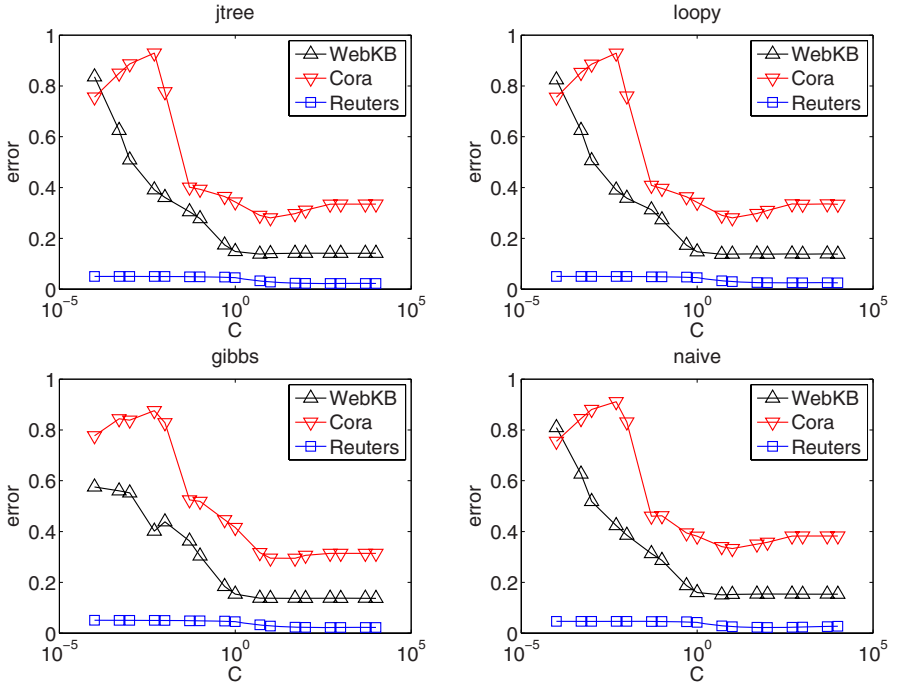


Fig. 4. Results of the parameter search

number of classes in our experimental setup. Focusing on only a subset of all classes implies that the bulk of the documents correspond to discarded labels which carry the dominant information: There are hardly significant co-occurrences of large classes within documents; highly correlated labels involve small classes in the majority of cases. As a consequence, transition probabilities are discarded from the training process and cannot be exploited by the link-based models. A similar observation has been made by Finley and Joachims [3]. In line with their findings is also the performance of loopy belief propagation that is significantly worse compared to the other inference strategies. Loopy belief propagation struggles with the fully connected graph and frequently ends up in local maxima. This is also reflected by the large standard errors.

5.4 Parameter Optimization

The results for the parameter search are shown in Figure 4 that depicts holdout errors across the search interval. For all data sets, the optimal values for the trade-off parameters lie in the right half of the search spaces. For WebKB and Reuters, all methods provide a large region in which an almost optimal value can be found, that is, $C^* \gg 1$. The optimal parameters for the Cora data set are found in a narrow interval around $C^* \approx 10$.

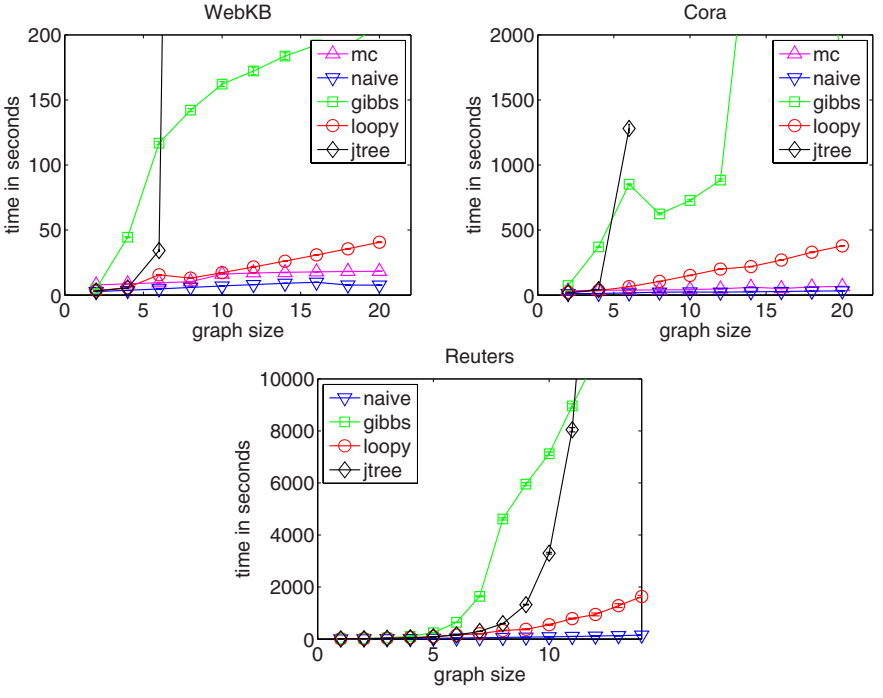


Fig. 5. Execution time for Reuters (left), Cora (center), and WebKB (right)

5.5 Execution Time

Intuitively, the exact junction tree exhibits the worst execution time for all data sets. The multiclass and naïve baselines that do not exploit link information perform comparably fast across the data sets; both are unaffected by graph sizes. Loopy belief propagation and Gibbs sampling differ significantly in their execution time, although they are both in $\mathcal{O}(|E||Y|^2)$. While loopy belief propagation scales well for increasing graph sizes, Gibbs sampling turns out to be computationally demanding.

For small graphs, the exact junction tree algorithm can be computed with only little additional resources compared to loopy belief propagation. However, the junction tree scales exponentially in the size of the graph and exhibits the worst execution time for moderately sized graphs; exact inference is not feasible for larger structures.

6 Discussion

For two out of three studied tasks, approximate inference with loopy belief propagation is competitive to the exact junction tree and leads to comparable results. We observe a positive correlation between the size of the graphs and the achieved

accuracies. For these tasks, the impact of the approximate inference on the SVM algorithm is negligible which can also be verified by monitoring the primal-dual gap. However, loopy belief propagation performs poorly for the Reuters data set.

Support vector machines with approximate Gibbs sampling are well suited for small graphs but the performance deteriorates quickly when the size of the graphs is increased. Of course, the number of iterations could also be increased to cope with increasing graph sizes but a look at the execution times of Gibbs sampling indicates that it is already demanding in terms of computational time. Increasing the number of iterations would clearly multiply the required ressources.

Inference with the junction tree algorithm consistently leads to the most accurate prediction models but due to its computational complexity, junction tree algorithms can only be applied to small graphs. However, the accuracies achieved by the junction tree algorithm for small graphs is hardly beaten by other methods. Thus, to circumvent the computational dead end, our findings indicate that it is often beneficial to split large graphs into smaller components for which the junction tree algorithm can be computed. The support vector optimization algorithm allows the inclusion of several training instances and thus balances the discarded information.

7 Conclusion

In this paper, we compared exact and approximate inference strategies for labeling graphs with support vector machines. We studied the exact junction tree algorithm and approximate inference with loopy belief propagation and Gibbs sampling. Our findings showed that the exact junction tree inference leads to the most reliable and to the most accurate prediction models in our discourse area. By contrast, the tested approximate inference strategies performed throughout inconsistently and led to poor predictions on some data sets.

To remedy the computational barrier of the junction tree, our results showed that decomposing large graphs into smaller subgraphs is beneficial in several ways: Including the subgraphs as additional training examples in the learning process not only rendered exact inference feasible but also preserved reliable results.

Acknowledgements. This work was supported in part by the German Science Foundation DFG and by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886. We are grateful to Steffen Bickel, Laura Dietz, and Klaus-Robert Müller for valuable discussions and comments.

References

1. Cowell, R.G., Philip, Lauritzen, S.L., Spiegelhalter, D.J.: Probabilistic Networks and Expert Systems (Information Science and Statistics). Springer, Heidelberg (May 2003)
2. Crammer, K., Singer, Y.: On the algorithmic implementation of multi-class kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2001)

3. Finley, T., Joachims, T.: Parameter learning for loopy markov random fields with structural support vector machines. In: ICML Workshop on Constraint Optimization and Structured Output Spaces (2007)
4. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. IEEE Trans. Pattern Analysis and Machine Intelligence 6, 721–741 (1984)
5. Joachims, T.: A support vector method for multivariate performance measures. In: Proceedings of the International Conference on Machine Learning (2005)
6. Lafferty, J., Zhu, X., Liu, Y.: Kernel conditional random fields: representation and clique selection. In: Proceedings of the International Conference on Machine Learning (2004)
7. Murphy, K., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: An empirical study. In: Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (1999)
8. Neville, J., Jensen, D.: Collective classification with relational dependency networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2003 (2003)
9. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, San Mateo (1988)
10. Sen, P., Getoor, L.: Empirical comparison of approximate inference algorithms for networked data. In: ICML Workshop on Statistical Relational Learning (SRL) (2006)
11. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (2002)
12. Taskar, B., Guestrin, C., Koller, D.: Max–margin Markov networks. In: Advances in Neural Information Processing Systems (2004)
13. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR 6, 1453–1484 (2005)

Extracting Semantic Networks from Text Via Relational Clustering

Stanley Kok and Pedro Domingos

Department of Computer Science and Engineering
University of Washington, Seattle WA 98195-2350, USA
{koks,pedrod}@cs.washington.edu

Abstract. Extracting knowledge from text has long been a goal of AI. Initial approaches were purely logical and brittle. More recently, the availability of large quantities of text on the Web has led to the development of machine learning approaches. However, to date these have mainly extracted ground facts, as opposed to general knowledge. Other learning approaches can extract logical forms, but require supervision and do not scale. In this paper we present an unsupervised approach to extracting semantic networks from large volumes of text. We use the TextRunner system [1] to extract tuples from text, and then induce general concepts and relations from them by jointly clustering the objects and relational strings in the tuples. Our approach is defined in Markov logic using four simple rules. Experiments on a dataset of two million tuples show that it outperforms three other relational clustering approaches, and extracts meaningful semantic networks.

1 Introduction

A long-standing goal of AI is to build an autonomous agent that can read and understand text. The natural language processing (NLP) community attempted to achieve this goal in the 1970's and 1980's by building systems for understanding and answering questions about simple stories [3,13,23,6]. These systems parsed text into a network of predefined concepts, and created a knowledge base from which inferences can be made. However, they required a large amount of manual engineering, only worked on small text sizes, and were not robust enough to perform well on unrestricted naturally occurring text. Gradually, research in this direction petered out.

Interest in the goal has been recently rekindled [16][7] by the abundance of easily accessible Web text, and by the substantial progress over the last few years in machine learning and NLP. The confluence of these three developments led to efforts to extract facts and knowledge bases from the Web [4]. Two recent steps in this direction are a system by Pasca et. al [18] and TextRunner [1]. Both systems extract facts on a large scale from Web corpora in an unsupervised manner. Pasca et. al's system derives relation-specific extraction patterns from a starting set of seed facts, acquires candidate facts using the patterns, adds high-scoring facts to the seeds, and iterates until some convergence criterion. TextRunner uses a domain-independent approach to extract a large set of

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 624–639, 2008.

relational tuples of the form $r(x, y)$ where x and y are strings denoting objects, and r is a string denoting a relation between the objects. It uses a lightweight noun phrase chunker to identify objects, and heuristically determines the text between objects as relations. These are good first steps, but they still fall short of the goal. While they can quickly acquire a large database of ground facts in an unsupervised manner, they are not able to learn general knowledge that is embedded in the facts.

Another line of recent research takes the opposite approach. Semantic parsing [26,17,29] is the task of mapping a natural language sentence into logical form. The logical statements constitute a knowledge base that can be used to perform some task like answering questions. Semantic parsing systems require a training corpus of sentences annotated with their associated logical forms (i.e., they are supervised). These systems are then trained to induce a parser that can convert novel sentences to their logical forms. Even though these systems can create knowledge bases directly, their need for annotated training data prevents them from scaling to large corpora like the Web.

In this paper, we present SNE, a scalable, unsupervised, and domain-independent system that simultaneously extracts high-level relations and concepts, and learns a semantic network [20] from text. It first uses TextRunner to extract ground facts as triples from text, and then extract knowledge from the triples. TextRunner's triples are noisy, sparse, and contain many co-referent objects and relations. Our system has to overcome these challenges in order to extract meaningful high-level relations and concepts from the triples in an unsupervised manner. It does so with a probabilistic model that clusters objects by the objects that they are related to, and that clusters relations by the objects they relate. This allows information to propagate between clusters of relations and clusters of objects as they are created. Each cluster represents a high-level relation or concept. A concept cluster can be viewed as a node in a graph, and a relation cluster can be viewed as links between the concept clusters that it relates. Together the concept clusters and relation clusters define a simple semantic network. Figure 1 illustrates part of a semantic network that our approach learns. SNE is short for Semantic Network Extractor.

SNE is based on Markov logic [22], and is related to the Multiple Relational Clusterings (MRC) model [12] we recently proposed. SNE is our first step towards creating a system that can extract an arbitrary semantic network directly from text. Ultimately, we want to tightly integrate the information extraction TextRunner component and the knowledge learning SNE component to form a self-contained *knowledge extraction* system. This tight integration will enable information to flow between both tasks, allowing them to be solved jointly for better performance [14].

We begin by briefly reviewing Markov logic in the next section. Then we describe our model in detail (Section 3). Next we describe related work (Section 4). After that, we report our experiments comparing our model with three alternative approaches (Section 5). We conclude with a discussion of future work (Section 6).

2 Markov Logic

Markov logic combines first-order logic with Markov networks.

In *first-order logic* [9], formulas are constructed using four types of symbols: constants, variables, functions, and predicates. (In this paper we use only function-free logic.) Constants represent objects in the domain of discourse (e.g., people: `Anna`, `Bob`, etc.). Variables (e.g., `x`, `y`) range over the objects in the domain. Predicates represent relations among objects (e.g., `Friends`), or attributes of objects (e.g., `Student`). Variables and constants may be typed. An *atom* is a predicate symbol applied to a list of arguments, which may be variables or constants (e.g., `Friends(Anna, x)`). A *ground atom* is an atom all of whose arguments are constants (e.g., `Friends(Anna, Bob)`). A *world* is an assignment of truth values to all possible ground atoms. A database is a partial specification of a world; each atom in it is true, false or (implicitly) unknown.

A *Markov network* or *Markov random field* [19] is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$. It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by $P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$ where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to $P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right)$. A feature may be any real-valued function of the state. This paper will focus on binary features, $f_j(x) \in \{0, 1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state $x_{\{k\}}$ of each clique, with its weight being $\log \phi_k(x_{\{k\}})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

A *Markov logic network (MLN)* is a set of weighted first-order formulas. Together with a set of constants representing objects in the domain, it defines a Markov network with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds x specified by the ground Markov network is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_{i \in F} \sum_{j \in G_i} w_i g_j(x)\right)$, where Z is the partition function, F is the set of all first-order formulas in the MLN, G_i is the set of groundings of the ith first-order formula, and $g_j(x) = 1$ if the jth ground formula is true and $g_j(x) = 0$ otherwise. Markov logic enables us to compactly represent complex models in non-i.i.d. domains. General algorithms for inference and learning in Markov logic are discussed in [22].

3 Semantic Network Extraction

We call our model SNE, for Semantic Network Extractor. SNE simultaneously clusters objects and relations in an unsupervised manner, without requiring the number of clusters to be specified in advance. The object clusters and relation clusters respectively form the nodes and links of a semantic network. A link exists between two nodes if and only if a true ground fact can be formed from the symbols in the corresponding relation and object clusters. SNE can cluster objects of different types, and relations of any arity.

When faced with the task of extracting knowledge from noisy and sparse data like that used in our experiments, we have to glean every bit of useful information from the data to form coherent clusters. SNE does this by jointly clustering objects and relations. In its algorithm, SNE allows information from object clusters it has created at each step to be used in forming relation clusters, and vice versa. As we shall see later in our experimental results, this joint clustering approach does better than clustering objects and relations separately.

SNE is defined using a form of finite second-order Markov logic in which variables can range over relations (predicates) as well as objects (constants). Extending Markov logic to second order involves simply grounding atoms with all possible predicate symbols as well as all constant symbols, and allows us to represent some models much more compactly than first-order Markov logic.

For simplicity, we assume that relations are binary in our definition of SNE, i.e., relations are of the form $r(x, y)$ where r is a relation symbol, and x and y are object symbols. (Extending the definition to an arbitrary number of n-ary relations is straightforward.) We use γ_i and Γ_i to respectively denote a cluster and clustering (i.e., a partitioning) of symbols of type i. If r, x, and y are respectively in cluster γ_r, γ_x, and γ_y, we say that $r(x, y)$ is in the *cluster combination* $(\gamma_r, \gamma_x, \gamma_y)$.

The learning problem in SNE consists of finding the cluster assignment $\Gamma = (\Gamma_r, \Gamma_x, \Gamma_y)$ that maximizes the posterior probability $P(\Gamma|R) \propto P(\Gamma, R) = P(\Gamma)P(R|\Gamma)$, where R is a vector of truth assignments to the observable $r(x, y)$ ground atoms.

We define one MLN for the likelihood $P(R|\Gamma)$ component, and one MLN for the prior $P(\Gamma)$ component of the posterior probability with just four simple rules.

The MLN for the likelihood component only contains one rule stating that the truth value of an atom is determined by the cluster combination it belongs to:

$$\forall r, x, y, +\gamma_r, +\gamma_x, +\gamma_y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y \Rightarrow r(x, y)$$

This rule is soft. The "+" notation is syntactic sugar that signifies that there is an instance of this rule *with a separate weight* for each cluster combination $(\gamma_r, \gamma_x, \gamma_y)$. This rule predicts the probability of query atoms given the cluster memberships of the symbols in them. This is known as the *atom prediction* rule. As shown in [12], given a cluster assignment, the MAP weight w_k of an instance of the atom prediction rule is given by $\log(t_k/f_k)$, where t_k is the empirical

number of true atoms in cluster combination k, and f_k is the number of false atoms. Adding smoothing parameters α and β, we estimate the MAP weight as $\log((t_k + \alpha)/(f_k + \beta))$.

Three rules are defined in the MLN for the prior component. The first rule states that each symbol belongs to exactly one cluster:

$$\forall x \; \exists^1 \gamma \; x \in \gamma$$

This rule is hard, i.e., it has infinite weight and cannot be violated.

The second rule imposes an exponential prior on the number of cluster combinations that contain at least one true ground atom. This rule combats the proliferation of cluster combinations and consequent overfitting, and is represented by the formula

$$\forall \gamma_r, \gamma_x, \gamma_y \; \exists r, x, y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y \wedge r(x, y)$$

with negative weight $-\lambda$. The parameter λ is fixed during learning, and is the penalty in log-posterior incurred by adding a cluster combination containing true ground atoms to the model. Thus larger λs lead to fewer cluster combinations being formed. This rule represents the complexity of the model in terms of the number of instances of the atom prediction rule (which is equal to the number of cluster combinations with true ground atoms).

The last rule encodes the belief that most symbols tend to be in different clusters. It is represented by the formula

$$\forall x, x', \gamma_x, \gamma'_x \;\; x \in \gamma_x \wedge x' \in \gamma'_x \wedge x \neq x' \Rightarrow \gamma_x \neq \gamma'_x$$

with positive weight μ. The parameter μ is also fixed during learning. We expect there to be many concepts and high-level relations in a large heterogenous body of text. The tuple extraction process samples instances of these concepts and relations sparsely, and we expect each concept or relation to have only a few instances sampled, in many cases only one. Thus we expect most pairs of symbols to be in different concept and relation clusters.

The equation for the log-posterior, as defined by the two MLNs, can be written in closed form as [1]

$$\log P(\Gamma|R) = \sum_{k \in K} t_k \log \frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} + f_k \log \frac{f_k + \beta}{t_k + f_k + \alpha + \beta} - \lambda m_{cc} + \mu d + \mathcal{C} \quad (1)$$

where K is the set of cluster combinations, m_{cc} is the number of cluster combinations containing at least one true ground atom, d is the number of pairs of symbols that belong to different clusters, and $\mathcal{C}$ is a constant.

[1] The derivation of the log-posterior is given in an online appendix at http://alchemy.cs.washington.edu/papers/kok08.

Rewriting the equation, the log-posterior can be expressed as

$$\log P(\Gamma|R) = \sum_{k \in K^+} t_k \log \frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} + f_k \log \frac{f_k + \beta}{t_k + f_k + \alpha + \beta} + \sum_{k \in K^-} f_k \log \frac{f_k + \beta}{t_k + f_k + \alpha + \beta} - \lambda m_{cc} + \mu d + \mathcal{C} \quad (2)$$

where K^+ is the set of cluster combinations that contains at least one true ground atom, and K^- is the set of cluster combinations that does not contain any true ground atoms. Observe that $|K^+| + |K^-| = |\Gamma_r||\Gamma_x||\Gamma_y|$. Even though it is tractable to compute the first summation over $|K^+|$ (which is at most the number of true ground atoms), it may not be feasible to compute the second summation over $|K^-|$ for large $|\Gamma_i|$s. Hence, for tractability, we assume that all tuples in K^- belong to a single 'default' cluster combination with the same probability p_{false} of being false. The log-posterior is simplified as

$$\log P(\Gamma|R) = \sum_{k \in K^+} \left[t_k \log \left(\frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} \right) + f_k \log \left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta} \right) \right] + \left(|S_r||S_x||S_y| - \sum_{k \in K^+} (t_k + f_k) \right) \log(p_{false}) - \lambda m_{cc} + \mu d + \mathcal{C} \quad (3)$$

where S_i is the set of symbols of type i, and $(|S_r||S_x||S_y| - \sum_{k \in K^+}(t_k + f_k))$ is the number of (false) tuples in K^-.

SNE simplifies the learning problem by performing hard assignment of symbols to clusters (i.e., instead of computing probabilities of cluster membership, a symbol is simply assigned to its most likely cluster). Since, given a cluster assignment, the MAP weights can be computed in closed form, SNE simply searches over cluster assignments, evaluating each assignment by its posterior probability.

SNE uses a bottom-up agglomerative clustering algorithm to find the MAP clustering (Table 1). The algorithm begins by assigning each symbol to its own unit cluster. Next we try to merge pairs of clusters of each type. We create candidate pairs of clusters, and for each of them, we evaluate the change in posterior probability (Equation 3) if the pair is merged. If the candidate pair improves posterior probability, we store it in a sorted list. We then iterate through the list, performing the best merges first, and ignoring those containing clusters that have already been merged. In this manner, we incrementally merge clusters until no merges can be performed to improve posterior probability.

To avoid creating all possible candidate pairs of clusters of each type (which is quadratic in the number of clusters), we make use of canopies [15]. A canopy for relation symbols is a set of clusters such that there exist object clusters γ_x and γ_y, and for all clusters γ_r in the canopy, the cluster combination $(\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom $r(x, y)$. We say that the clusters in the canopy share the *property* (γ_x, γ_y). Canopies for object symbols x and y are similarly defined. We only try to merge clusters in a canopy that is no larger than a

Table 1. The SNE algorithm

function $SNE(S_r, S_x, S_y, R)$
  inputs: S_r, set of relation symbols
    S_x, set of object symbols that appear as first arguments
    S_y, set of object symbols that appear as second arguments
    R, ground $r(x, y)$ atoms formed from the symbols in S_r, S_x, and S_y
  output: a semantic network, $\{(\gamma_r, \gamma_x, \gamma_y) \in \Gamma_r \times \Gamma_x \times \Gamma_y : (\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom$\}$
for each $i \in \{r, x, y\}$
  $\Gamma_i \leftarrow unitClusters(S_i)$
$mergeOccurred \leftarrow true$
while $mergeOccurred$
  $mergeOccurred \leftarrow false$
  for each $i \in \{r, x, y\}$
    $CandidateMerges \leftarrow \emptyset$
    for each $(\gamma, \gamma') \in \Gamma_i \times \Gamma_i$
      $\Delta P \leftarrow$ change in $P(\{\Gamma_r, \Gamma_x, \Gamma_y\}|R)$ if γ, γ' are merged
      if $\Delta P > 0$, $CandidateMerges \leftarrow CandidateMerges \cup \{(\gamma, \gamma')\}$
    sort $CandidateMerges$ in descending order of ΔP
    $MergedClusters \leftarrow \emptyset$
    for each $(\gamma, \gamma') \in CandidateMerges$
      if $\gamma \notin MergedClusters$ and $\gamma' \notin MergedClusters$
        $\Gamma_i \leftarrow (\Gamma_i \setminus \{\gamma, \gamma'\}) \cup \{\gamma \cup \gamma'\}$
        $MergedClusters \leftarrow MergedClusters \cup \{\gamma\} \cup \{\gamma'\}$
        $mergedOccurred \leftarrow true$
return $\{(\gamma_r, \gamma_x, \gamma_y) \in \Gamma_r \times \Gamma_x \times \Gamma_y : (\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom$\}$

parameter $CanopyMax$. This parameter limits the number of candidate cluster pairs we consider for merges, making our algorithm more tractable. Furthermore, by using canopies, we only try 'good' merges, because symbols in clusters that share a property are more likely to belong to the same cluster than those in clusters with no property in common.

Note that we can efficiently compute the change in posterior probability (ΔP in Table 1) by only considering the cluster combinations with true ground atoms that contain the merged clusters γ and γ'. Below we give the equation for computing ΔP when we merge relation clusters γ_r and γ_r' to form γ_r''. The equations for merging object clusters are similar. Let TF_k be a shorthand for $t_k \log(\frac{t_k+\alpha}{t_k+f_k+\alpha+\beta}) + f_k \log(\frac{f_k+\beta}{t_k+f_k+\alpha+\beta})$.

$$\begin{aligned}
\Delta P = & \sum_{(\gamma_r'', \gamma_1, \gamma_2) \in K^+_{\gamma_r'' \gamma_r' \gamma_r}} \left[TF_{(\gamma_r'', \gamma_1, \gamma_2)} - TF_{(\gamma_r', \gamma_1, \gamma_2)} - TF_{(\gamma_r, \gamma_1, \gamma_2)} + \lambda \right] \\
& + \sum_{(\gamma_r'', \gamma_1, \gamma_2) \in K^+_{\gamma_r'' \cdot \gamma_r}} \left[TF_{(\gamma_r'', \gamma_1, \gamma_2)} - f_{(\gamma_r', \gamma_1, \gamma_2)} \log(p_{false}) - TF_{(\gamma_r, \gamma_1, \gamma_2)} \right] \\
& + \sum_{(\gamma_r'', \gamma_1, \gamma_2) \in K^+_{\gamma_r'' \gamma_r' \cdot}} \left[TF_{(\gamma_r'', \gamma_1, \gamma_2)} - TF_{(\gamma_r', \gamma_1, \gamma_2)} - f_{(\gamma_r, \gamma_1, \gamma_2)} \log(p_{false}) \right] \\
& - \mu |\gamma_r{}'| |\gamma_r| \qquad (4)
\end{aligned}$$

where $K^+_{\gamma_r'' \gamma_r' \gamma_r}$ is the set of cluster combinations with true ground atoms such that each cluster combination $(\gamma_r'', \gamma_1, \gamma_2)$ in the set has the property that

$(\gamma_r^{'}, \gamma_1, \gamma_2)$ and $(\gamma_r, \gamma_1, \gamma_2)$ also contains true atoms. $K^{+}_{\gamma_r^{''}.\gamma_r}$ is the set of cluster combinations with true ground atoms such that each cluster combination $(\gamma_r^{''}, \gamma_1, \gamma_2)$ in the set has the property that $(\gamma_r, \gamma_1, \gamma_2)$, but not $(\gamma_r^{'}, \gamma_1, \gamma_2)$, contains true ground atoms. $K^{+}_{\gamma_r^{''}\gamma_r^{'}.}$ is similarly defined. Observe that we only sum over cluster combinations with true ground atoms that contains the affected clusters γ_r, $\gamma_r^{'}$ and $\gamma_r^{''}$, rather than over all cluster combinations with true ground atoms.

4 Related Work

Rajaraman and Tan [21] propose a system that learns a semantic network by clustering objects but not relations. While it anecdotally shows a snippet of its semantic network, an empirical evaluation of the network is not reported. Hasegawa et. al [10] propose an unsupervised approach to discover relations from text. They treat the short text segment between each pair of objects as a relation, and cluster pairs of objects using the similarity between their relation strings. Each cluster corresponds to a relation, and a pair of objects can appear in at most one cluster (relation). In contract, SNE allows a pair of objects to participate in multiple relations (semantic statements). Shinyama and Sekine [25] form (possibly overlapping) clusters of tuples of objects (rather than just pairs of objects). They use the words surrounding the objects in the same sentence to form a pattern. Objects in sentences with the same pattern are deemed to be related in the same way, and are clustered together. All three previous systems are not domain-independent because they rely on name entity (NE) taggers to identify objects in text. The concepts and relations that they learn are restricted by the object types that can be identified with the NE taggers. All three systems also use ad-hoc techniques that do not give a probability distribution over possible worlds, which we need in order to perform inference and answer queries. By only forming clusters of (tuples of) objects, and not relations, they do not explicitly learn high-level relations like SNE.

ALICE [2] is a system for lifelong knowledge extraction from a Web corpus. Like SNE, it uses TextRunner's triples as input. However, unlike SNE, it requires background knowledge in the form of an existing domain-specific concept taxonomy, and does not cluster relations into higher level ones.

RESOLVER [28] is a system that takes TextRunner's triples as input, and resolves references to the same object and relations by clustering the references together (e.g., `Red_Planet` and `Mars` are clustered together). In contrast, SNE learns abstract concepts and relations (e.g., `Mars`, `Venus`, `Earth`, etc. are clustered together to form the concept of 'planet'). Unlike SNE, RESOLVER's probabilistic model clusters objects and relations separately rather than jointly. To allow information to propagate between object clusters and relation clusters, RESOLVER uses an ad-hoc approach. In its experiments, RESOLVER gives similar results with or without the ad-hoc approach. In contrast, we show in our experiments that SNE gives better performance with joint rather than separate clustering (see Table 3). In a preliminary experiment where we adapt SNE to

only use string similarities between objects (and relations), we find that SNE performs better than RESOLVER on an entity resolution task on the dataset described in Section 5.

5 Experiments

Our goal is to create a system that is capable of extracting semantic networks from what is arguably the largest and most accessible text resource — the Web. Thus in our experiments, we use a large Web corpus to evaluate the effectiveness of SNE's relational clustering approach in extracting a simple semantic network from it. Since to date, no other system could do the same, we had to modify three other relational clustering approaches so that they could run on our large Web-scale dataset, and compared SNE to them. The three approaches are Multiple Relational Clusterings [12], Information-Theoretic Co-clustering [5], and the Infinite Relational Model [11].

5.1 Multiple Relational Clusterings

Like SNE, MRC is a model that simultaneously clusters objects and relations without requiring the number of clusters to be specified in advance. However, unlike SNE, MRC is able to find multiple clusterings, rather than just one. MRC is also defined using finite second-order Markov logic. The main difference between SNE and MRC is in the search algorithm used. MRC also differs from SNE in having an exponential prior on the number of clusters rather than on the number of cluster combinations with true ground atoms. MRC calls itself recursively to find multiple clusterings. We can view MRC as growing a tree of clusterings, and it returns the finest clusterings at the leaves. In each recursive call, MRC uses a top-down generate-and-test greedy algorithm with restarts to find the MAP clustering of the subset of relation and constant symbols it received. While this 'blind' generate-and-test approach may work well for small datasets, it will not be feasible for large Web-scale datasets like the one used in our experiments. For such large datasets, the search space will be so enormous that the top-down algorithm will generate too many candidate moves to be tractable. In our experiments, we replaced MRC's search algorithm with the algorithm in Table 1. We use MRC1 to denote an MRC model that is restricted to find a single clustering.

5.2 Information-Theoretic Co-clustering

The ITC model [5] clusters discrete data in a two-dimensional matrix along both dimensions simultaneously. It greedily searches for the hard clusterings that optimize the mutual information between the row and column clusters. The model has been shown to perform well on noisy and sparse data. ITC's top-down search algorithm has the flavor of K-means, and requires the number of row and column clusters to be specified in advance. At every step, ITC finds

the best cluster for each row or column by iterating through all clusters. This will not be tractable for large datasets like our Web dataset, which can contain many clusters. Thus, we instead use the algorithm in Table 1 (ΔP in Table 1 is set to the change in mutual information rather than the change in log-posterior probability). Notice that, even if ITC's search algorithm were tractable, we would not be able to apply it to our problem because it only works on two-dimensional data. We extend ITC to three dimensions by optimizing the mutual information among the clusters of three dimensions. Furthermore, since we do not know the exact number of clusters in our Web dataset a priori, we follow [5]'s suggestion of using an information-theoretic prior (BIC [24]) to select the appropriate number of clusters. We use ITC-C and ITC-CC to respectively denote the model with a BIC prior on clusters, and the model with a BIC prior on cluster combinations that contain at least one true ground atom. Note that, unlike SNE, ITC does not give a probability distribution over possible worlds, which we need in order to do inference and answer queries (although that is not the focus of this paper).

5.3 Infinite Relational Model

Like SNE, the IRM [11] is a model that simultaneously clusters objects and relations without requiring the number of clusters to be specified in advance. It defines a generative model for the predicates and cluster assignments. It assumes that the predicates are conditionally independent given the cluster assignments, and the cluster assignments for each type are independent. IRM uses a Chinese restaurant process (CPR) prior on the cluster assignments. Under the CRP, each new object is assigned to an existing cluster with probability proportional to the cluster size. IRM assumes that the probability p of an atom being true conditioned on cluster membership is generated according to a Beta distribution, and that the truth values of atoms are then generated according to a Bernoulli distribution with parameter p. IRM finds the MAP cluster assignment using a top-down search similar to MRC, except that it also searches for the optimal values of its CRP and Beta parameters. As mentioned earlier, top-down search is not feasible for large Web-scale data, so we replace IRM's search algorithm with the one in Table 1. We also fixed the values of the CRP and Beta parameters. As in SNE, we assumed that the atoms in cluster combinations with only false atoms belonged to a default cluster combination, and they had the same probability p_{false} of being false. We also experimented with a CRP prior on cluster combinations that contain at least one true atom. We use IRM-C and IRM-CC to respectively denote the IRM with a CRP prior on clusters, and the IRM with a CRP prior on cluster combinations. Xu et al. [27] proposed a model closely related to the IRM.

5.4 Dataset

We compared the various models on a dataset of about 2.1 million triples[2] extracted in a Web crawl by TextRunner [1]. Each triple takes the form $r(x, y)$

[2] Publicly available at http://knight.cis.temple.edu/~yates/data/resolver_data.tar.gz

where r is a relation symbol, and x and y are object symbols. Some example triples are: `named_after`(`Jupiter`, `Roman_god`) and `upheld`(`Court`, `ruling`). There are 15,872 distinct r symbols, 700,781 distinct x symbols, and 665,378 distinct y symbols. Two characteristics of TextRunner's extractions are that they are sparse and noisy. To reduce the noise in the dataset, our search algorithm (Table 1) only considered symbols that appeared at least 25 times. This leaves 10,214 r symbols, 8942 x symbols, and 7995 y symbols. There are 2,065,045 triples that contain at least one symbol that appears at least 25 times. In all experiments, we set the $CanopyMax$ parameter to 50. We make the closed-world assumption for all models (i.e., all triples not in the dataset are assumed false).

5.5 SNE vs. MRC

We compared the performances of SNE and MRC1 in learning a *single* clustering of symbols. We set the λ, μ and p_{false} parameters in SNE to 100, 100 and 0.9999 respectively based on preliminary experiments. We set SNE's α and β parameters to 2.81×10^{-9} and $10 - \alpha$ so that $\frac{\alpha}{\alpha+\beta}$ is equal to the fraction of true triples in the dataset. (A priori, we should predict the probability that a ground atom is true to be this value.) We evaluated the clusterings learned by each model against a gold standard manually created by the first author. The gold standard assigns 2688 r symbols, 2568 x symbols, and 3058 y symbols to 874, 511, and 700 non-unit clusters respectively. We measured the pairwise precision, recall and F1 of each model against the gold standard. Pairwise precision is the fraction of symbol pairs in learned clusters that appear in the same gold clusters. Pairwise recall is the fraction of symbol pairs in gold clusters that appear in the same learned clusters. F1 is the harmonic mean of precision and recall. For the weight of MRC1's exponential prior on clusters, we tried the following values and pick the best: 0, 1, 10–100 (in increments of 10), and 110–1000 (in increments of 100). We report the precision, recall and F1 scores that are obtained with the best value of 80. From Table 2, we see that SNE performs significantly better than MRC1.

We also ran MRC to find multiple clusterings. Since the gold standard only defines a single clustering, we cannot use it to evaluate the multiple clusterings. We provide a qualitative evaluation instead. MRC returns 23,151 leaves that contain non-unit clusters, and 99.8% of these only contain 3 or fewer clusters of size 2. In contrast, SNE finds many clusters of varying sizes (see Table 6). The

Table 2. Comparison of SNE and MRC1 performances on gold standard. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.460	0.061	**0.108**	0.558	0.062	**0.112**
MRC1	0.054	0.044	0.049	0.031	0.007	0.012	0.059	0.011	0.018

Table 3. Comparison of SNE performance when it clusters relation and object symbols jointly and separately. SNE-Sep clusters relation and object symbols separately. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.460	0.061	**0.108**	0.558	0.062	**0.112**
SNE-Sep	0.597	0.116	0.194	0.519	0.045	0.083	0.551	0.047	0.086

poor performance of MRC in finding multiple clusterings is due to data sparsity. In each recursive call to MRC, it only receives a small subset of the relation and object symbols. Thus with each call the data becomes sparser, and there is not enough signal to cluster the symbols.

5.6 Joint vs. Separate Clustering of Relations and Objects

We investigated the effect of having SNE only cluster relation symbols, first-argument object symbols, or second-argument object symbols, e.g., if SNE cluster relation symbols, then it does not cluster both kinds of object symbols. From Table 3, we see that SNE obtains a significantly higher F1 when it clusters relations and objects jointly than when it clusters them separately.

5.7 SNE vs. IRM and ITC

We compared IRM-C and IRM-CC with respect to the gold standard. We set IRM's Beta parameters to the values of SNE's α and β, and set p_{false} to the same value as SNE's. We tried the following values for the parameter of the CRP priors: 0.25, 0.5, 0.75, 1–10 (in increments of 1), 20–100 (in increments of 10). We found that the graphs showing how precision, recall, and F1 vary with the CRP value are essentially flat for both IRM-C and IRM-CC. Both system perform about the same. The slightly higher precision, recall, and F1 scores occur at the low end of the values we tried, and we use the best one of 0.25 for the slightly better-performing IRM-CC system. Henceforth, we denote this IRM as IRM-CC-0.25, and use it for other comparisons.

We also compared SNE, IRM-CC-0.25, ITC-C, and ITC-CC. From Table 4, we see that ITC performs better with a BIC prior on cluster combinations than a BIC prior on clusters. We also see that SNE performs the best in terms of F1.

We then evaluated SNE, IRM-CC-0.25 and ITC-CC in terms of the semantic statements that they learned. A cluster combination that contains a true ground atom corresponds to a semantic statement. SNE, IRM-CC-0.25 and ITC-CC respectively learned 1,464,965, 1,254,995 and 82,609 semantic statements. We manually inspected semantic statements containing 5 or more true ground atoms, and counted the number that were correct. Table 5 shows the results. Even though SNE's accuracy is smaller than IRM-CC-0.25's and ITC-CC's by 11%

Table 4. Comparison of SNE, IRM-CC-0.25, ITC-CC, and ITC-C performances on gold standard. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.461	0.061	**0.108**	0.558	0.062	**0.112**
IRM-CC-0.25	0.201	0.089	0.124	0.252	0.043	0.073	0.307	0.041	0.072
ITC-CC	0.773	0.003	0.006	0.470	0.047	0.085	0.764	0.002	0.004
ITC-C	0.000	0.000	0.000	0.571	0.000	0.000	0.333	0.000	0.000

Table 5. Evaluation of semantic statements learned by SNE, IRM-CC-0.25, and ITC-CC

Model	**Total Statements**	**Num. Correct**	**Fract. Correct**
SNE	1241	965	0.778
IRM-CC-0.25	487	426	0.874
ITC-CC	310	259	0.835

and 7% respectively, SNE more than compensates for the lower accuracy by learning 127% and 273% more correct statements respectively. Figure 1 shows examples of correct semantic statements learned by SNE.

SNE, IRM-CC-0.25 and ITC-CC respectively ran for about 5.5 hours, 9.5 hours, and 3 days on identically configured machines. ITC-CC spent most of its time computing the mutual information among three clusters. To compute the mutual information, given any two clusters, we have to retrieve the number of cluster combinations that contain the two clusters. Because of the large number of cluster pairs, we choose to use a data structure (red-black tree) that is space-efficient, but pays a time penalty when looking up the required values.

5.8 Comparison of SNE with WordNet

We also compared the object clusters that SNE learned with WordNet [8], a hand-built semantic lexicon for the English language. WordNet organizes 117,798 distinct nouns into a taxonomy of 82,115 concepts. There are respectively 4883 first-argument, and 5076 second-argument object symbols that appear at least 25 times in our dataset, and also in WordNet. We converted each node (synset) in WordNet's taxonomy into a cluster containing its original concepts, and all its children concepts. We then matched each SNE cluster to the WordNet cluster that gave the best F1 score. We measured F1 as the harmonic mean of precision and recall. Precision is the fraction of symbols in an SNE cluster that is also in the matched WordNet cluster. Recall is the fraction of symbols in a WordNet cluster that is also in the corresponding SNE cluster. Table 6 shows how precision, recall, and F1 vary with cluster sizes. (The scores are averaged over all

Table 6. Comparison of SNE object clusters with WordNet

Cluster Size	Num. Clusters	Level	Prec.	Recall	F1
47	1	7.0±0.0	0.8±0.0	0.2±0.0	0.4±0.0
36	1	8.0±0.0	0.3±0.0	0.3±0.0	0.3±0.0
24	1	6.0±0.0	0.2±0.0	0.3±0.0	0.2±0.0
19	1	7.0±0.0	0.2±0.0	0.3±0.0	0.2±0.0
16	1	7.0±0.0	0.3±0.0	0.3±0.0	0.3±0.0
12	3	7.0±0.7	0.5±0.1	0.7±0.1	0.5±0.2
11	1	6.0±0.0	0.9±0.0	0.7±0.0	0.8±0.0
10	2	5.5±0.7	0.6±0.1	0.9±0.1	0.5±0.1
8	5	7.0±0.9	0.4±0.2	0.7±0.4	0.3±0.1
7	4	6.0±1.4	0.7±0.3	0.8±0.2	0.9±0.1
6	12	6.6±1.7	0.4±0.2	0.6±0.2	0.6±0.2
5	12	7.2±1.6	0.4±0.2	0.5±0.3	0.7±0.1
4	84	7.2±1.7	0.4±0.1	0.7±0.2	0.6±0.2
3	185	7.3±1.8	0.5±0.2	0.7±0.2	0.7±0.2
2	1419	7.2±1.8	0.6±0.1	0.7±0.1	0.8±0.1

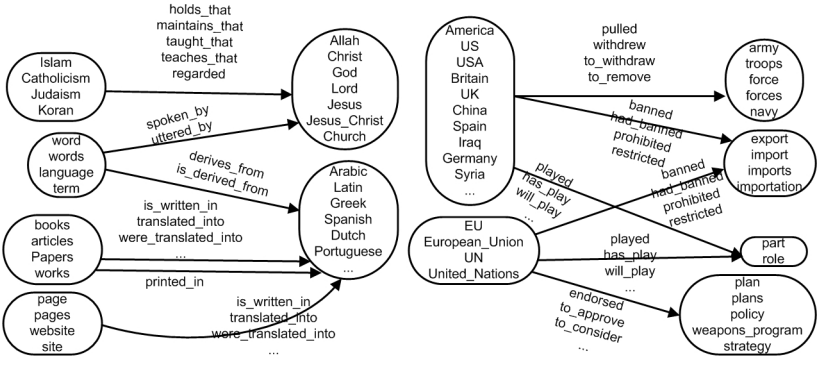


Fig. 1. Fragments of a semantic network learned by SNE. Nodes are concept clusters, and the labels of links are relation clusters. More fragments are available at http://alchemy.cs.washington.edu/papers/kok08.

object clusters of the same size). We see that the F1s are fairly good for object clusters of size 7 or less. The table also shows how the level of the matched cluster in WordNet's taxonomy vary with cluster size. The higher the level, the more specifc the concept represented by the matched WordNet cluster. For example, clusters at level 7 correspond to specific concepts like 'country', 'state', 'dwelling', and 'home', while the single cluster at level 0 (i.e., at the root of the taxonomy) corresponds to 'all entities'. We see that the object clusters correspond to fairly specifc concepts in WordNet. We did not compare the relation clusters to WordNet's verbs because the overlap between the relation symbols and the verbs are too small.

6 Conclusion and Future Work

We presented SNE, a scalable, unsupervised, domain-independent system for extracting knowledge in the form of simple semantic networks from text. SNE is based on second-order Markov logic. It uses a bottom-up agglomerative clustering algorithm to jointly cluster relation symbols and object symbols, and allows information to propagate between the clusters as they are formed. Empirical comparisons with three systems on a large real-world Web dataset show the promise of our approach.

Directions for future work include: integrating tuple extraction into SNE's Markov logic framework so that information can flow between semantic network learning and tuple extraction, potentially improving the performance of both; extending the learning mechanism so as to learn richer semantic networks as well as complex logical theories from text; etc.

Acknowledgments. This research was partly funded by DARPA contracts NBCH-D030010/02-000225, FA8750-07-D-0185, and HR0011-07-C-0060, DARPA grant FA8750-05-2-0283, NSF grant IIS-0534881, and ONR grant N-00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

1. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proc. IJCAI 2007, Hyderabad, India. AAAI Press, Menlo Park (2007)
2. Banko, M., Etzioni, O.: Strategies for lifelong knowledge extraction from the web. In: Proc. K-CAP-2007, British Columbia, Canada (2007)
3. Charniak, E.: Toward a Model of Children's Story Comprehension. PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Boston, MA (1972)
4. Craven, M.W., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., Slattery, S.: Learning to extract symbolic knowledge from the World Wide Web. In: Proc. AAAI 1998, Madison, WI, pp. 509–516. AAAI Press, Menlo Park (1998)
5. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proc. KDD 2003, Washington, DC (2003)
6. Dyer, M.G.: In-Depth Understanding. MIT Press, Cambridge (1983)
7. Etzioni, O., Banko, M., Cafarella, M.J.: Machine reading. In: Proc. 2007 AAAI Spring Symposium on Machine Reading, Palo Alto, CA. AAAI Press, Menlo Park (2007)
8. Gellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
9. Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann, San Mateo (1987)

10. Hasegawa, T., Sekine, S., Grishman, R.: Discovering relations among named entities from large corpora. In: Proc. ACL 2004, Barcelona, Spain (2004)
11. Kemp, C., Tenenbaum, J.B., Griffiths, T.L., Yamada, T., Ueda, N.: Learning systems of concepts with an infinite relational model. In: Proc. AAAI 2006, Boston, MA. AAAI Press, Menlo Park (2006)
12. Kok, S., Domingos, P.: Statistical predicate invention. In: Proc. ICML 2007, Corvallis, Oregon, pp. 440–443. ACM Press, New York (2007)
13. Lehnert, W.G.: The Process of Question Answering. Erlbaum, Hillsdale (1978)
14. McCallum, A., Jensen, D.: A note on the unification of information extraction and data mining using conditional-probability, relational models. In: Proc. IJCAI 2003 Workshop on Learning Statistical Models from Relational Data, Acapulco, Mexico, pp. 79–86. IJCAII (2003)
15. McCallum, A., Nigam, K., Ungar, L.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Proc. KDD 2000, pp. 169–178 (2000)
16. Mitchell, T.: Reading the web: A breakthrough goal for AI. AI Magazine 26(3), 12–16 (2005)
17. Mooney, R.J.: Learning for semantic parsing. In: Gelbukh, A. (ed.) CICLing 2007. LNCS, vol. 4394, pp. 311–324. Springer, Heidelberg (2007)
18. Pasca, M., Lin, D., Bigham, J., Lifchits, A., Jain, A.: Names and similarities on the web: Fact extraction on the fast lane. In: Proc. ACL/COLING 2006 (2006)
19. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
20. Quillian, M.R.: Semantic memory. In: Minsky, M.L. (ed.) Semantic Information Processing, pp. 216–270. MIT Press, Cambridge (1968)
21. Rajaraman, K., Tan, A.-H.: Mining semantic networks for knowledge discovery. In: Proc. ICMD 2003 (2003)
22. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62, 107–136 (2006)
23. Schank, R.C., Riesbeck, C.K.: Inside Computer Understanding. Erlbaum, Hillsdale (1981)
24. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 6, 461–464 (1978)
25. Shinyama, Y., Sekine, S.: Preemptive information extraction using unrestricted relation discovery. In: Proc. HLT-NAACL 2006, New York (2006)
26. Wong, Y.W., Mooney, R.J.: Learning synchronous grammars for semantic parsing with lambda calculus. In: Proc. ACL 2007, Prague, Czech Republic (2007)
27. Xu, Z., Tresp, V., Yu, K., Kriegel, H.-P.: Infinite hidden relational models. In: Proc. UAI 2006, Cambridge, MA (2006)
28. Yates, A., Etzioni, O.: Unsupervised resolution of objects and relations on the web. In: Proc. NAACL-HLT 2007, Rochester, NY (2007)
29. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: Structured classification with probabilistic categorial grammers. In: Proc. UAI 2005, Edinburgh, Scotland (2005)

Ranking the Uniformity of Interval Pairs

Jussi Kujala and Tapio Elomaa

Department of Software Systems
Tampere University of Technology
P. O. Box 553, FI-33101 Tampere, Finland
jussi.kujala@tut.fi, elomaa@cs.tut.fi

Abstract. We study the problem of finding the most uniform partition of the class label distribution on an interval. This problem occurs, e.g., in supervised discretization of continuous features, where evaluation heuristics need to find the location of the best place to split the current feature. The weighted average of empirical entropies of the interval label distributions is often used in this task. We observe that this rule is suboptimal, because it prefers short intervals too much. Therefore, we proceed to study alternative approaches. A solution that is based on compression turns out to be the best in our empirical experiments. We also study how these alternative methods affect the performance of classification algorithms.

1 Introduction

We consider the problem of processing labeled and sequential data into intervals — contiguous subsequences — that can be utilized in prediction. This task is encountered, e.g., in the discretization of numerical attributes when learning classifiers. Top-down greedy heuristics reduce this problem to a simpler one [1]: How to rank and compare the uniformity of two adjacent intervals?

Arguably the most often used measure of uniformity of adjacent example intervals is the weighted average entropy over the class label distributions [2]. Using entropy is a well-founded approach that should in principle lead to good results. We cannot, however, compute the true entropy underlying the label distribution. Instead, we need samples to estimate it through the observed empirical entropy.

We will evaluate the suitability of the empirical entropy in finding the least uniform intervals. Our empirical evaluation shows that the estimation of entropy often fails in this task, because it prefers too short intervals.

A similar and related problem occurs in the top-down induction of decision trees, where the tree-building algorithm searches for the most informative attribute. It is known that criteria based on empirical entropy prefer too much attributes with several values. On the other hand, in the interval selection problem we only have two values for the attribute — the left and the right interval. Hence, the failure does not occur because of too many values for the attribute. See the previous work discussed in [3] for more information on attribute selection in decision trees.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 640–655, 2008.

The reasons behind this suboptimal behavior of empirical entropy in decision trees and in splitting an interval are similar. First, the estimation error of entropy is significant, and even more so when fewer samples are available. In particular, this error is biased towards intervals with less data, and some interval pairs *always* contain a short interval, because we consider *all* splits of a larger interval. Second, perhaps a more minor reason is that minimizing entropy does not necessarily coincide with minimum empirical error. Kohavi and Sahami [4] discuss entropy-based and error-based discretization more generally.

In this paper we evaluate three simple approaches intended to rectify the problems of the approach based on empirical entropy. The methods also take into account the number of samples that is available to estimate the distribution. Previously, only the empirical frequencies have been used. We evaluate these three approaches by generating synthetic data from known distributions and observing the distributions of the resulting split points. Our empirical evaluation demonstrates that in terms of reducing the absolute error, the proposed approaches are successful, but their utility in improving prediction error of Naïve Bayes (NB) is smaller.

In the next section we introduce the required preliminaries for the rest of the text; the NB classifier and the recursive entropy heuristic. Section 3 illustrates the failure of average empirical entropy in always choosing the best split point. We also provide a theoretical explanation for this shortcoming. In Section 4 three approaches that try to overcome the problems are put forward: The first approach is based on choosing the split point that yields the best compression of class labels. We can also consider it as maximizing a certain posterior likelihood. In the second approach the maximum likelihood estimation of probability in entropy calculation is replaced by a Bayesian estimate. The third approach replaces entropy with another function with different concavity properties. Section 5 evaluates the proposed approaches empirically. The implications for classification of the approaches studied in this paper are the topic of Section 6. In particular, we consider NB and test the implications also empirically. Finally, we put forward the concluding remarks of this work.

2 Preliminaries

Let us first introduce the NB classifier and then discuss how it relates to discretization and finding non-uniform intervals. Let y denote a variable that takes a value of a class label, and let x denote a vector of d features $\langle x^1, \ldots, x^d \rangle$. We are given a set of n examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$. In the NB classifier we assume that the features are statistically independent given the class, which results in the following probability for a label y given a vector x:

$$\mathbf{P}(y \mid x) \propto \mathbf{P}(y) \prod_{i=1}^{d} \mathbf{P}\left(x^i \mid y\right),$$

where x^i are the independent features. The NB classifier then selects the label with maximal probability. Note that although the statistical independence is

Table 1. The EMP-ENT split point selection method and the recursive entropy heuristic

---

Function EMP-ENT
Input: An interval I.
Output: Two subintervals (I_1, I_2) which partition I.
Algorithm: For all class labels y_j, $j = 1, \ldots, m$, let $\mathbf{P}_I(y_j)$ stand for the *empirical probability* of observing label y_j on interval I. In other words, it is the ratio of labels y_j to all labels in interval I. Now, the *empirical entropy* of the class label distribution of I is

$$H(I) = -\sum_{j=1}^{m} \mathbf{P}_I(y_j) \lg \mathbf{P}_I(y_j).$$

Let $|I|$ denote the number of examples in interval I. The average empirical entropy of a particular split (I'_1, I'_2) is

$$\frac{|I'_1|}{|I|} H(I'_1) + \frac{|I'_2|}{|I|} H(I'_2).$$

Return the split that minimizes the average empirical entropy.

Function RECURSIVE ENTROPY HEURISTIC
Input: An interval I.
Output: A contiguous sequence of subintervals $(I_1, \ldots, I_k)$ which partitions I.
Algorithm: Out of all splits (I'_1, I'_2) select the one given by the EMP-ENT method. If a stopping criterion, such as the MDL rule used by Fayyad and Irani [2] is satisfied, then return only I. Otherwise, return a concatenation of outputs of recursive calls to the recursive entropy heuristic with inputs I'_1 and I'_2.

---

used to derive the decision rule, the NB classifier does not necessarily fail if features are correlated, because it works as long as the correct label has maximal probability [5,6,7].

For discrete features the marginal probabilities $\mathbf{P}(x^i \mid y)$ are easy to estimate by counting from the training examples. One usually smooths the count e.g., with a Laplacian estimate, to prevent assigning the zero probability to some events. Continuous features are more difficult to deal with, but there are several solutions for handling them, none of which appear to dominate the other [8].

Discretization is a solution in which an interval is divided into discrete bins, and the marginal probability is estimated by counting the items that fall into these bins. This approach has attracted significant attention [9]. The best performing methods are founded on recursive entropy heuristic [1,2]. Its aim is to minimize the empirical entropy of the bins, while avoiding creating adjacent bins that appear to come from the same underlying distribution. For k bins, the number of possible bin borders for n items is $\binom{n-1}{k-1}$, which scales in $\mathcal{O}(n^{k-1})$ (we do not accept empty bins in this count). Hence, a brute force approach to the combinatorial explosion is unattainable.

Thus, the recursive entropy heuristic uses the greedy top-down approach. In it one successively splits in two the interval yielding minimum entropy until some stopping criterion is satisfied. This heuristic is detailed in Table 1.

3 How Minimizing Empirical Entropy Can Fail

We have observed empirically that the EMP-ENT rule often proposes using a very short interval — from one to five items. This behavior appears to be a result of random noise in the labels, rather than a correct decision. In this section we first discuss the suitability of using the entropy in finding a location to split, and then proceed to study how empirical estimation of entropy is difficult.

3.1 The Role of Entropy in Selecting Uniform Intervals

The 0/1-loss is the probability of predicting a wrong label. It gives a simple method to select the split location: minimizing the 0/1-loss of the resulting subintervals. In this case we predict the majority label on each subinterval. However, 0/1-loss is blind to those differences in the label distribution that do not change the majority label [9,10].

For example, if the most likely label remains the same on the whole interval, then the 0/1-loss of all possible splits is the same constant. Naturally, if we use these subintervals to predict labels under 0/1-loss, then it is not useful to split the interval in this case. Nevertheless, if we combine the prediction from this and another feature, then we would like the combined predictor to perform well. In this case the interval should be split at the location where it provides the least error for the combined predictor.

This is the motivation for minimizing the joint entropy of subintervals, it can distinguish changes of the label distribution. For example, let $p(x)$ denote the probability of generating the label a at the position x on the interval, and let $1 - p(x)$ be the corresponding probability for the label b. Then the analysis of the Lagrangian reveals that the entropy is minimized only in those locations in which either the majority label changes or the derivative $p'(x)$ is zero.

However, minimizing entropy does not necessarily minimize 0/1-loss, although these two often coincide in practice. Figure 1 illustrates a situation in which the entropy prefers to split at a location where the resulting subintervals are non-uniform. The 0/1-loss, however, is minimized at another location. Hence, using entropy is not justified if we, for example, have a single feature and want to select a single split point on it for a further use in the NB classifier.

Note also that, although the first split point may minimize 0/1-loss, the second does not necessarily give the minimum 0/1-loss for two split points. In the situation of Figure 2 the entropy is minimized in the middle. However, the two split points that minimize 0/1-loss are at the locations in which the most likely label changes.

Therefore, both 0/1-loss and entropy fail in some sense. The loss is blind to some differences in the label distribution, and the entropy fails to provide the

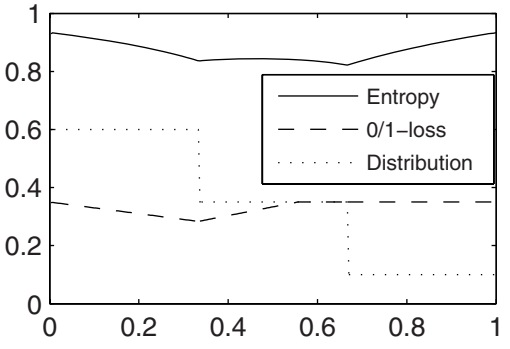


Fig. 1. Entropy and 0/1-loss as functions of a split point on the interval $[0, 1]$. The distribution that generates labels a and b is also shown. The probability of label a changes from 0.6 to 0.35 at the location 0.33 and to 0.1 at the location 0.66. The optimal split point is at the location 0.66 according to entropy, but at the location 0.33 according to 0/1-loss.

minimum 0/1-loss. However, in experiments the entropy usually performs better than minimizing 0/1-loss [4]. Two facts could provide an explanation. First, the problem domains have several features, so several features jointly interact in prediction. Second, due to the recursive nature of the heuristic, the splitting continues until 0/1-loss is minimized or nearly minimized. A small data size might be a problem in this case.

3.2 Empirical Estimation of Entropy

Although entropy is an acceptable measure, we cannot use it directly, because it depends on the hidden underlying distribution. Instead, we have a sample from this distribution. It is known that entropy is difficult to estimate; for instance, there is no unbiased estimate for it [11]. More information on the estimation of the entropy is given, for example, in [11,12].

Let us first give a simple demonstration of using the empirical entropy in our application of splitting an interval and, then, a reasoning behind the observation. Figure 3 illustrates the distribution of the optimal split point according to EMPENT rule of Table 1 on two separate class label distributions. In both cases thirty class labels were drawn so that class label a initially has probability $1/3$ and after the change point probability $2/3$. The change point in distribution A is located in the middle and in distribution B after the fifth item. Ten thousand random intervals were drawn from both distributions.

For both label distributions the correct change point is clearly the most often identified split point location, but there are also noteworthy concentrations at both ends of the interval. These concentrations have no particular justification, they are just bogus local optima. In Section 5 we observe that this behavior occurs, regardless of the interval length, if the optimal split point location of the distribution is unclear.

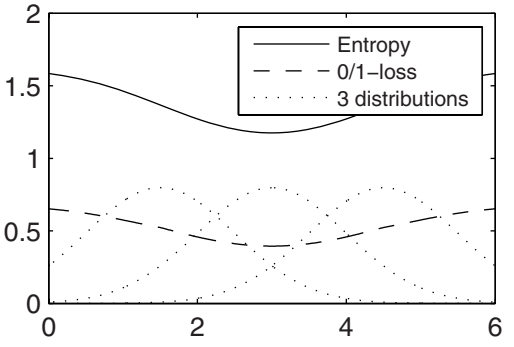


Fig. 2. Three Gaussians generate three labels. Both entropy and 0/1-loss are minimized in the middle. For two split points the 0/1-loss is minimized at the locations, where the most likely label changes.

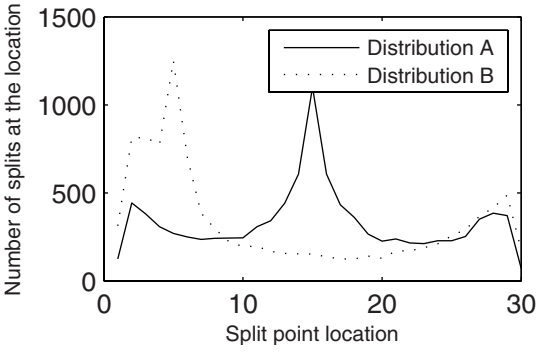


Fig. 3. The optimal split point distribution according to the EMP-ENT rule, for two different distributions of the intervals. Ten thousand random intervals were generated.

Let us consider the reasons for the phenomenon illustrated in Figure 3. Let $\boldsymbol{p}$ denote the real distribution on an interval under consideration and let $\hat{\boldsymbol{p}}$ denote the empirical distribution of a draw from $\boldsymbol{p}$. The empirical entropy $\widehat{H} = H(\hat{\boldsymbol{p}})$ is a random variable of labels drawn from $\boldsymbol{p}$. Two factors contribute to the difference between $H = H(\boldsymbol{p})$ and $\widehat{H}$ [13,6,14]:

1. The *bias*, which we define as

$$H\left(\boldsymbol{p}\right) - \mathbf{E}(H\left(\hat{\boldsymbol{p}}\right)).$$

2. The *variance*, which tells how much $H(\hat{\boldsymbol{p}})$ changes around its expectation:

$$\mathbf{E}\big((H\left(\hat{\boldsymbol{p}}\right) - \mathbf{E}(H\left(\hat{\boldsymbol{p}}\right)))^2\big).$$

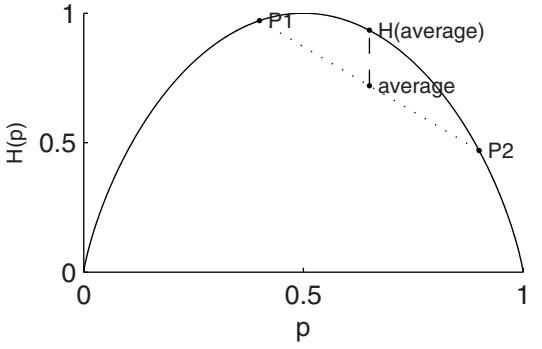


Fig. 4. The concave binary entropy function. Note how the expectation (average) of entropies at points $P1$ and $P2$ is less than the entropy of their expectation.

It is well known that the *mean squared error* (MSE) of estimation is related to these values [6]:

$$\underbrace{\mathbf{E}\left(\left(\widehat{H}-H\right)^2\right)}_{\text{MSE}} = \underbrace{\left(\mathbf{E}\left(\widehat{H}\right)-H\right)^2}_{\text{bias}} + \underbrace{\mathbf{E}\left(\left(\widehat{H}-\mathbf{E}\left(\widehat{H}\right)\right)^2\right)}_{\text{variance}}.$$

Both the bias and variance are affected by the concavity of entropy $H(\boldsymbol{p})$. Jensen's inequality [15] asserts that for any concave function f it holds that

$$\mathbf{E}(f(X)) \leq f\left(\mathbf{E}(X)\right).$$

An immediate consequence is that

$$\mathbf{E}\left(\widehat{H}\right) = \mathbf{E}(H\left(\hat{\boldsymbol{p}}\right)) \leq H\left(\mathbf{E}(\hat{\boldsymbol{p}})\right) = H\left(\boldsymbol{p}\right) = H.$$

For example, the empirical entropy of one sample is always zero.

Another complication is the variance of the empirical entropy. Let $\hat{\boldsymbol{p}}_{\text{ML}}$ be the maximum likelihood estimate of $\boldsymbol{p}$. The estimate is unbiased, but it naturally has a high variance with small sample sizes. This variance can result in larger than expected deviations in $\widehat{H}$ for small values, because the entropy H grows fast around a non-uniform distribution. For example, if we draw 30 labels from a distribution over the labels $\{a, b\}$ with $\mathbf{P}(a)$ equal to $1/5$, then $H = 0.72$ and $\mathbf{E}(\widehat{H}) = 0.7$. Although these values are close to each other, the probability that $\widehat{H} > \mathbf{E}(\widehat{H})$ is 0.57. This implies that low values for $\widehat{H}$ are further from $\mathbf{E}(\widehat{H})$ than high values for it.

Of course, both the bias and the variance tend to zero as the number of samples grows, because the maximum likelihood estimate $\hat{\boldsymbol{p}}_{\text{ML}}$ concentrates around a point and the entropy is almost linear in the neighborhood of this point. Jensen's inequality is strict when the concave function f is linear and, in fact, the difference $f(\mathbf{E}(X)-\mathbf{E}(f(X)))$ depends on the "curvature" of f. Figure 4 demonstrates this graphically.

4 Alternatives to Empirical Entropy

We now study three alternative solutions to the problem(s) identified in the previous section. They differ in their justification and operation. We will study their performance empirically in the next section.

4.1 COMPRESS: Compression

There is another potential justification for the empirical entropy, compression. The MDL principle, roughly, advocates the selection of the model that compresses the data the most [16].

If we know $\hat{\boldsymbol{p}}$, then $H(\hat{\boldsymbol{p}})$ is approximately the expected number of bits needed to encode labels from the distribution $\hat{\boldsymbol{p}}$. However, this is not equal to compressing the interval I, because we know the *exact* number of labels, not just their probabilities. In addition, $\hat{\boldsymbol{p}}$ is unlikely to be close to the true distribution for small sample sizes.

If we know the empirical frequencies of the labels on an interval I, then we can compress it by identifying the permutation that transforms a known interval to I. The number of permutations of the labels on I is

$$\mathbf{Perm}\,(I) = \binom{n}{n_1, \ldots, n_m} = \frac{n!}{n_1! \cdots n_m!},$$

where n is the number of items on the interval I and n_i is the number of items with label i. Therefore, the interval I can be identified with $\lg \mathbf{Perm}\,(I)$ bits, if we know the original permutation that is being transformed to I. This permutation depends on the label counts, which in turn depend on the location of the split point. There are $\binom{n+2(m-1)+1}{2(m-1)+1}$ possible ways to assign these, because in addition to one split point we select for both subintervals $(m-1)$ dummy items that mark the empirical counts of labels. Note that here we allow empty intervals. As the above count is the same for both subintervals, we can ignore it.

Hence, the COMPRESS method, given in Table 2, selects the split point that compresses the labels the most. Previously Kononenko [17] has suggested a similar rule for decision trees. Note that COMPRESS has a relation to the entropy:

$$\lg \mathbf{Perm}\,(I) = \lg n! - \sum_{i=1}^{m} \lg n_i!$$

$$H\left(\hat{\boldsymbol{p}}_{\mathrm{ML}}\right) = n \lg n - \sum_{i=1}^{m} n_i \lg n_i.$$

Stirling's formula asserts that $n! \approx \sqrt{2\pi n}(n/e)^n$, and so $\lg n! \approx n \lg n - n \lg e$.

COMPRESS also has a probabilistic justification. Generate $\boldsymbol{p}$ according to a uniform prior distribution for the possible vectors and then generate I from $\boldsymbol{p}$. Then the probability of observing an interval I is

$$\mathbf{P}(I) = \binom{n+m-1}{m-1}^{-1} \binom{n}{n_1, \ldots, n_m}^{-1}, \tag{1}$$

Table 2. COMPRESS: Selection of the best split point

Input: An interval I.
Output: A splitted interval (I_1, I_2) that partitions I.
Algorithm: For each candidate split (I'_1, I'_2) calculate the following value

$$\mathbf{Perm}\ I'_1\ \cdot \mathbf{Perm}\ I'_2\ ,$$

and return the candidate with the smallest value.

which follows from the normalizing constant of the Dirichlet distribution [18]. The first part consists of the possible empirical frequencies, and is the same for all frequencies. The second part depends on the empirical frequencies and it is $\lg \mathbf{Perm}\,(I)$. Therefore, if we set the prior probability of a split point to be

$$\mathbf{P}(\text{split at } (I_1, I_2) \mid I) = \frac{\binom{|I_1|+m-1}{m-1}\binom{|I_2|+m-1}{m-1}}{\binom{|I|+2(m-1)+1}{2(m-1)+1}},$$

then we select the same decisions as COMPRESS. Note that $\mathbf{P}(\text{split at } (I_1, I_2) \mid I)$ is a proper probability distribution, because it sums to one.

The numerator in the above equation is $\Theta(|I_1||I_2|)^{m-1}$, in which the dependence on m and $|I|$ is ignored. Hence, COMPRESS maximizes a posterior likelihood that gives more prior probability to splits in the middle of the interval. This is intuitive in the sense that we have more information available for these splits.

4.2 BAY-ENT: Bayesian Estimation of the Real Distribution

Another simple approach is to give an estimate $\hat{\boldsymbol{p}}$ of $\boldsymbol{p}$ and use it to estimate the entropy. We already observed that the maximum likelihood estimate can perform poorly, so let us evaluate a Bayesian estimate of $\boldsymbol{p}$.

First, we note that a good estimate for entropy is not necessarily what we want. For example, one correction, which takes into account part of the bias of the empirical entropy, is the Miller-Madow estimate $\widehat{H}(I) + (m-1)/|I|$ [11]. However, using this estimate gives the same result as using only $\widehat{H}(I)$, because after averaging the entropy estimates we add the same constant to the value of each split.

Let us study a Bayesian estimate for $\boldsymbol{p}$, where we give a prior probability $\mathbf{P}(\boldsymbol{p})$ for each $\boldsymbol{p}$. Then, after observing our data I, we can update our beliefs on the distribution of $\boldsymbol{p}$:

$$\mathbf{P}(\boldsymbol{p} \mid I) \propto \mathbf{P}(I \mid \boldsymbol{p})\,\mathbf{P}(\boldsymbol{p})\,.$$

We choose the uniform prior on $\boldsymbol{p}$, which gives an equal likelihood to *observing* any combination of empirical frequencies, as a corollary to Equation (1).

The posterior is a distribution over $\boldsymbol{p}$, but for simplicity we want a single point estimate. It is known, as Zhu and Lu [19] note, that the Dirichlet prior $\mathbf{P}(\boldsymbol{p}) \propto \prod_{i=1}^{m} p_i^{x_i - 1}$ gives a posterior with the expectation

$$\hat{\boldsymbol{p}}_{\text{BAYES}} = \left(\frac{x_1 + n_1}{x + n}, \ldots, \frac{x_m + n_m}{x + m} \right).$$

Setting the x_is to one we obtain our desired point estimate. Note that we can interpret also this method as maximizing a likelihood, because the entropy $H(\hat{\boldsymbol{p}})$ equals $-\lg \mathbf{P}(\hat{\boldsymbol{p}} \mid \hat{\boldsymbol{p}})$.

If the real $\boldsymbol{p}$ is generated according to the prior that we use, then the estimated posterior gives the true posterior distribution. Hence, in this case $\hat{\boldsymbol{p}}_{\text{BAYES}}$ is a good estimate. Unfortunately, often in practice the prior does not hold. Then, theoretically, relatively little can be guaranteed, except that the posterior converges to the real $\boldsymbol{p}$ with enough samples (if the prior is non-zero everywhere) [20]. In this case Gelman [21] suggests trying several non-informative priors, and trusting the results if they agree.

4.3 CONC: Preferring Non-uniform Intervals Less

Let us also study how changing the objective function from entropy to another concave function affects the performance. Note that any symmetric concave function with a mode at the uniform distribution gives the same ranking for two distributions. However, the decisions differ when pairs of distributions are compared with each other, as is the case with possible splits of an interval. More concave functions prefer a split point with more non-uniform intervals.

One problem that we identified with the empirical entropy was the susceptibility to random noise in the labels. This was caused by the preference to the non-uniform intervals. We suggest the following function, which is nearly linear as a function of distance to the uniform distribution.

$$\text{CONC}(\boldsymbol{p}) = \left(1 - \frac{\|\boldsymbol{u} - \boldsymbol{p}\|_2}{Z} \right)^{1-\epsilon},$$

where $\boldsymbol{u}$ is the uniform distribution, Z is the normalizing value $\max \|\boldsymbol{u} - \boldsymbol{p}\|_2$, and ϵ is a small value, such as 0.99.

As this function resembles 0/1-loss, we expect it to fail in similar conditions. On the other hand, it behaves better near a non-uniform distribution. Hence, it is interesting to see its performance. Note that we use $\text{CONC}(\boldsymbol{p})$ instead of a linear function, because it prefers non-uniform pairs. A linear function would give the same rank for all splits, where the majority label does not change. This is often the case when we empirically estimate the distribution, because the sample size is restricted.

Kearns and Mansour [22] have analyzed in connection of decision trees top-down algorithms that are related to discretization. In fact, these algorithms solve the same problem when restricted to a continuous feature. The authors proved

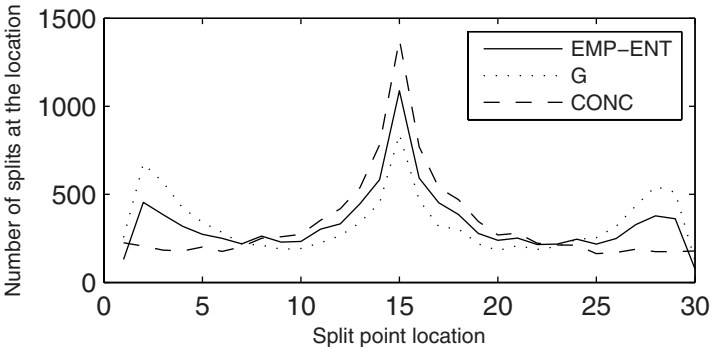


Fig. 5. A quick comparison of functions with different concavity. The distribution that generates the labels and the majority label both change in the middle.

Table 3. Mean squared errors in the tests of Figure 6

	EMP-ENT	COMPRESS	BAY-ENT	CONC
b)	317.50	231.75	272.90	202.91
c)	48.49	44.59	50.60	56.64
d)	15.79	14.25	13.46	11.47
e)	43.89	41.46	41.38	40.51
f)	0.49	0.46	0.44	0.43

formal bounds in the PAC framework. Their analysis suggest using a function that grows faster than the entropy around non-uniform distributions,

$$G(p) = 2\sqrt{p(1-p)}.$$

However, as we have noted, using a concave function is dangerous with small sample sizes. Figure 5 demonstrates this empirically.

5 Empirical Evaluation of the Suggested Solutions

The illustration in Figure 3 gave one example of the behavior of EMP-ENT method, but it does not tell how often we observe, or suffer, from this behavior in practice. We now give results for several different kinds of experiments, and also compare EMP-ENT to the methods put forward in the previous section.

Figure 6 plots the results of six different tests and Table 3 gives the corresponding MSE values. In it MSE is measured from the distance to the correct split point, which is defined by the entropy of the generating distribution. We use MSE in order to penalize more for splitting significantly away from the true location.

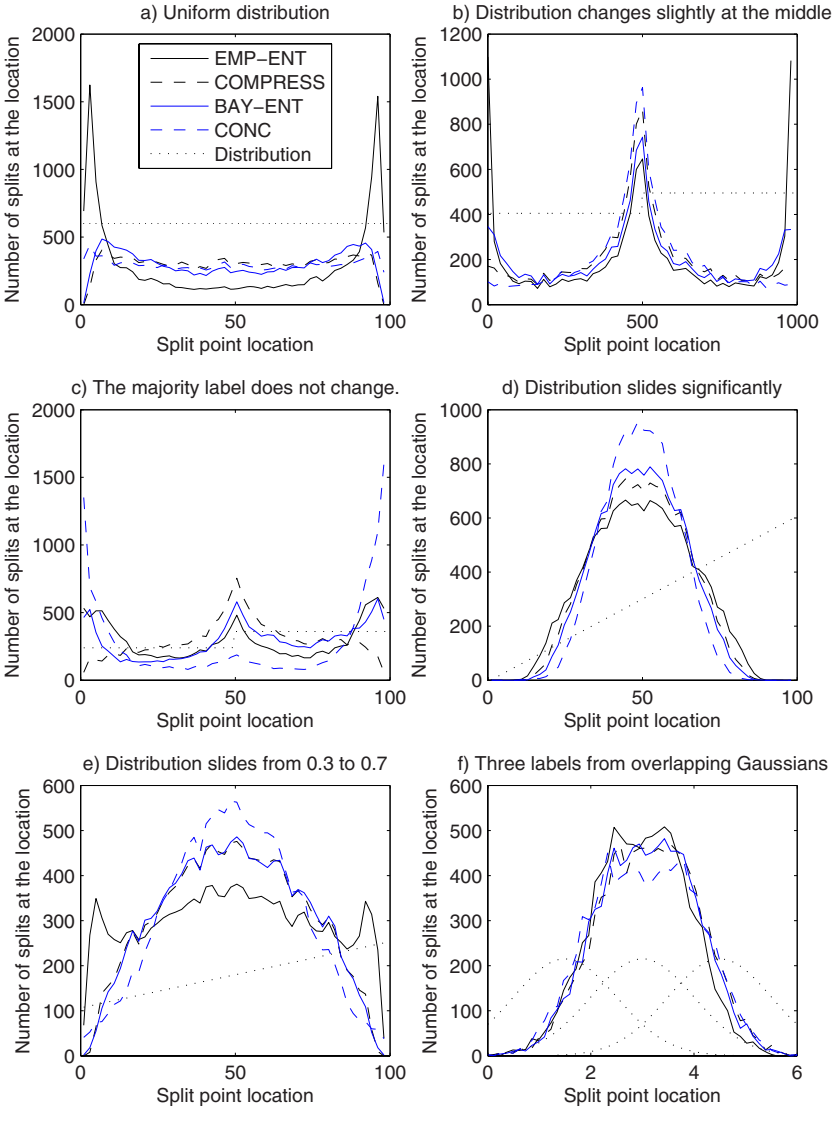


Fig. 6. Comparison of the methods for six different distributions for the interval. In a)-e) the dotted black line is the probability for label a, and in f) the dotted line gives the generating Gaussians for three different labels.

In test a) the intervals are generated from a uniform distribution. Hence, there is no correct place to split, because all subintervals are similar. In the figure we see how EMP-ENT prefers short intervals, and how CONC resists this behavior.

In test b) the generating distribution changes at the middle, first giving probability 0.47 and then 0.53 to the label a. The interval length is 1 000 items.

Although b) has a long interval length, the fact that the distribution changes only slightly, causes EMP-ENT to choose short intervals.

In test c) the distribution also changes at the middle point, but in contrast to test b) the majority label does not. The probability of the label a is first 0.2 and 0.3 after the middle point. Note how CONC fails in this case, although it otherwise performs well. Interestingly, COMPRESS is the only approach to perform well.

In tests d) and e) the generating distribution changes as a linear function of the interval position. In d) the change for the label a is from 0 to 1. In e) the change is from 0.3 to 0.7. The correct split position is in the middle. We see that the more difficult decision in e) causes EMP-ENT to choose also short intervals.

Finally, test f) demonstrates intervals generated from three overlapping Gaussians. Each label — a, b, and c — is generated from a Gaussian distribution with variance one on the unit line. The centers are at 1.5, 3, and 4.5. The correct split position is at 3.

Of course, these results do not guarantee similar performance in a setting that differs significantly from those introduced here. However, we chose these tests to demonstrate the performance in as different circumstances as we could find. They imply that failures with EMP-ENT tend to happen, if the decision is not easy to begin with. Otherwise, the failure rate does not depend on the parameters of the test, such as length of the interval.

6 Implications to Classification: Experimental Evaluation

In this section we study how the suggested solutions affect our motivating application, the NB classifier. We evaluate the splitting methods on 16 commonly used problems from the UCI machine learning repository. We carry out 100 iterations for each problem. During each iteration two-thirds of the data is assigned to a training set and the rest is assigned to a test set. The performances as a probability of predicting the correct label for the test sets are given in Table 4. We also give the number of generated bins for each problem, counted over all features. The BAY-ENT method is omitted, because its performance was close to that of COMPRESS.

We note that the performance difference between EMP-ENT and COMPRESS is negligible. Either the problematic cases that we have discussed do not occur in practice, or due to the nature of either MDL stopping rule or the recursive heuristic the mistakes are not important. In the latter case it could be that MDL is too conservative in its decisions to take advantage in a better splitting, because it maximizes the split probability when using EMP-ENT, as we have noted in [23]. Another potential reason is the use of the recursive heuristic which could hide the problematic decisions. By this we mean that, even if we erroneously split few labels away from an interval, the recursive splitting guarantees that we will also split at the correct place.

Also, CONC performed well, expect in the Bupa domain. Hence, the behavior depicted in test c) of Figure 6 appears not to happen frequently.

Table 4. Performance of splitting methods on Naïve Bayes. The average classification accuracy is over 100 repetitions of randomized training set selection for 16 UCI domains. The average number of bins in each domain is also given.

	Accuracy			Number of bins		
	EMP-ENT	**COMP**	**CONC**	**EMP-ENT**	**COMP**	**CONC**
Iris	94.0	94.0	93.9	6.6	6.6	6.7
Glass	67.8	66.4	67.4	14.4	14.6	13.6
Bupa	62.8	63.5	59.4	6.2	6.3	6.1
Pima	74.0	74.4	74.0	10.3	10.2	11.2
Ecoli	85.1	85.3	85.6	8.2	7.8	7.8
Segmentation	83.2	82.7	82.4	44.4	44.6	44.1
Wine	98.3	98.0	97.3	19.8	19.8	18.9
Australian	85.6	85.5	85.8	14.4	14.4	14.8
German	73.7	73.2	73.8	24.1	24.0	24.9
Iono	88.9	88.5	89.3	88.1	87.5	81.0
Sonar	75.9	76.5	75.9	60.9	59.8	60.5
Wisconsin	97.5	97.5	97.7	17.5	17.6	18.2
Letter	73.7	73.6	73.4	128.8	129.4	129.3
Abalone	58.5	58.5	58.5	41.3	40.9	37.6
Vehicle	59.4	59.1	59.2	44.1	42.6	42.6
Page	93.3	93.2	93.5	46.9	46.1	40.8
Average	79.5	79.4	79.2	36.0	35.8	34.9

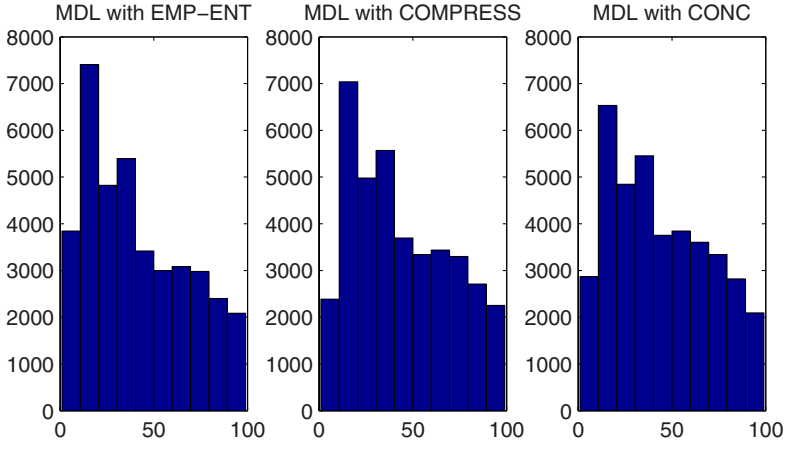


Fig. 7. Distribution of intervals with length less than 100 counted over all domains

We also investigated further what kind of intervals the splitting rules form. The results are given in Figure 7. We see that the EMP-ENT indeed chooses more short intervals than COMPRESS. On average, the number of intervals that COMPRESS selects is slightly lower than for EMP-ENT. This is not surprising, because EMP-ENT maximizes the likelihood of splitting with MDL stopping rule.

7 Conclusions

We gave observations on the behavior of the empirical entropy and noted that for small sample sizes its bias is significant. Then we suggested new methods for choosing the best split point and we empirically evaluated them. A method based on compression fared well in these tests, although the implications were negligible when the splitting methods were applied in the NB classifier. One reason for this behavior could be that the stopping rule and the split point selection method interact in the recursive heuristic. Hence, one possible future direction is to investigate these interactions further.

Acknowledgments

This work has been financially supported by Academy of Finland projects ALEA (210795) and "Machine learning and online data structures" (119699).

References

1. Catlett, J.: On changing continuous attributes into ordered discrete attributes. In: Kodratoff, Y. (ed.) EWSL 1991. LNCS, vol. 482, pp. 164–178. Springer, Heidelberg (1991)
2. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)
3. Raileanu, L.E., Stoffel, K.: Theoretical comparison between the gini index and information gain criteria. Annals of Mathematics and Artificial Intelligence 41, 77–93 (2004)
4. Kohavi, R., Sahami, M.: Error-based and entropy-based discretization of continuous features. In: Simoudis, E., Han, J.W., Fayyad, U. (eds.) Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 114–119. AAAI Press, Menlo Park (1996)
5. Hand, D.J., Yu, K.: Idiot Bayes? not so stupid after all. International Statistical Review 69, 385–398 (2001)
6. Friedman, J.H.: On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery 1(1), 55–77 (1997)
7. Domingos, P., Pazzani, M.J.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning 29, 103–130 (1997)
8. Bouckaert, R.R.: Naive Bayes classifiers that perform well with continuous variables. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 1089–1094. Springer, Heidelberg (2004)
9. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Prieditis, A., Russell, S. (eds.) Proceedings of the Twelfth International Conference on Machine Learning, pp. 194–202. Morgan Kaufmann, San Francisco (1995)
10. Elomaa, T., Rousu, J.: Fast minimum training error discretization. In: Sammut, C., Hoffmann, A.G. (eds.) Machine Learning, Proceedings of the Nineteenth International Conference, pp. 131–138. Morgan Kaufmann, San Francisco (2002)

11. Paninski, L.: Estimation of entropy and mutual information. Neural Computation 15, 1191–1253 (2003)
12. Bialek, W., Nemenman, I. (eds.): Estimation of Entropy and Information of Undersampled Probability Distributions – Theory, Algorithms, and Applications to the Neural Code. Satellite of the Neural Information Processing Systems Conference (NIPS 2003) (2003)
13. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Saitta, L. (ed.) Machine Learning, Proceedings of the Thirteenth International Conference, pp. 275–283. Morgan Kaufmann, San Francisco (1996)
14. Domingos, P.: A unified bias-variance decomposition for zero-one and squared loss. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 564–569. MIT Press, Cambridge (2000)
15. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons, New York (1991)
16. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge (2007)
17. Kononenko, I.: On biases in estimating multi-valued attributes. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pp. 1034–1040. Morgan Kaufmann, San Francisco (1995)
18. Wilks, S.S.: Mathematical Statistics. John Wiley & Sons, New York (1962)
19. Zhu, M., Lu, A.Y.: The counter-intuitive non-informative prior for the Bernoulli family. Journal of Statistics Education 12 (2004)
20. Kass, R.E., Wasserman, L.: The selection of prior distributions by formal rules. Journal of the American Statistical Association 91, 1343–1370 (1996)
21. Gelman, A.: Prior distribution. In: Encyclopedia Environmetrics, vol. 3, pp. 1634–1637. John Wiley & Sons, Chichester (2002)
22. Kearns, M.J., Mansour, Y.: On the boosting ability of top-down decision tree learning algorithms. Journal of Computer and System Sciences 58, 109–128 (1999)
23. Kujala, J., Elomaa, T.: Improved algorithms for univariate discretization of continuous features. In: Kok, J.N., Koronacki, J., López de Mántaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 188–199. Springer, Heidelberg (2007)

Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs

Lior Kuyer[1], Shimon Whiteson[1], Bram Bakker[1], and Nikos Vlassis[2]

[1] Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{lkuijer,whiteson,bram}@science.uva.nl
[2] Department of Production Engineering and Management
Technical University of Crete, Chania, Greece
vlassis@dpem.tuc.gr

Abstract. Since traffic jams are ubiquitous in the modern world, optimizing the behavior of traffic lights for efficient traffic flow is a critically important goal. Though most current traffic lights use simple heuristic protocols, more efficient controllers can be discovered automatically via multiagent reinforcement learning, where each agent controls a single traffic light. However, in previous work on this approach, agents select only locally optimal actions without coordinating their behavior. This paper extends this approach to include explicit coordination between neighboring traffic lights. Coordination is achieved using the max-plus algorithm, which estimates the optimal joint action by sending locally optimized messages among connected agents. This paper presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. It also provides empirical evidence that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Furthermore, it provides a new understanding of the properties a traffic network must have for such coordination to be beneficial and shows that max-plus outperforms previous methods on networks that possess those properties.

Keywords: multiagent systems, reinforcement learning, coordination graphs, max-plus, traffic control.

1 Introduction

Traffic jams are ubiquitous in the modern world and are getting worse, due to rapidly increasing populations and vehicle usage rates. They commonly occur in urban settings, where traffic lights are the most typical control mechanism. Existing road infrastructure is often strained to its limits and expansion is infeasible due to spatial, environmental, and economic constraints. Therefore, optimizing the behavior of traffic lights for efficient traffic flow is a critically important goal.

In practice, most traffic lights use very simple protocols that merely alternate red and green lights for fixed intervals. The interval lengths may change during peak hours but are not otherwise optimized. Since such controllers are far

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 656–671, 2008.

from optimal, several researchers have investigated the application of machine learning to automatically discover more efficient controllers. The methods employed include fuzzy logic [5], neural networks [12] and evolutionary algorithms [7]. These methods perform well but can only handle networks with a relatively small number of controllers.

Since traffic control is fundamentally a problem of sequential decision making, it is perhaps best suited to the framework of reinforcement learning, in which an agent learns from trial and error via direct interaction with its environment. Each action results in immediate rewards and new observations about the state of the world. Over time, the agent learns a control policy that maximizes the expected long-term reward it receives.

One way to apply reinforcement learning to traffic control is to train a single agent to control the entire system, i.e. to determine how every traffic light in the network is set at each timestep. However, such centralized controllers scale very poorly, since the size of the agent's action set is exponential in the number of traffic lights.

An alternative approach is to view the problem as a multiagent system where each agent controls a single traffic light [3,14]. Since each agent observes only its local environment and selects only among actions related to one traffic light, this approach can scale to large numbers of agents. The primary limitation is that the individual agents do not coordinate their behavior. Consequently, agents may select individual actions that are locally optimal but that together result in global inefficiencies.

This paper extends the reinforcement learning approach to traffic control by using cooperative learning and explicit coordination among agents. We make the relaxing assumption that an agent is affected only by those agents with a direct influence on its environment, i.e. its neighbors in the network. Under this assumption, the global coordination problem may be decomposed into a set of local coordination problems and can be solved with the use of *coordination graphs* [9].

Since the system must perform under time constraints, an efficient method for finding optimal joint actions in such graphs is required. For this reason we apply *max-plus* [10], which estimates the optimal joint action by sending locally optimized messages among connected agents. It also allows the agents to report their current best action at any time (even if the action found so far may be suboptimal).

This paper makes several contributions. First, it augments the existing framework of reinforcement learning for traffic control by allowing scalable coordination of neighboring traffic lights. Second, it presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. Third, it provides empirical evidence that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Fourth, it provides a new understanding of the properties a traffic network must have for such coordination to be beneficial and shows that max-plus outperforms previous methods on networks that possess those properties.

The remainder of this paper is organized as follows. Section 2 introduces the traffic model used in our experiments. Section 3 describes the traffic control problem as a reinforcement learning task. Section 4 describes coordination graphs and the max-plus algorithm and Section 5 describes how max-plus is applied to the traffic control problem. Section 6 presents experimental results and Section 7 discusses these results. Section 8 outlines directions for future work and Section 9 concludes.

2 Traffic Model

All experiments presented in this paper were conducted using The Green Light District (GLD) traffic simulator[1] [3,14]. GLD is a *microscopic* traffic model, i.e. it simulates each vehicle individually, instead of simply modeling aggregate properties of traffic flow. The dynamic variables of the model represent microscopic properties such as the position and velocity of each vehicle. Vehicles move through the network according to their physical characteristics (e.g. length, speed, etc.), fundamental rules of motion, and predefined rules of driver behavior. GLD's simulation is based on *cellular automata*, in which discrete, partially connected cells can occupy various states. For example, a road cell can be occupied by a vehicle or be empty. Local transition rules determine the dynamics of the system and even simple rules can lead to a highly dynamic system.

The GLD infrastructure consists of roads and nodes. A road connects two nodes, and can have several lanes in each direction. The length of each road is expressed in cells. A node is either an intersection where traffic lights are operational or an edge node. There are two types of agents that occupy such an infrastructure: vehicles and traffic lights (or intersections). All agents act autonomously and are updated every timestep. Vehicles enter the network at edge nodes and each edge node has a certain probability of generating a vehicle at each timestep. Each generated vehicle is assigned one of the other edge nodes as a destination. The distribution of destinations for each edge node can be adjusted.

There are several types of vehicles, defined by their speed, length, and number of passengers. In this paper, all vehicles have equal length and an equal number of passengers. The state of each vehicle is updated every timestep. It either moves the distance determined by its speed and the state around it (e.g. vehicles in front may limit how far it can travel) or remains in the same position (e.g. the next position is occupied or a traffic light prevents its lane from moving).

When a vehicle crosses an intersection, its driving policy determines which lane it goes to next. Once a lane is selected, the vehicle cannot switch to a different lane. For each intersection, there are several light configurations that are safe. At each timestep, the intersection must choose from among these configurations, given the current state.

Figure 1 shows an example GLD intersection. It has four roads, each consisting of four lanes (two in each direction). Vehicles occupy n cells of a lane, depending

[1] Available at http://sourceforge.net/projects/stoplicht

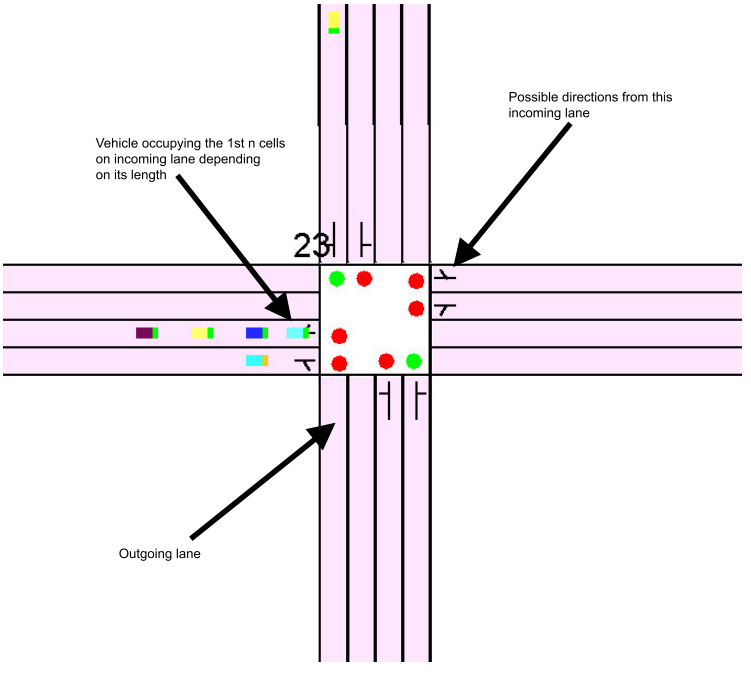


Fig. 1. An example GLD intersection

on their length. Traffic on a given lane can only travel in the directions allowed on that lane. This determines the possible safe light configurations. For example, the figure shows a lane where traffic is only allowed to travel straight or right.

The behavior of each vehicle depends on how it selects a path to its destination node and how it adjusts its speed over time. In our experiments, the vehicles always select the shortest path to their destination node. In previous work [3,14], vehicles always traveled at constant speed and only a single vehicle could cross an intersection at each timestep. We extend the simulator to allow more dynamic behavior. Three speeds (2, 4, or 6 cells per timestep) are now possible. Vehicles enter the network with a speed of 4 and at each timestep there is a 78% chance the vehicle will keep its current speed when it is 4 and an 88% chance when it is either 2 or 6. Furthermore, multiple vehicles from a single lane can now cross an intersection during each timestep. The number depends on the vehicles' speed and on the state of the destination lanes.

3 Reinforcement Learning for Urban Traffic Control

Several techniques for learning traffic controllers with model-free reinforcement learning methods like Sarsa [13] or Q-learning [1,11] have previously been developed. However, they all suffer from the same problem: they do not scale to

large networks since the size of the state space grows rapidly. Hence, they are either applied only to small networks or are used to train homogeneous controllers (by training on a single isolated intersection and copying the result to each intersection in the network).

A more tractable approach is to use model-based reinforcement learning, in which the transition and reward functions are estimated from experience and then used to find a policy via planning methods like *dynamic programming* [4]. A full transition function would have to map the location of every vehicle in the system at one timestep to the location of every vehicle at the next timestep. Doing so is clearly infeasible, but learning a model is nonetheless possible if a *vehicle-based* representation [14] is used. In this approach, the global state is decomposed into local states based on each individual vehicle. The transition function maps one vehicle's location at a given timestep to its location at the next timestep. As a result, the number of states grows linearly in the number of cells and can scale to much larger networks. Furthermore, the transition function can generalize from experience gathered in different locations, rather than having to learn separate mappings for each location.

To represent the model, we need only keep track of the number of times each transition (s, a, s') has occurred and each state-action pair (s, a) has been reached. The transition model can then be estimated via the maximum likelihood probability $\frac{|(s,a,s')|}{|(s,a)|}$. Hence, each timestep produces new data which is used to update the model. Every time the model changes, the value function computed via dynamic programming must be updated too. However, rather than having to update each state, we can update only the states most likely to be affected by the new data, using an approach based on *prioritized sweeping* [2]. The remainder of this section describes the process of learning the model in more detail.

Given a vehicle-based representation, the traffic control problem consists of the following components:

- $s \in S$: the fully observable global state
- $i \in I$: an intersection controller
- $a \in A$: an action, which consists of setting to green a subset of the traffic lights at the intersection; $A_i \subseteq A$ is the subset of actions that are safe at intersection i
- $l \in L$: a traffic lane; $L_i \subseteq L$ is the subset of incoming lanes for intersection i
- $p \in P$: a position; $P_l \subseteq P$ is the subset of positions for lane l

The global transition model is $P(s'|s, a)$ and the global state s decomposes into a vector of local states, $s = \langle s_{p_{l_i}} \rangle$, with one for each position in the network. The action-value function decomposes as:

$$Q(s, a) = \sum_i Q_i(s_i, a_i) \tag{1}$$

where

$$Q_i(s_i, a_i) = \sum_{l_i} \sum_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i) \tag{2}$$

The vehicle-based update rule is then given by:

$$Q_{p_{l_i}}(s_{p_{l_i}}, a_i) := \sum_{s'_{p_{l_i}} \in S'} P(s'_{p_{l_i}} | a_i, s_{p_{l_i}})[r(s_{p_{l_i}}, s'_{p_{l_i}}) + \gamma V(s'_{p_{l_i}})] \tag{3}$$

where S' are all possible states that can be reached from $s_{p_{l_i}}$ given the current traffic situation and the vehicle's properties (e.g. its speed and length). Like Wiering, we use a learning rate $\gamma = 0.9$. $V(s_{p_{l_i}})$ estimates the expected waiting time at p_{l_i} and is given by:

$$V(s_{p_{l_i}}) := \sum_{a_i} P(a_i | s_{p_{l_i}}) Q(s_{p_{l_i}}, a_i) \tag{4}$$

The transition model can be estimated using maximum likelihoods by counting state transitions and corresponding actions. The update is given by:

$$P(s'_{p_{l_i}} | s_{p_{l_i}}, a_i) := \frac{C(s_{p_{l_i}}, a_i, s'_{p_{l_{ij}}})}{C(s_{p_{l_i}}, a_i)} \tag{5}$$

where $C(\cdot)$ is a function that counts the number of times the event occurs. To estimate $V(s_{p_{l_i}})$, we also need to estimate the probability that a certain action will be taken given the state, which is done using the following update:

$$P(a_i | s_{p_{l_i}}) := \frac{C(s_{p_{l_i}}, a_i)}{C(s_{p_{l_i}})} \tag{6}$$

The global reward function decomposes as:

$$r(s, s') = \sum_i \sum_{l_i} \sum_{p_{l_i}} r(s_{p_{l_i}}, s'_{p_{l_i}}) \tag{7}$$

and

$$r(s_{p_{l_i}}, s'_{p_{l_i}}) = \begin{cases} 0 & s_{p_{l_i}} \neq s'_{p_{l_i}} \\ -1 \text{ otherwise} \end{cases} \tag{8}$$

Given the current model, the optimal value function is estimated using dynamic programming with a fixed number of iterations. Wiering [14] performs only one iteration per timestep and uses ϵ-greedy exploration to ensure the estimated model obtains sufficiently diverse data.

Bakker et al. [3] extend Wiering's approach by including congestion information in the state representation. The value function $Q_{p_{l_i}}(s_{p_{l_i}}, a_i)$ is extended to $Q_{p_{l_i}}(s_{p_{l_i}}, c_{dest}, a_i)$ where $c_{dest} \in \{0, 1\}$ is a single bit indicating the congestion level at the next lane for the vehicle currently at p_{l_i}. If the congestion at the next lane exceeds some threshold then $c_{dest} = 1$ and otherwise it is set to 0. This extension allows the agents to learn different state transition probabilities and value functions when the outbound lanes are congested. This method has been shown to outperform Wiering's approach on a saturated network. The cost of including such congestion information is a larger state space and potentially

slower learning. It also requires the vehicles to communicate with the controllers, since the latter need to know the destination lanes of each vehicle.

4 Coordination Graphs

The primary limitation of the approaches developed by Wiering and Bakker et al. is that the individual agents do not coordinate their behavior. Consequently, agents may select individual actions that are locally optimal but that together result in global inefficiencies. Coordinating actions can be difficult since the size of the joint action space is exponential in the number of agents. However, in many cases, the best action for a given agent may depend on only a small subset of the other agents. If so, the global reward function can be decomposed into local functions involving only subsets of agents. The optimal joint action can then be estimated by finding the joint action that maximizes the sum of the local rewards.

A *coordination graph* [9], which can be used to describe the dependencies between agents, is an undirected graph $G = (V, E)$ in which each node $i \in V$ represents an agent and each edge $e(i, j) \in E$ between agents i and j indicates a dependency between them. The global coordination problem is then decomposed into a set of local coordination problems, each involving a subset of the agents. Since any arbitrary graph can be converted to one with only pairwise dependencies [16], the global action-value function can be decomposed into pairwise value functions given by:

$$Q(s, a) = \sum_{i,j \in E} Q_{ij}(s, a_i, a_j) \tag{9}$$

where a_i and a_j are the corresponding actions of agents i and j, respectively. Using such a decomposition, the *variable elimination* [9] algorithm can compute the optimal joint action by iteratively eliminating agents and creating new conditional functions that compute the maximal value the agent can achieve given the actions of the other agents on which it depends. Although this algorithm always finds the optimal joint action, it is computationally expensive, as the execution time is exponential in the induced width of the graph [15]. Furthermore, the actions are known only when the entire computation completes, which can be a problem for systems that must perform under time constraints. In such cases, it is desirable to have an *anytime* algorithm that improves its solution gradually.

One such algorithm is *max-plus* [8,10], which approximates the optimal joint action by iteratively sending locally optimized messages between connected nodes in the graph. While in state s, a message from agent i to neighboring agent j describes a local reward function for agent j and is defined by:

$$\mu_{ij}(a_j) = \max_{a_i} \{Q_{ij}(s, a_i, a_j) + \sum_{k \in \Gamma(i) \backslash j} \mu_{ki}(a_i)\} + c_{ij} \tag{10}$$

where $\Gamma(i) \backslash j$ denotes all neighbors of i except for j and c_{ij} is either zero or can be used to normalize the messages. The message approximates the maximum value

agent i can achieve for each action of agent j based on the function defined between them and incoming messages to agent i from other connected agents (except j). Once the algorithm converges or time runs out, each agent i can select the action

$$a_i^* = \arg\max_{a_i} \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) \tag{11}$$

Max-plus has been proven to converge to the optimal action in finite iterations, but only for tree-structured graphs, not those with cycles. Nevertheless, the algorithm has been successfully applied to such graphs [6,10,16].

5 Max-Plus for Urban Traffic Control

Max-plus enables agents to coordinate their actions and learn cooperatively. Doing so can increase robustness, as the system can become unstable and inconsistent when agents do not coordinate. By exploiting coordination graphs, max-plus minimizes the expense of computing joint actions and allows them to be approximated within time constraints.

In this paper, we combine max-plus with Wiering's model-based approach to traffic control. We use the vehicle-based representation defined in Section 3 but add dependence relationships between certain agents. If $i, j \in J$ are two intersections connected by a road, then they become neighbors in the coordination graph, i.e. $i \in \Gamma(j)$ and $j \in \Gamma(i)$. The local value functions are:

$$Q_i(s_i, a_i, a_j) = \sum_{l_i} \sum_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i, a_j) \tag{12}$$

Using the above, we can define the pairwise value functions used by max-plus:

$$Q_{ij}(s, a_i, a_j) = \sum_{p_{l_i}} O_{p_{l_i}} Q_{p_{l_i}}(s_{p_{l_i}}, a_i, a_j) + \sum_{p_{l_j}} O_{p_{l_j}} Q_{p_{l_j}}(s_{p_{l_j}}, a_j, a_i) \tag{13}$$

where $O_{p_{l_i}}$ is a binary operator which indicates occupancy at p_{l_i}:

$$O_{p_{l_i}} = \begin{cases} 0 \; p_{l_i} \text{ not occupied} \\ 1 \text{ otherwise} \end{cases} \tag{14}$$

These local functions are plugged directly into Equation 10 to implement max-plus. Note that the functions are symmetric such that $Q_{ij}(s, a_i, a_j) = Q_{ji}(s, a_j, a_i)$. Thus, using Equation 13, the joint action can be estimated directly by the max-plus algorithm. Like Wiering, we use one iteration of dynamic programming per timestep and ϵ-greedy exploration. We also limit max-plus to 3 iterations per timestep.

Note that there are two levels of value propagation among agents. On the lower level, the vehicle-based representation enables estimated values to be propagated between neighboring agents and eventually through the entire network, as in

Wiering's approach. On the higher level, agents use max-plus when computing joint actions to inform their neighbors of the best value they can achieve, given the current state and the values received from other agents.

Using this approach, agents can learn cooperative behavior, since they share value functions with their neighbors. Furthermore, they can do so efficiently, since the number of value functions is linear in the induced width of the graph. Stronger dependence relationships could also be modeled, i.e. between intersections not directly connected by a road, but we make the simplifying assumption that it is sufficient to model the dependencies between immediate neighbors in the traffic network.

6 Results

In this section, we compare the novel approach described in Section 5 to the TC-1 (Traffic Controller 1) developed by Wiering [14] and the TC-SBC (Traffic Controller with State Bit for Congestion) extension of Bakker et al. [3]. Wiering compared TC-1 to two heuristic strategies, one that always sets the lights at each intersection to maximize throughput and another that always gives right-of-way to the longest queue. These heuristics perform well in light traffic but TC-1 substantially outperforms them in heavy traffic. Therefore, we focus our experiments on comparisons between the novel method, TC-1, and TC-SBC in highly saturated conditions.

These experiments are designed to test the hypothesis that, under highly saturated conditions, coordination is beneficial when the amount of *local traffic* is small. Local traffic consists of vehicles that cross a single intersection and then exit the network, thereby interacting with just one learning agent. If this hypothesis is correct, coordinated learning with max-plus should substantially outperform TC-1 and TC-SBC when most vehicles pass through multiple intersections.

In particular, we consider three different scenarios. In the *baseline* scenario, the traffic network includes routes, i.e. paths from one edge node to another, that cross only a single intersection. Since each vehicle's destination is chosen from a uniform distribution, there is a substantial amount of local traffic. In the *nonuniform destinations* scenario, the same network is used but destinations are selected to ensure that each vehicle crosses two or more intersections, thereby eliminating local traffic. To ensure that any performance differences we observe are due to the absence of local traffic and not just to a lack of uniform destinations, we also consider the *long routes* scenario. In this case, destinations are selected uniformly but the network is altered such that all routes contain at least two intersections, again eliminating local traffic.

While a small amount of local traffic will occur in real-world scenarios, the vast majority is likely to be non-local. Thus, the baseline scenario is used, not for its realism, but to help isolate the effect of local traffic on each method's performance. The nonuniform destinations and long routes scenarios are more

challenging and realistic, as they require the methods to cope with an abundance of non-local traffic.

We present initial proof-of-concept results in small networks and then study the same three scenarios in larger networks to show that the max-plus approach scales well and that the qualitative differences between the methods are the same in more realistic scenarios.

For each case, we consider three different metrics: 1) *average trip waiting time* (ATWT): the total waiting time of all vehicles that have reached their destination divided by the number of such vehicles, 2) *ratio of stopped vehicles* (RSV): the fraction of all vehicles in the network that do not move in a given timestep, and 3) *total queue length* (TQL): the number of vehicles that have been generated but are still waiting to enter the network because the outbound lane of their edge node is full.

Due to lack of space, we present graphs only for ATWT. In most cases, ATWT is sufficient to determine the methods' relative performance, i.e. lower ATWT implies lower RSV and TQL. Therefore, we mention the RSV and TQL results only when they are qualitatively different from ATWT. All results are averaged over 10 independent runs.

6.1 Small Networks

Figure 2 shows the small network used for the baseline and nonuniform destinations scenarios. Each intersection allows traffic to cross from only one direction at a time. All lanes have equal length and all edge nodes have equal spawning rates (vehicles are generated with probability 0.2 per timestep). The left side of Figure 3 shows results from the baseline scenario, which have uniform destinations. As a result, much of the traffic is local and hence there is no significant performance difference between TC-1 and max-plus. TC-SBC performs worse than the other methods, which is likely due to slower learning as a result of a larger state space.

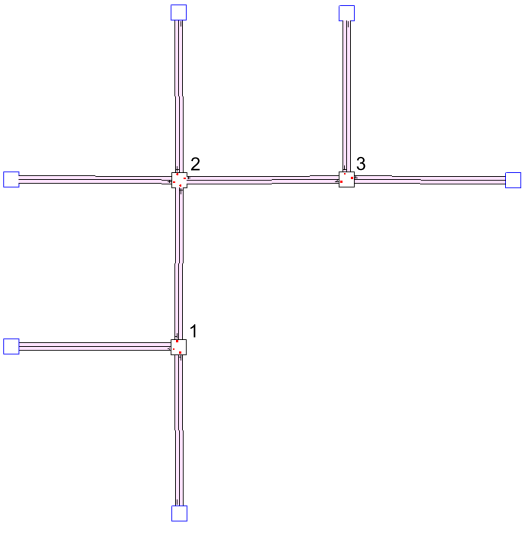


Fig. 2. The small network used in the baseline and nonuniform destinations scenarios

The right side of Figure 3 shows results from the nonuniform destinations scenario. In this case, all traffic from intersections 1 and 3 is directed to intersection 2. Traffic from the top edge node of intersection 2 is directed to intersection 1 and

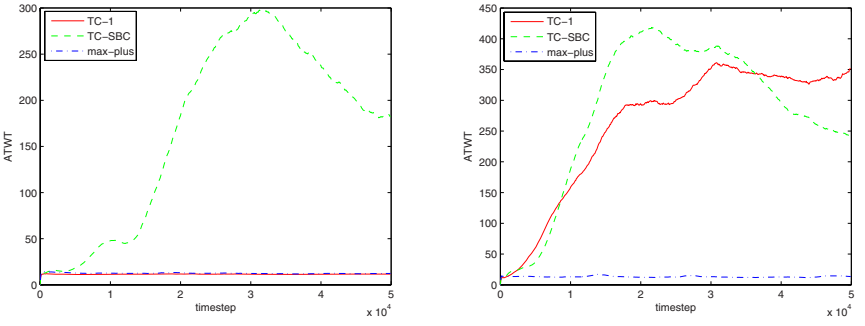


Fig. 3. Average ATWT per timestep for each method in the small network for the baseline (left) and nonuniform destinations (right) scenarios

traffic from the left edge node is directed to intersection 3. Consequently, there is no local traffic. This results in a dramatic performance difference between max-plus and the other two methods.

This result is not surprising since the lack of uniform destinations creates a clear incentive for the intersections to coordinate their actions. For example, the lane from intersection 1 to 2 is likely to become saturated, as all traffic from edge nodes connected to intersection 1 must travel through it. When such saturation occurs, it is important for the two intersections to coordinate, since allowing incoming traffic to cross intersection 1 is pointless unless intersection 2 allows that same traffic to cross in a "green wave".

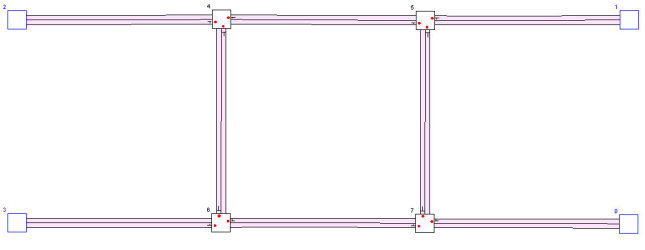


Fig. 4. The small network used in the long routes scenario

To ensure that the performance difference between the baseline and nonuniform destinations scenarios is due to the removal of local traffic and not some other effect of nonuniform destinations, we also consider the long routes scenario. Destinations are kept uniform, but the network structure is altered such that all routes involve at least two intersections. Figure 4 shows the new network, which has a fourth intersection that makes local traffic impossible. Figure 5 shows the results from this scenario.

As before, max-plus substantially outperforms the other two methods, suggesting its advantage is due to the absence of local traffic rather than other factors. TC-1 achieves a lower ATWT than TC-SBC but actually performs much worse. In fact, TC-1's joint actions are so poor that the outbound lanes of some edge nodes become full and its TQL skyrockets. As a result, the ATWT is not updated, leading to an artificially low score. At the end of each run, TC-1 had an average of TQL of 9259.7 while TC-SBC had only 3966.9 and max-plus had 0.0. The low quality of TC-1's joint actions becomes clear when comparing the RSV: 0.92 for TC-1, 0.55 for TC-SBC, and 0.09 for max-plus.

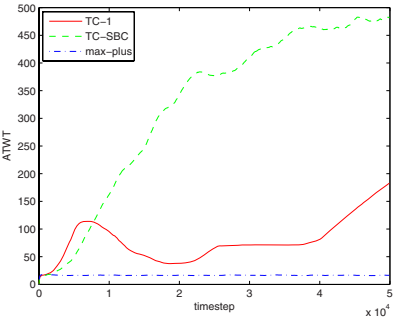


Fig. 5. Average ATWT per timestep in the small network for the long routes scenario

6.2 Large Networks

We also consider the same three scenarios in larger networks to show that the max-plus approach scales well and that the qualitative differences between the methods are the same in more realistic scenarios. Figure 6 shows the network used for the baseline and nonuniform destinations scenarios. It includes 15 agents and roads with four lanes. The left side of Figure 7 shows results from the baseline scenario, which has uniform destinations. As with the smaller network, max-plus and TC-1 perform very similarly in this scenario, though max-plus's coordination results in slightly slower learning. However, TC-SBC no longer performs worse than the other two methods, probably because the network is now large enough to incur substantial congestion. TC-SBC, thanks to its congestion bit, can cope with this occurrence better than TC-1.

The right side of Figure 7 shows results from the nonuniform destinations scenario. In this case, traffic from the top edge nodes travel only to the bottom edge nodes and vice versa. Similarly, traffic from the left edge nodes travel only to right edge nodes and vice versa. As a result, all local traffic is eliminated and max-plus performs much better than TC-1 and TC-SBC. TC-SBC performs substantially better than TC-1, as the value of its congestion bit is even greater in this scenario.

To implement the long routes scenario, we remove one edge node from the two intersections that have two edge nodes (the top and bottom right nodes in Figure 6). Traffic destinations are uniformly distributed but the new network structure ensures that no local traffic occurs. The results of the long routes scenario are shown in Figure 8. As before, max-plus substantially outperforms the other two methods, confirming that its advantage is due to the absence of local traffic rather than other factors.

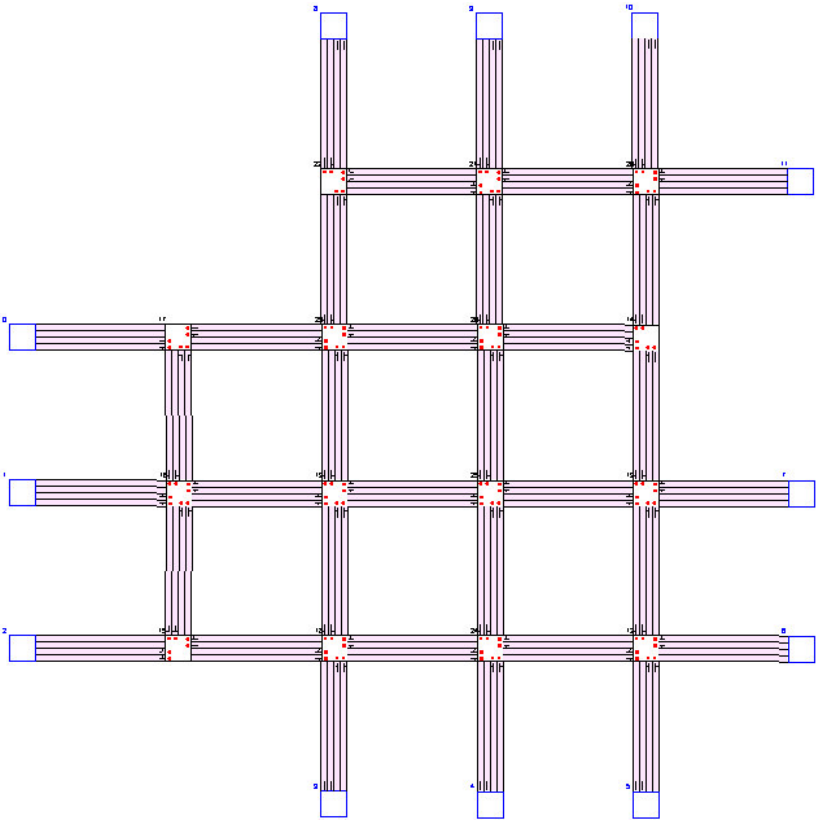


Fig. 6. The large network used in the baseline and nonuniform destinations scenarios

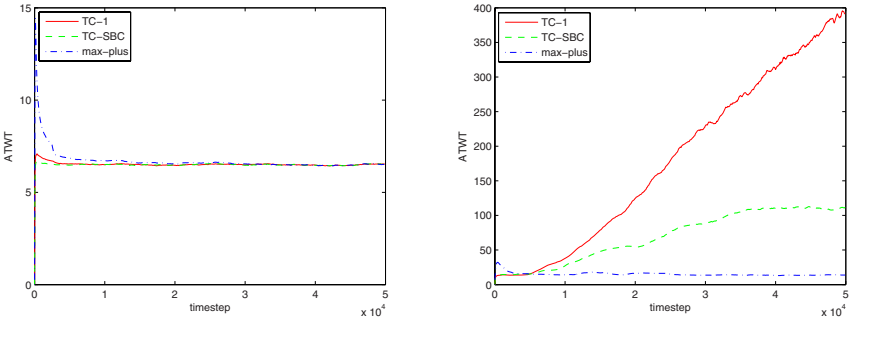


Fig. 7. Average ATWT per timestep for each method in the large network for the baseline (left) and nonuniform destinations (right) scenarios

7 Discussion

The experiments presented above demonstrate a strong correlation between the amount of local traffic and the value of coordinated learning. The max-plus method consistently outperforms both non-coordinated methods in each scenario where local traffic has been eliminated. Hence, these results help explain under what circumstances coordinated methods can be expected to perform better. More specifically, they confirm the hypothesis that, under highly saturated conditions, coordination is beneficial when the amount of local traffic is small.

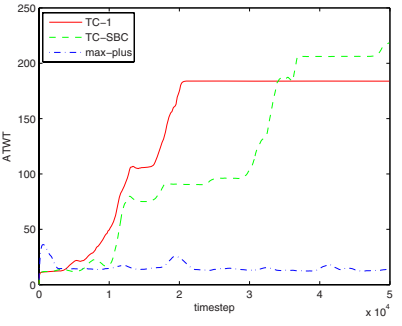


Fig. 8. Average ATWT per timestep for each method in the long routes scenario

Even when there is substantial local traffic, the max-plus method achieves the same performance as the alternatives, though it learns more slowly. Hence, this method appears to be substantially more robust, as it can perform well in a much broader range of scenarios.

By testing both small and large networks, the results also demonstrate that max-plus is practical in realistic settings. While max-plus has succeeded in small applications before [10], this paper presents its first application to a large-scale problem. In fact, in the scenarios without local traffic, the performance gap between max-plus and the other methods was consistently larger in the big networks than the small ones. In other words, as the number of agents in the system grows, the need for coordination increases. This property makes the max-plus approach particularly attractive for solving large problems with complex networks and numerous agents.

Finally, these results also provide additional confirmation that max-plus can perform well on cyclic graphs. The algorithm has been shown to converge only for tree-structured graphs, though empirical evidence suggests it also excels on small cyclic graphs [10]. The results presented in this paper show that this performance also occurs in larger graphs, even if they are not tree-structured.

8 Future Work

There are several ways in which the work presented this paper could be extended or improved. First, the algorithm could be augmented to automatically discover the best coordination graph for the problem. We currently use fixed coordination graphs with dependencies only between intersections connected by a road. In some cases, other coordination graphs could lead to better performance. Optimization methods such as genetic algorithms could potentially be used to search for the best coordination graph for a given problem.

Second, max-plus could be implemented in a distributed fashion. The current implementation is centralized and uses iterations. In each iteration, an agent sends messages to its neighbors in a predefined order. The same functionality could be achieved with a distributed implementation where each agent sends an updated message as soon as it receives a new message from a neighbor. Since messages would be sent in parallel, computational savings would occur. However, such an implementation would require the development of protocols and other functionality not presently supported by the simulator.

Third, vehicle route selection could be adapted on-line, i.e. traffic controllers and vehicles could coordinate their behavior based on real-time traffic conditions. This functionality could avoid bottlenecks by distributing traffic more wisely over the network. Such an approach may be difficult to implement in practice since it requires vehicles to cooperate and communicate with the traffic controllers. However, previous work [14] has generated promising initial results for such an approach.

Fourth, several simplifying assumptions could be removed from the simulator. The environment is currently fully observable and stationary. Communication costs are not modeled. Many real-world factors such as weather, pedestrian behavior, vehicle accidents, illegal parking, etc. are not considered. Overall, the current simulation does exhibit most of the crucial characteristics that make urban traffic control difficult. Nonetheless, future work should focus on constructing even more realistic environments and developing the algorithmic extensions necessary to tackle them.

9 Conclusions

This paper presents a novel method for learning efficient urban traffic controllers. Previous work used multiagent reinforcement learning but the agents selected only locally optimal actions without coordinating their behavior. This paper extends this approach to include explicit coordination between neighboring traffic lights. Coordination is achieved using the max-plus algorithm, which estimates the optimal joint action by sending locally optimized messages among connected agents. This paper presents the first application of max-plus to a large-scale problem and thus verifies its efficacy in realistic settings. Empirical results on both large and small traffic networks demonstrate that max-plus performs well on cyclic graphs, though it has been proven to converge only for tree-structured graphs. Furthermore, the results provide a new understanding of the properties a traffic network must have for such coordination to be beneficial and show that max-plus outperforms previous methods on networks that possess those properties.

References

1. Abdulhai, B., et al.: Reinforcement Learning for True Adaptive Traffic Signal Control. ASCE Journal of Transportation Engineering 129(3), 278–285 (2003)
2. Moore, A.W., Atkenson, C.G.: Prioritized Sweeping: Reinforcement Learning with less data and less time. Machine Learning 13, 103–130 (1993)

3. Bakker, B., Steingrover, M., Schouten, R., Nijhuis, E., Kester, L.: Cooperative multi-agent reinforcement learning of traffic lights. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720. Springer, Heidelberg (2005)
4. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton (1957)
5. Chiu, S.: Adaptive Traffic Signal Control Using Fuzzy Logic. In: Proceedings of the IEEE Intelligent Vehicles Symposium, pp. 98–107 (1992)
6. Crick, C., Pfeffer, A.: Loopy belief propagation as a basis for communication in sensor networks. In: Proceedings of Uncertainty in Artificial Intelligence (UAI) (2003)
7. Foy, M.D., Benekohal, R.F., Goldberg, D.E.: Signal timing determination using genetic algorithms. Transportation Research Record No. 1365, 108–115
8. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 498–519 (2001)
9. Guestrin, C., Lagoudakis, M.G., Parr, R.: Coordinated reinforcement learning. In: Proceedings Nineteenth International Conference on Machine Learning, pp. 227–234 (2002)
10. Kok, J.R., Vlassis, N.: Collaborative Multiagent Reinforcement Learning by Payoff Propagation. J. Mach. Learn. Res. 7, 1789–1828 (2006)
11. Shoufeng, M., et al.: Agent-based learning control method for urban traffic signal of single intersection. Journal of Systems Engineering 17(6), 526–530 (2002)
12. Spall, J.C., Chin, D.C.: Traffic-Responsive Signal Timing for System-wide Traffic Control. Transportation Research Part C: Emerging Technologies 5(3), 153–163 (1997)
13. Thorpe, T.L., Andersson, C.: Traffic light control using sarsa with three state representations. Technical report, IBM corporation (1996)
14. Wiering, M.: Multi-Agent Reinforcement Learning for Traffic Light Control. In: Proc. 17th International Conf. on Machine Learning, pp. 1151–1158 (2000)
15. Vlassis, N.: A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. Synthesis Lectures in Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Francisco (2007)
16. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. In: Exploring Artificial Intelligence in the New Millennium, ch. 8, pp. 239–269 (2003)
17. Zhang, N.L., Poole, D.: Exploiting causal independence in Bayesian network inference. Journal of Artificial Intelligence Research 5, 301–328 (1996)

StreamKrimp: Detecting Change in Data Streams

Matthijs van Leeuwen and Arno Siebes

Department of Computer Science
Universiteit Utrecht
{mleeuwen,arno}@cs.uu.nl

Abstract. Data streams are ubiquitous. Examples range from sensor networks to financial transactions and website logs. In fact, even market basket data can be seen as a stream of sales. Detecting changes in the distribution a stream is sampled from is one of the most challenging problems in stream mining, as only limited storage can be used. In this paper we analyse this problem for streams of transaction data from an MDL perspective. Based on this analysis we introduce the StreamKrimp algorithm, which uses the Krimp algorithm to characterise probability distributions with code tables. With these code tables, StreamKrimp partitions the stream into a sequence of substreams. Each switch of code table indicates a change in the underlying distribution. Experiments on both real and artificial streams show that StreamKrimp detects the changes while using only a very limited amount of data storage.

1 Introduction

Data streams are rapidly becoming a dominant data type or data source. Common examples of streams are sensor data, financial transactions, network traffic and website logs. Actually, it would also be appropriate to regard supermarket basket data as a stream, as this is often a – seemingly never-ending – flow of transactions.

Detecting change in streams has traditionally attracted a lot of attention [1,7,10,13], both because it has many possible applications and because it is a hard problem. In the financial world, for example, quick reactions to changes in the market are paramount. Supermarkets and on-line stores have to respond quickly to changing interests of their customers. As a final example, web hosts have to respond to changes in the way users use their websites.

The unbounded growth of a stream causes the biggest challenges in stream mining: after all, only limited storage and computational capacity is available. To address this, many existing algorithms use a sliding window [1,7,10]. The problem with this approach is that often a fixed window size has to be set in advance, which strongly influences the results. Some algorithms avoid this by using an adaptive window size [16]. Many current methods focus on single-dimensional item streams or multi-dimensional real-valued streams [1,2,10,11,13].

In this paper, we address the problem of detecting change in what we call *data streams*, that is, streams of transactions. A change in such a stream of transactions is a change in the distribution the transactions are sampled from. So, given a data stream,

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part I, LNAI 5211, pp. 672–687, 2008.

we would like to identify, on-line, a series of consecutive substreams that have different sampling distributions.

Our approach to the problem is based on the MDL principle. We define pattern based models, called code tables, that compress the data stream. These code tables characterise the sampling distributions and allow the detection of shifts between such distributions. If we would have unbounded data storage, the MDL-optimal partitioning of the data stream would be that one that minimises the total compressed length. However, unbounded storage is clearly not realistic and we will have to resort to a solution that is at best locally optimal.

With bounded storage, the best approach is to first identify distribution P_1 at the beginning of the stream and then look for a shift to distribution P_2. When P_2 has been identified, we look for the next shift, etc. We give MDL-based solutions for both of these sub-problems.

We turn this MDL-based analysis of the problem into algorithms using our earlier KRIMP algorithm [12]. It induces code tables that characterise data distributions in detail [14,15]. Here, we use the code tables given by this algorithm as foundation for the change detection algorithm STREAMKRIMP, which characterises streams on-the-fly.

We empirically test STREAMKRIMP on a variety of data streams. Results on two types of artificial streams show that changes in distribution are detected at the right moment. Furthermore, the experiment on a real stream shows that large data streams pose no problem and changes are accurately spotted while noise is neglected.

2 The Problem Assuming Unbounded Storage

2.1 Preliminaries

We assume that our data consists of a *stream* of *transactions* over a fixed set of *items* $\mathcal{I}$. That is, each transaction is simply a subset of $\mathcal{I}$ and a stream S is an unbounded ordered sequence of transactions, i.e., $S = s_1 s_2 s_3 \cdots$ in which $s_i \subseteq \mathcal{I}$. The individual transactions in a stream are identified by an integer; without loss of generality we assume that this index is consecutive.

A *finite* stream T is a *substream* of S if T consists of consecutive elements of S. In particular $S(i, j)$ is the substream that starts at the i-th element and stops at the j-th.

2.2 The Problem in Distribution Terms

We assume that S consists of a, possibly infinite, set of consecutive non-overlapping subsequences $S = S_1 S_2 S_3 \cdots$ such that

- S_i is drawn i.i.d. from distribution P_i on $\mathcal{P}(\mathcal{I})$.
- $\forall i \in \mathbb{N} : P_i \neq P_{i+1}$

So, informally, the problem is:

Given such a sequence S, identify the subsequences S_i.

The, somewhat loosely formulated, assumption underlying this problem is that the S_i are big enough to identify. If the source of the stream would change the sample distribution at every transaction it emits, identification would be impossible.

2.3 From Distributions to MDL

The two main ingredients of the problem are, of course:

1. How do we identify the distributions P_i?
2. How do we recognise the shift from P_i to P_{i+1}?

If both these problems are solved, the identification of the S_i is trivial.

If the P_i would belong to some well-known family of distributions, the first problem would be solvable by parameter estimation. Unfortunately, this is not a reasonable assumption.

Rather than trying to estimate the underlying distributions directly, we resort, again, to the Minimum Description Length principle (MDL) [9]. The MDL principle can be paraphrased as: *Induction by Compression*. Slightly more formal, this can be described as follows.

Given a set of models $\mathcal{H}$, the best model $H \in \mathcal{H}$ is the one that minimises

$$L(H) + L(D|H)$$

in which
- $L(H)$ is the length, in bits, of the description of H, and
- $L(D|H)$ is the length, in bits, of the description of the data when encoded with H.

In our earlier research on MDL for item set data we have shown that MDL captures the underlying distribution very well indeed [12,14]. In this paper, we employ MDL both to identify the P_i and to identify the shifts from P_i to P_{i+1}.

Streams of transactions are subtly different from transaction databases. The most important difference is that streams are unbounded. This means, e.g., that some care has to be taken to define the support of an item set in a stream.

2.4 Item Sets in Streams

An item set *I* is, as usual, a set of items. That is, $I \subseteq \mathcal{I}$. An item set *I occurs* in a transaction s_i in stream *S*, iff $I \subseteq s_i$. While streams may be infinite, at any point in time we will only have seen a finite substream. In other words, we only have to consider the support of item sets on finite streams. The support of an item set *I* on a finite stream *S* is defined as usual: the number of transactions in *S* in which *I* occurs.

2.5 Coding Finite Data Streams

As in our previous work, we use code tables to compress data streams. Such a code table is defined as follows.

Definition 1. *Let $\mathcal{I}$ be a set of items and $\mathcal{C}$ a set of code words. A code table CT for $\mathcal{I}$ and $\mathcal{C}$ is a two column table such that:*

1. *The first column contains item sets over $\mathcal{I}$, this column contains at least all singleton item sets and is ordered descending on item set 1) length and 2) support.*
2. *The second column contains elements from $\mathcal{C}$, such that each element of $\mathcal{C}$ occurs at most once.*

An item set $I \in \mathcal{P}(\mathcal{I})$ occurs in CT, denoted by $I \in CT$, iff I occurs in the first column of CT, similarly for a code $C \in \mathcal{C}$. For $I \in CT$, $code_{CT}(I)$ denotes its code, i.e., the corresponding element in the second column.

To encode a finite data stream *S* over $\mathcal{I}$ with code table *CT*, we use the COVER algorithm from [12] given in Algorithm 1. Its parameters are a code table *CT* and a transaction *s*, the result is a set of elements of *CT* that cover *s*. COVER is a well-defined function on any code table and any transaction *s*, since *CT* contains at least the singletons.

ALGORITHM 1. COVER

```
1  COVER(CT, s)
2      T := first element c ∈ CT for which c ⊆ s
3      if s \ T = ∅
4          then Res := {T}
5          else Res:= {T} ∪ COVER(CT, s \ T)
6      return Res
```

To encode finite stream *S*, we simply replace each transaction $s \in S$ by the codes of the item sets in its cover. Note, to ensure that we can decode an encoded stream uniquely, we assume that $\mathcal{C}$ is a *prefix code*.

Since MDL is concerned with the best compression, the codes in *CT* should be chosen such that the most often used code has the shortest length. That is, we should use an optimal prefix code, i.e., the Shannon code. To define this for our code tables, we need to know how often a code is used. We define the *frequency* of an item set *I* in *CT* as the number of transactions in *S* in which *I* occurs in its cover. Normalised, this frequency represents the probability that that code is used in the encoding of an arbitrary $s \in S$:

$$P(I|S) = \frac{freq_S(I)}{\sum_{J \in \mathcal{I}} freq_S(J)}$$

The optimal code length is then $-\log$ of this probability and the coding table is optimal if all its codes have their optimal length. That is, a code is optimal for *S* iff

$$|code_{CT}(I)| = -\log(P(I|S))$$

CT is code-optimal for *S* if all its codes $C \in CT$ are optimal for *S*. From now on, we assume that code tables are code-optimal, unless we state differently.

For any finite data stream *S* and any (code-optimal) code table *CT*, we can now compute *L*(*S* | *CT*). The encoded size of a transaction *s*, denoted $l_{CT}(s)$, is simply the sum of the sizes of the codes of the item sets in its cover:

$$l_{CT}(s) = \sum_{I \in \text{COVER}(CT,s)} |code_{CT}(I)|$$

The size of a data stream *S*, denoted $L_{CT}(S)$, is simply the sum of the sizes of its transactions:

$$L_{CT}(S) = \sum_{s \in S} l_{CT}(s)$$

The remaining problem is, what is the size of a code table? For the second column this is clear as we know the size of each of the codes, but what about the first column? For this, we use the simplest code table, i.e, the code table that contains only the singleton elements. This code table, with optimal code lengths for a finite data stream *S*, is called the standard code table for *S*, denoted by *ST*. With this choice, the size of *CT*, denoted by $L_S(CT)$, is given by:

$$L_S(CT) = \sum_{I \in CT} |code_{ST}(I)| + |code_{CT}(I)|$$

With these results, we know the total size of our encoded data stream. It is simply the sum of the size of the encoded data stream plus the size of the code table. That is, we have the following theorem.

Theorem 1. *Let S be a finite data stream over* $\mathcal{I}$ *and let CT be a code table that is code-optimal for S. The total size of the encoded data stream, denoted by* $L(CT, S)$*, is given by:*

$$L(CT, S) = L_{CT}(S) + L_S(CT)$$

Clearly, two different code tables will yield a different encoded size, an *optimal* code table is one that minimises the total size.

Definition 2. *Let S be a finite data stream over* $\mathcal{I}$ *and let* $\mathcal{CT}$ *be the set of code tables that are code-optimal for S.* $CT^{opt} \in \mathcal{CT}$ *is called* optimal *if*

$$CT^{opt} = \operatorname*{argmin}_{CT' \in \mathcal{CT}} L(CT', S)$$

The total size of the stream S encoded with an optimal code table CT^{opt} *is called its* optimal size *and is denoted by* $\mathcal{L}(S)$*:*

$$\mathcal{L}(S) = L(CT^{opt}, S)$$

2.6 The Problem in MDL Terms

Now that we know how to code finite data streams, we can formulise our problem in MDL terminology:

> *Let S be a finite data stream,* partition *S into consecutive substreams* $S_1, \ldots, S_k$*, such that*
>
> $$\sum_{i=1}^{k} \mathcal{L}(S_i) \text{ is minimised}$$

3 The Problem Assuming Bounded Storage

3.1 The Problem of Streams

Let S, T and U be finite data streams, such that U is the concatenation of S and T. There is no guarantee that the optimal partition of U coincides with the optimal partition of S on the S-part of U. This observation points out two disadvantages of the problem as stated above.

1. It assumes knowledge of the complete stream; this is a flagrant contradiction to the main idea of data streams: they are too big to store.
2. It disregards the dynamic nature of data streams. Changes in the underlying distribution can only be detected after the whole stream has been observed. Clearly, such a posteriori results are not that useful.

In other words, we will have to settle for a partitioning that is at best *locally optimal*.

3.2 Too Large to Store: One Distribution

If the stream S is sampled i.i.d. from one distribution only, the estimates of $P(I \mid S(1,n))$ get closer and closer to their true value. That is, we have the following lemma.

Lemma 3. *Let data stream S be drawn i.i.d from distribution Q on $\mathcal{P}(\mathcal{I})$, then*

$$\forall I \in \mathcal{I} : \lim_{n \to \infty} P(I|S(1,n)) = Q(I)$$

This well-known statistical result has an interesting result for code tables: code tables converge! To make this more precise, denote by CT_n an optimal code table on S_n. Moreover, let $CT(S(1, j))$ be a shorthand for $L_{CT}(S(1,j))$.

Theorem 2. *Let data stream S be drawn i.i.d from distribution Q on $\mathcal{P}(\mathcal{I})$, then*

$$\forall k \in \mathbb{N} : \lim_{n \to \infty} |CT_n(S(1,n)) - CT_{n+k}(S(1,n))| = 0$$

Proof. Let *FCT* be a code table in which only the left-hand column is specified. Lemma 3 implies that

$$\forall I \in \mathcal{I}\ \forall k \in \mathbb{N} : \lim_{n \to \infty} |P(I|S(1,n)) - P(I|S(1,n+k))| = 0$$

In other words, the optimal codes we assign to the item sets in *FCT* become the same in the limit. But this implies that an optimal code table on $S(1, n + k)$ is, in the limit, also an optimal code table on $S(1, n)$. □

That is, if our stream comes from one distribution only, we do not need to store the complete stream to induce the optimal code table. A large enough sample suffices. Denote by $CT^{app}(S)$ the optimal code table induced from a large enough "head" of the stream, i.e., after convergence has set in. This head of the stream is denoted by $H(S)$.

Note that, Theorem 2 also suggests a way to check that the sample is large enough. If for some reasonable sized k,

$$|L(S(1,n), CT_n) - L(S(1,n), CT_{n+k})|$$

gets small, we may conclude convergence. Small is, of course, a relative notion: if $L(S(1, n), CT_n)$ is millions of bits, a difference of a few thousand bits can already be considered as small. Hence, it is better to look at a weighted version; which is our improvement rate, defined as follows.

Definition 3. *With the notations from above, the* Improvement Rate *IR is given by:*

$$\frac{|L(S(1,n), CT_n) - L(S(1,n), CT_{n+k})|}{L(S(1,n), CT_n)}$$

When *IR* becomes small in an absolute sense, we may conclude convergence. We return to this observation later.

3.3 Too Large to Store: Detecting Change

So, for a data stream that comes from one distribution, the problem is easy. The optimal code table can be computed from a large enough head of the stream. After this code table has been computed, no further data storage is necessary anymore. The problem is, however, that after a while the distribution changes. How can we detect that?

Let the, finite, stream $S = S_1S_2$ such that S_i is sampled from distribution P_i. Moreover, let CT_i denote the optimal code table on S_i. To detect the change in distribution, we need that:

$$L(S_1, CT_1) + L(S_2, CT_2) < L(S, CT)$$

This equation translates to:

$$L(S_1, CT_1^{app}) + L(S_2, CT_2^{app}) < L(S_1, CT_1^{app}) + L(S_2|CT_1^{app})$$

Note that $L(S, CT)$ translates to the sum of the two heads encoded with CT_1^{app} because CT_1^{app} has converged. That is, if there is *no* change in the underlying distribution, CT_1^{app} is still the correct code table. The second summand has the bar |, since we count $L(CT_1^{app})$ only once.

Because S may be too big to store, we store $H(S)$. To weigh both code tables equally, we approximate the inequality as follows in the definition of a *split*.

Definition 4. *Let the, finite, stream $S = S_1S_2$ such that S_i is sampled from distribution P_i. Moreover, let CT_i^{app} denote the approximated optimal code table for S_i. The pair (S_1, S_2) is called a* split *of S if:*

$$L(H(S_2), CT_2^{app}) < L(H(S_2), CT_1^{app})$$

A split is called minimal *if there is no other split (T_1, T_2) of S such that T_1 is a substream of S_1.*

Note that this definition implies that we do *not* have to store $H(S_1)$ to detect a change in the underlying definition. CT_1^{app} provides sufficient information.

3.4 The Problem for Data Streams with Bounded Storage

We can now formalise our problem for data streams with bounded storage.

Let S be a data stream, partition S *into consecutive substreams* $S_1, \ldots, S_k, \ldots$, *such that*

$$\forall S_i : (S_i, S_{i+1}) \text{ is the minimal split of } S_i S_{i+1}$$

4 The Algorithm

4.1 KRIMP Preliminaries

In [12] we proposed a heuristic algorithm – later called KRIMP – to approximate the optimal code table from a database. For this, it needs a database and a set of candidate item sets. As candidates, all or closed frequent item sets up to a given *minsup* are used. The candidate set is ordered descending on support, item set length and lexicographically. The algorithm starts with the standard code table *ST*. The code table is ordered descending on length and support. One by one, each pattern in the candidate set is added to the code table to see if it helps to improve database compression. If it does, it is kept in the code table, otherwise it is removed. After this decision, the next candidate is tested. Pruning is applied in all experiments reported in this paper, meaning that each time an item set is kept in the code table, all other elements are tested to see whether they still contribute to compression. Elements that don't are permanently removed. See [12] for further details.

Furthermore, in [15] we introduced a method that can be used to generate databases from a KRIMP code table. All statistics showed that the generated databases are very similar to the original databases from which the code tables were induced. We will use this method to generate synthetic streams in the experiment section.

4.3 Finding the Right Code Table on a Stream

We can now translate the formal scheme presented in Subsection 3.2 to a practical implementation: assume that the stream *S* is sampled i.i.d. from one distribution only and find CT^{app} using KRIMP.

The general idea of the algorithm presented in Algorithm 2 is simple: run KRIMP on the growing head of a stream *S* until the resulting code tables converge. As we know that individual transactions don't make a large difference, we work with blocks of *blockSize* transactions. Start with one block and obtain a code table. For each block added to the head, a new code table is induced and the Improvement Rate is computed. Whenever the IR drops below *maxIR*, the code table is good enough and returned.

The other parameters are an *offset* that makes it possible to start anywhere within the stream and the *minsup* used for mining KRIMP candidates.

ALGORITHM 2. FINDCODETABLEONSTREAM

```
1   FINDCODETABLEONSTREAM(S, offset, blockSize,  minsup, maxIR)
2       numTransactions = blockSize
3       CT = KRIMP(S(offset, offset+numTransactions), minsup)
4       ir = Infinite
5       while ir > maxIR
6           numTransactions += blockSize
7           newCT = KRIMP(S(offset, offset+numTransactions), minsup)
8           ir = ComputeIR(CT, newCT)
9           CT = newCT
10      return CT
```

Moreover, a Laplace correction is applied to each code table returned by KRIMP; this to ensure that each code table can encode each possible transaction.

4.4 Detecting Change in a Stream

Given a code table induced on the head of a data stream, we would now like to detect change in the sampling distribution of the rest of the stream. More formally, we would like to detect the minimal split given CT_1^{app}.

The minimal split can be found by inducing code tables on consecutive heads of the stream until a split is encountered. We would rather avoid building a code table for each and every consecutive head, but luckily we can speed things up in two different ways. First of all, change does not come in a single transaction, so again we iterate over blocks instead. Secondly, we can skip each block that obviously belongs to CT_1^{app}.

For this second optimisation, we apply a statistical test that tests whether the encoded size of the current block deviates from the expected size. If it does not, discard it and skip to the next block. Before discarding the head of a converged code table, this data is used to randomly sample encoded block sizes from. Both the lower and upper *leaveOut* percent samples are removed. If the encoded size of a new block falls within the range of the remaining samples, the block is considered to belong to the distribution of CT_1^{app} and skipped.

For each block that is not skipped, we have to test whether it marks a split or not. For this, we have to induce a code table CT_2^{app}. To be able to reject a code table that is only just better than the previous one, we introduce the Code Table Difference:

Definition 5. *Given a code table CT_1^{app} and a code table CT_2^{app} induced on H(S_2), the* Code Table Difference *CTD is given by:*

$$\frac{L(H(S_2), CT_1^{app}) - L(H(S_2), CT_2^{app})}{L(H(S_2), CT_2^{app})}$$

Normalised the same way as the Improvement Rate, the CTD tells us how many percent CT_2^{app} compresses the new head better than CT_1^{app}. We can now define a minimum CTD in the overall algorithm, which is presented next.

Algorithm 3: StreamKrimp

```
1   StreamKrimp(S, minsup, blockSize, maxIR, leaveOut, minCTD)
2       i = 1
3       CT_i = FindCodeTableOnStream(S, 0, blockSize, minsup, maxIR)
4       pos = CT_i.endPos
5       while pos < sizeof(S)
6           pos = SkipBlocks(S, CT_i , pos, blockSize, leaveOut)
7           candCT = FindCodeTableOnStream(S, pos, blockSize, minsup, maxIR)
8           if ComputeCTD(S, CT_i, candCT) >= minCTD
9               i++
10              CT_i = candCT
11              pos = candCT.endPos
12          else
13              pos += blockSize
14      return CT
```

4.5 StreamKrimp

Putting together the algorithms of the previous subsections, we are able to partition a stream into consecutive substreams with minimal splits. The complete algorithm is shown in Algorithm 3.

It starts with finding the code table on the head of the stream (line 3) and then iterates over the rest of the stream. Each iteration starts with skipping as many blocks as possible (6). When a block cannot be skipped straightaway, it is used as starting position for a new candidate code table (7). The Code Table Difference of this candidate to the current code table is computed (8) and the code table is either accepted (9-11) or rejected (13). When finished, the complete set of code tables is returned (14). Naturally, these could be inspected and used while the algorithm is still running as well.

4.6 How Large is Zero?

Or: How should we set our parameters? We will here motivate the default values we suggest for the algorithm, which we will use throughout the rest of the paper.

minsup – Lower *minsup* levels result in more candidate patterns and therefore better compression and better quality code tables. Sensitivity of the change detection scheme is influenced through this parameter: lower values result in a higher sensitivity. To avoid overfitting on very small data segments, we use a *minsup* of 20 in the experiments presented in this paper.

blockSize – The resolution at which StreamKrimp works should always be high enough. This will be the case if we choose the size of a block such that, on average, every possible item occurs once in every block. Therefore, we choose *blockSize* to be equal to $|\mathcal{I}|$.

leaveOut – Set to 0.01: both the lower 1% and the upper 1% of the randomly sampled blocksizes are discarded by SkipBlocks.

maxIR – Set to 0.02: if a new code table compresses less than 2% better than its predecessor, we decide it has converged.

minCTD – Set to 0.10: a new code table is accepted only if it compresses at least 10% better than the previous code table.

The choices for *maxIR* and *minCTD* may seem arbitrary, but this is not the case. They are actually comparable to the dissimilarity values we reported before [15]. Dissimilarity values between random samples from a single dataset range from 0.045 to 0.177 on UCI datasets (also reported on in the next section). Therefore, 0.02 and 0.10 are very conservative and may be considered zero for all practical purposes: with these thresholds, code tables converge and significant changes are detected.

5 Experiments

5.1 Artificial Streams – UCI Datasets

The first series of experiments is done on a selection of the largest datasets from the well-known UCI repository [6], as shown in Table 1. These datasets are transaction databases and not streams, but they have the advantage that each of them consists of multiple known classes. This allows for easy validation of the identified splits.

To transform a UCI dataset into a stream, each dataset is split on class label and the class labels are removed from all transactions. This results in a transaction database per class. The transactions within each database are ordered (randomly) to form a stream. The resulting streams are concatenated into one single stream (in random order). Because of this randomisation, each dataset is turned into a stream 10 times.

The main results are summarised in Table 2. The '#CTs' column tells us how many code tables have been identified for each of the datasets. If we compare these numbers to the actual number of classes in Table 1, we see that StreamKrimp finds the right number of distributions in the stream. Only for Chess, the algorithm doesn't find enough splits, but this is not surprising as there are quite many classes and some of them are rather small. Analysing the splits reveals that indeed the larger classes are identified and only the smallest ones go undetected.

The next column, 'Blocks per CT', tells us that approximately 4 to 6 blocks are enough for code table construction on these datasets. For some datasets, such as Adult, Chess and Nursery, quite some code tables are rejected, as is shown under '#CTs rejected'. However, also quite some blocks are skipped by SkipBlocks. These values vary quite a bit for the different datasets, telling us that the conservative statistical skip test seems to work better for one dataset than for another.

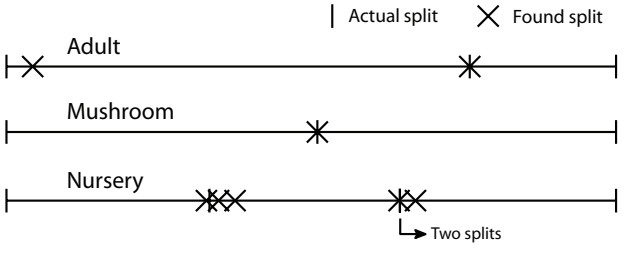


Fig. 1. Actual and found splits for three of the datasets

The last columns show baseline and obtained purity values. Purity is the size of the majority class relative to the entire segment, averaged over all identified segments. Baseline purity is the size of the largest class. Although the transactions of the classes are not interleaved and the task is therefore easier than clustering, the attained purity values are very high. This indicates that the algorithm correctly identifies the boundaries of the classes in the streams. This is supported by Figure 1, which depicts actual and found splits for three datasets. No expected splits are missed by STREAMKRIMP and the shift moments are accurately detected. (For each dataset, a single run with average performance is shown.)

Table 1. Properties of 7 UCI datasets: number of rows, classes and items

Dataset	**#rows**	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{I}\rvert$
Adult	48842	2	97
Chess (kr-k)	28056	18	58
Led7	3200	10	24
LetRecog	20000	26	102
Mushroom	8124	2	119
Nursery	12960	5	32
PenDigits	10992	10	86

5.2 Artificial Streams – Synthetic

The experiments in the previous subsection show that the proposed algorithm accurately detects changes in a stream. The objective of the following experiments is simple: which elementary distribution changes are detected?

We manually create simple code tables and use the KRIMP data generator [15] to generate a series of synthetic datasets. Each generated stream consists of two parts: 5000 rows generated with one code table, followed by 5000 rows generated by a variation on this code table. In these experiments, $\lvert\mathcal{I}\rvert$ is 10 and each item is in the code table with a count of 1 (i.e., all individual items are generated with equal probability). The databases have 5 two-valued attributes (resulting in a total of 10 possible items). So, each transaction consists of 5 items.

Because the number of different items is very small (only 10) we manually set the *blockSize* for these experiments to 200. This way, we ensure that KRIMP gets enough data and candidate patterns to learn the structure that is in the data.

Table 2. Results for 7 UCI datasets. For each dataset, the following is listed: the number of code tables found, the average number of blocks used for construction per CT, the number of code tables rejected, the number of blocks skipped and finally base and obtained purity. Averages over 10 randomisations (class order, transaction order within classes).

Dataset	**#CTs**	**Blocks per CT**	**#CTs rejected**	**Blocks skipped**	**Purity**	
					Baseline	*Actual*
Adult	3.4	4.7	118	367	76.1%	99.7%
Chess (kr-k)	13	4.2	165	264	17.8%	80.1%
Led7	12	3.9	3.5	82	10.9%	95.2%
LetRecog	27	5.9	0.2	32	4.1%	80.1%
Mushroom*	2.7	6.2	6.2	44	51.7%	96.5%
Nursery	6.2	5.7	140	228	33.3%	98.5%
PenDigits	15	6.0	2.3	34	10.4%	87.2%

* Closed frequent item sets used as candidates instead of all frequent item sets.

The basic distribution changes made halfway in the synthetic datasets (through changing the code tables) are depicted in Figure 2. Each rounded box represents an item set, with the individual items given as numbers. All counts are set to 5, except for those item sets where a multiplier is shown (x4 means 5x4=20). As data generation is a stochastic process, 10 streams were generated for each change and the main results shown in Table 3 are averaged over these. The last column indicates the number of times the algorithm found the optimal solution; two code tables and 100% purity (i.e., a split after 5000 rows). From the results it is clear that StreamKrimp is very capable at detecting 4 out of 6 of the tested types of change. Only detection of the subtle addition of a single (small) pattern and changing the frequency of an existing pattern turns out to be difficult occasionally. In these cases, change is often detected but this takes some blocks, resulting in lower purity values.

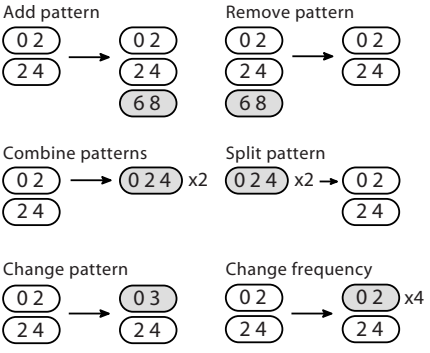


Fig. 2. Changes inflicted in the code tables used for stream generation

5.3 A Real Stream – Accidents Dataset

A more realistic dataset is taken from [8]. It contains data obtained from the National Institute of Statistics (NIS) for the region of Flanders (Belgium) for the period 1991-2000. More specifically, the data are obtained from the Belgian 'Analysis Form for Traffic Accidents' that should be filled out by a police officer for each traffic accident that occurs with injured or deadly wounded casualties on a public road in Belgium. In total, 340,184 traffic accident records are included in the data set.

No timestamps are available, but accidents are ordered on time and it is an interesting question whether structural changes can be detected. With over 340,000 transactions over a large number of items ($|\mathcal{I}| = 468$), running any regular pattern mining algorithm on the entire dataset is a challenge. Therefore, it is a perfect target for finding 'good enough' code tables and detecting change. As Krimp candidates we use closed frequent item sets with minimum support 500 and we rounded the block size to 500.

Table 3. Results for 6 synthetic streams. For each dataset, the following is listed: the number of code tables found, the number of code tables rejected, obtained purity and the number of optimal solutions found. Averages over 10 generated streams (except for the last column).

Change	#CTs	#CTs rejected	Purity %	Optimal (out of 10)
Add pattern	1	1.8	79.8	0
Remove pattern	1.3	3.8	97.0	6
Combine patterns	1.3	2.8	98.6	6
Split pattern	1.1	2.6	98.8	8
Change pattern	1.1	1.8	98.6	7
Change frequency	1.9	15.6	75.2	1

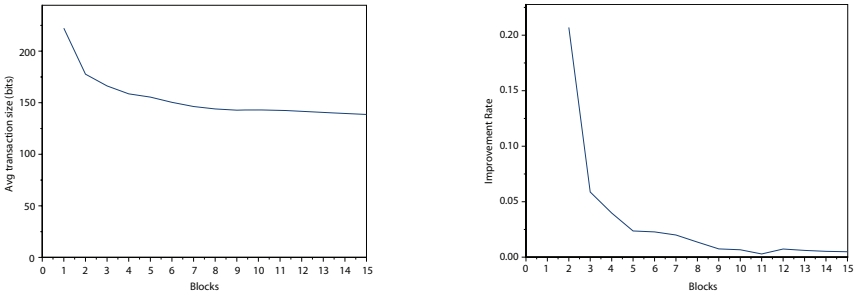


Fig. 3. Average encoded length per transaction (left) and improvement rates (right) for the code tables built on 15 consecutive blocks from the Accidents dataset

An important question we have not yet addressed with the (much smaller) artificial streams is how well the FINDCODETABLEONSTREAM algorithm approximates the best possible code table on a stream. To assess this, the average encoded size per transaction is plotted for a series of code tables in Figure 3 on the left. On the right, Figure 3 shows the computed Improvement Rates for the same set of code tables. Each code table is built on a head of x blocks, where x is the number of blocks indicated on the x-axis. Average encoded size is computed on all transactions the code table is induced from. The graphs clearly show that the most gain in compression is obtained in the first few blocks. After that, the average size per transaction decreases only slowly and this is also reflected in the Improvement Rate. With *maxIR* set to 0.02, STREAMKRIMP would pick the code table built on 8 blocks, which seems a good choice: after that, improvement is marginal.

Running STREAMKRIMP on the entire dataset resulted in only 14 code tables that characterise the entire stream of over 340,000 transactions. 140 blocks were skipped, 429 code tables were built but rejected. On average, 7.43 blocks of data were required for a code table to converge and the average Code Table Difference of accepted code tables was 0.14. This means that each consecutive distribution differs about 14% from its predecessor in terms of compression!

To further illustrate the differences between the identified substreams, Figure 4 shows compressed block sizes over the entire dataset for three consecutive code tables. Substreams clearly consist of blocks that are equally well compressed. The split the end of

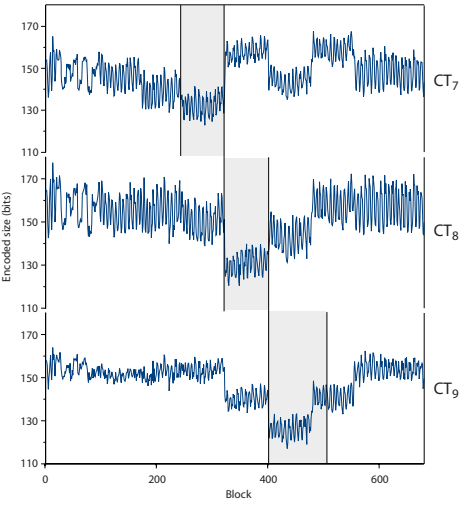


Fig. 4. Encoded length per block for three consecutive substreams on Accidents. The blocks belonging to each of the code tables are indicated with the grey blocks.

the last substream shown seems to be a tad late, the rest is spot on. In other words, the change detection is both quick and accurate, also on large datasets.

6 Related Work

Stream mining has attracted a lot of attention lately, which is nicely illustrated by the recent book by Aggarwal et al. [3]. Here, we focus on change detection.

In many cases, streams are considered to be sequences of single (real-valued) numbers or items. Kifer et al. [10] use two sliding windows to obtain two samples of which it is statistically determined whether they belong to different distributions. Papadimitriou et al. [13] use autoregressive modelling and wavelets, Muthukrishnan et al. [11] avoid a fixed window size by introducing a method based on sequential hypothesis testing.

A second class of stream mining algorithms considers multi-dimensional, real-valued streams. Aggarwal et al. [1] visualise evolving streams using velocity density estimation. The visualisation is inherently 2-dimensional and it is not possible to accurately estimate densities with increasing dimensionality. In [2], Aggarwal et al. use a polynomial regression technique to compute statistically expected values.

Dasu et al. [7] take an information-theoretic approach by using the Kullback-Leibler distance to measure the difference between distributions. They experiment on multi-dimensional real-valued data, but claim the method can also be applied to categorical data. However, a fixed window size strongly influences the changes that can be detected and the method seems better suited for relatively few dimensions (<10).

Widmer and Kubat [16] use an adaptive window size to do online learning in domains with concept drift. Predictive accuracy is used to detect drift and adjust the window size heuristically. This does require (known) binary class labels though.

The final class of algorithms considers streams of categorical transactions, as we do in this paper. Chen et al. [5] propose a method to visualise changes in the clustering structure of such streams. A disadvantage is that snapshots of these visualisations have to be manually analysed. Recently, Calders et al. [4] proposed an alternative 'minimum support' measure for patterns in streams called max-frequency. This measure uses flexible windows to maintain the max-frequency on patterns in the stream.

7 Discussion

The results on both the artificial and realistic streams show that StreamKrimp is very capable at detecting changes in large data streams. No actual splits are missed and the results on the synthetic streams show that even small modifications in the distribution can be detected.

The algorithm satisfies the general requirements for stream mining, as only very limited data storage is required and online mining is possible. Also, the resulting code tables are much smaller than the data itself and can therefore be stored for a much longer time. This means that it is possible to store a full characterisation of the entire stream.

In many stream mining algorithms, a window size has to be defined. This window size determines what changes can and can not be found; nothing outside the window is seen. Contrary, the block size of our algorithm is only the resolution which determines how quickly a distribution is detected and characterised.

8 Conclusions

We introduce StreamKrimp, an algorithm that detects changes in the sampling distribution of a stream of transactions. Based on an analysis from MDL perspective, it partitions a stream into a sequence of substreams. For each substream, it uses Krimp to characterise its distribution with a code table and each subsequent code table indicates a change in the underlying distribution. Only a very limited amount of data storage is required and StreamKrimp facilitates online mining of streams.

The results of experiments on both artificial and realistic streams show that StreamKrimp detects the changes that make a difference, no relevant changes are missed and noise is neglected. Finally, large streams with many attributes pose no problems.

References

1. Aggarwal, C.C.: A framework for diagnosing changes in evolving data streams. In: Proceedings of ACM SIGMOD 2003 (2003)
2. Aggarwal, C.C.: On Abnormality Detection in Spuriously Populated Data Streams. In: Proceedings of SIAM Conference on Data Mining 2005 (2005)
3. Aggarwal, C.C. (ed.): Data Streams: Models and Algorithms. Springer, Heidelberg (2007)
4. Calders, T., Dexters, N., Goethals, G.: Mining Frequent Itemsets in a Stream. In: Proceedings of IEEE ICDM 2007 (2007)
5. Chen, K., Liu, L.: Detecting the Change of Clustering Structure in Categorical Data Streams. In: Proceedings of SIAM Conference on Data Mining 2006 (2006)
6. Coenen, F. The LUCS-KDD Discretised/normalised ARM and CARM Data Library (2003), http://www.csc.liv.ac.uk/~frans/KDD/Software/
7. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An Information-Theoretic Approach to Detecting Changes in Multi-Dimensional Data Streams. In: Proceedings of Interface 2006 (2006)
8. Geurts, K., Wets, G., Brijs, T., Vanhoof, K.: Profiling of high-frequency accident locations using association rules. In Transportation research record 1840 (2003)
9. Grünwald, P.D.: Minimum description length tutorial. In: Grünwald, P.D., Myung, I.J., Pitt, M.A. (eds.) Advances in Minimum Description Length. MIT Press, Cambridge (2005)
10. Kifer, D., Ben-David, S., Gehrke, J.: Detecting Change in Data Streams. In: Proceedings of VLDB 2004 (2004)
11. Muthukrishnan, S., van den Berg, E., Wu, Y.: Sequential Change Detection on Data Streams. In: Proceedings of the ICDM Workshops 2007 (2007)
12. Siebes, A., Vreeken, J., Van Leeuwen, M.: Item Sets That Compress. In: Proc. of the ACM SIAM Conference on Data Mining, pp. 393–404 (2006)
13. Papadimitriou, S., Brockwell, A., Faloutsos, C.: Adaptive, unsupervised stream mining. The VLDB Journal 13(3), 222–239 (2004)
14. Vreeken, J., Van Leeuwen, M., Siebes, A.: Characterising the Difference. In: Proceedings of ACM SIGKDD 2007 (2007)
15. Vreeken, J., Van Leeuwen, M., Siebes, A.: Preserving Privacy through Generation. In: Proceedings of IEEE ICDM 2007 (2007)
16. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23, 69–101 (1996)

Author Index