

Lecture Notes in Artificial Intelligence

1810

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Ramon López de Mántaras Enric Plaza (Eds.)

Machine Learning: ECML 2000

11th European Conference on Machine Learning
Barcelona, Catalonia, Spain, May 31 – June 2, 2000
Proceedings



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Ramon López de Mántaras
Enric Plaza
Institut d'Investigació en Intelligència Artificial, IIIA
Spanish Council for Scientific Research, CSIC
Campus, U.A.B.
08193 Bellaterra, Catalonia, Spain
E-mail: {mantaras/enric}@iia.csic.es

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Machine learning : proceedings / ECML 2000, 11th European Conference
on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2,
2000. Ramón López de Mántaras ; Enric Plaza (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris
; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1810 : Lecture notes in
artificial intelligence)
ISBN 3-540-67602-3

CR Subject Classification (1998): I.2, F.2.2

ISSN 0302-9743

ISBN 3-540-67602-3 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-Tex Gerd Blumenstein
Printed on acid-free paper SPIN 10720254 06/3142 5 4 3 2 1 0

Foreword

The biennial European Conference on Machine Learning (ECML) series is intended to provide an international forum for the discussion of the latest high quality research results in machine learning and is the major European scientific event in the field. The eleventh conference (ECML 2000) held in Barcelona, Catalonia, Spain from May 31 to June 2, 2000, has continued this tradition by attracting high quality papers from around the world.

Scientists from 21 countries submitted 100 papers to ECML 2000, from which 20 were selected for long oral presentations and 23 for short oral presentations. This selection was based on the recommendations of at least two reviewers for each submitted paper. It is worth noticing that the number of papers reporting applications of machine learning has increased in comparison to past ECML conferences. We believe this fact shows the growing maturity of the field.

This volume contains the 43 accepted papers as well as the invited talks by Katharina Morik from the University of Dortmund and Pedro Domingos from the University of Washington at Seattle. In addition, three workshops were jointly organized by ECML 2000 and the European Network of Excellence MLnet: “Dealing with Structured Data in Machine Learning and Statistics Web-sites”, “Machine Learning in the New Information Age” , and “Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination”. Finally, a special workshop on “Learning Agents” was jointly organized by ECML 2000 and the co-located International Conference on Autonomous Agents. Information on the workshops can be found on the ECML 2000 web page: <http://www.iia.csic.es/ecml2000/>.

We gratefully acknowledge the work of the invited speakers and the authors of the submitted papers that made this conference possible. We also thank the program committee and the additional reviewers for their effort during the paper selection process. Our gratitude also goes to the ECML 2000 sponsors: MLnet, CSIC, CICYT, and ACIA, as well as to Gemma Sales for her assistance in the organization of the conference.

March 2000

Ramon López de Mántaras
Enric Plaza

Organization

The 11th European Conference on Machine Learning took place on May 30th to June 2nd 2000 in Barcelona, Catalonia (Spain).

Program Chairs

Ramon López de Mántaras (mantaras@iia.csic.es), Enric Plaza (enric@iia.csic.es), Institut d'Investigació en Intel·ligència Artificial (IIIA-CSIC), Bellaterra, Catalonia (Spain)

Conference Support

ECML 2000 was organized by the Institut d'Investigació en Intel·ligència Artificial (IIIA), Consejo Superior de Investigaciones Científicas (CSIC), Bellaterra, Catalonia (Spain). Institutions that provided support for ECML 2000 are ACIA (Catalan Association for Artificial Intelligence), CICYT (Spanish Science and Technology Council), CSIC (Spanish Council for Scientific Research), and ML-net (European Network of Excellence on Machine Learning IST-29288)

Program Committee

A. Aamodt (Norway)	Y. Kodratoff (France)
D. Aha (USA)	I. Kononenko (Slovenia)
K. D. Althoff (Germany)	N. Lavrac (Slovenia)
F. Bergadano (Italy)	R. López de Mántaras (Spain)
D. Borrajo (Spain)	S. Matwin (Canada)
I. Bratko (Slovenia)	K. Morik (Germany)
P. Brazdil (Portugal)	V. Moustakis (Greece)
E. Costa (Portugal)	S. Muggleton (UK)
K. DeJong (USA)	G. Nakhaeizadeh (Germany)
L. De Raedt (Germany)	C. Nedellec (France)
T. G. Dietterich (USA)	E. Plaza (Spain)
P. Domingos (USA)	C. Rouveirol (France)
M. Dorigo (Belgium)	S. Russell
S. Dzeroski (Slovenia)	L. Saitta (Italy)
F. Esposito (Italy)	M. van Someren (Netherlands)
D. H. Fisher (USA)	D. Sleeman (UK)
P. Flach (UK)	S. Thrun (USA)
J. Fuerkranz (Austria)	C. Torras (Spain)
J. G. Ganascia (France)	S. Weiss (Germany)
O. Gascuel (France)	G. Widmer (Austria)
M. Goldszmidt (USA)	S. Wrobel (Germany)

Additional Reviewers

J. Aguilar Martin (France)	M. Liquiere (France)
K. Barker (Canada)	P. Machado (Portugal)
K. Bartlmae (Germany)	D. Malerba (Italy)
M. Birattari (Italy)	J. M. Molina (Spain)
M. Borth (Germany)	F. B. Pereira (Portugal)
O. A. Brevik (Norway)	F. A. Rauscher (Germany)
P. Brockhausen (Germany)	I. Renz (Germany)
T. Brychcy (Germany)	M. Robnik-Sikonja (Slovenia)
A. Cornuejols (France)	G. Ruffo (Italy)
D. Faure (France)	V. Ruiz de Angulo (Spain)
R. Gavalda (Spain)	M. Saerens (Belgium)
M. Grobelnik (Slovenia)	T. Scheffer (Germany)
D. Gunetti (Italy)	R. Schneider (Germany)
T. Hrycej (Germany)	G. Semeraro (Italy)
P. Isasi (Spain)	L. Todorovski (Slovenia)
A. Jorge (Portugal)	L. Torgo (Portugal)
T. Joachims (Germany)	F. Torre (France)
H. Kauderer (Germany)	C. Ungerer (Germany)
R. Klinkenberg (Germany)	M. Wiering (Italy)
M. Kukar (Slovenia)	D. Zupanic (Slovenia)
C. Lanquillon (Germany)	

Table of Contents

Invited Papers

Beyond Occam's Razor: Process-Oriented Evaluation	3
<i>Pedro Domingos</i>	
The Representation Race – Preprocessing for Handling Time Phenomena	4
<i>Katharina Morik</i>	

Contributed Papers

Short-Term Profiling for a Case-Based Reasoning Recommendation System	23
<i>Esma Aimeur and Mathieu Vézeau</i>	
K-SVCR. A Multi-class Support Vector Machine	31
<i>Cecilio Angulo and Andreu Català</i>	
Learning Trading Rules with Inductive Logic Programming	39
<i>Liviu Badea</i>	
Improving Knowledge Discovery Using Domain Knowledge in Unsupervised Learning	47
<i>Javier Béjar</i>	
Exploiting Classifier Combination for Early Melanoma Diagnosis Support ..	55
<i>E. Blanzieri, C. Eccher, S. Forti and A. Sboner</i>	
A Comparison of Ranking Methods for Classification Algorithm Selection ..	63
<i>Pavel B. Brazdil and Carlos Soares</i>	
Hidden Markov Models with Patterns and Their Application to Integrated Circuit Testing	75
<i>Laurent Bréhélin, Olivier Gascuel and Gilles Caraux</i>	
Comparing Complete and Partial Classification for Identifying Latently Dissatisfied Customers	88
<i>Tom Brijs, Gilbert Swinnen, Koen Vanhoof and Geert Wets</i>	
Wrapper Generation via Grammar Induction	96
<i>Boris Chidloukii, Jon Ragethi and Maarten de Rijke</i>	
Diversity versus Quality in Classification Ensembles Based on Feature Selection	109
<i>Pádraig Cunningham and John Carney</i>	

Minimax TD-Learning with Neural Nets in a Markov Game	117
<i>Fredrik A. Dahl and Ole Martin Halck</i>	
Boosting Applied to Word Sense Disambiguation	129
<i>Gerard Escudero, Lluís Màrquez and German Rigau</i>	
A Multiple Model Cost-Sensitive Approach for Intrusion Detection	142
<i>Wei Fan, Wenke Lee, Salvatore J. Stolfo and Matthew Miller</i>	
Value Miner: A Data Mining Environment for the Calculation of the Customer Lifetime Value with Application to the Automotive Industry	154
<i>Katja Gelbrich and Reza Nakhaeizadeh</i>	
Investigation and Reduction of Discretization Variance in Decision Tree Induction	162
<i>Pierre Geurts and Louis Wehenkel</i>	
Asymmetric Co-evolution for Imperfect-Information Zero-Sum Games	171
<i>Ole Martin Halck and Fredrik A. Dahl</i>	
A Machine Learning Approach to Workflow Management	183
<i>Joachim Herbst</i>	
The Utilization of Context Signals in the Analysis of ABR Potentials by Application of Neural Networks	195
<i>Andrzej Izvorski, Ryszard Tadeusiewicz and Andrzej Paslawski</i>	
Complexity Approximation Principle and Rissanen's Approach to Real-Valued Parameters	203
<i>Yuri Kalnishkan</i>	
Handling Continuous-Valued Attributes in Decision Tree with Neural Network Modeling	211
<i>DaeEun Kim and Jaeho Lee</i>	
Learning Context-Free Grammars with a Simplicity Bias	220
<i>Pat Langley and Sean Stromsten</i>	
Partially Supervised Text Classification: Combining Labeled and Unlabeled Documents Using an EM-like Scheme	229
<i>Carsten Lanquillon</i>	
Toward an Explanatory Similarity Measure for Nearest-Neighbor Classification	238
<i>Mathieu Latourrette</i>	
Relative Unsupervised Discretization for Regression Problems	246
<i>Marcus-Christopher Ludl and Gerhard Widmer</i>	
Metric-Based Inductive Learning Using Semantic Height Functions	254
<i>Zdravko Markov and Ivo Marinchev</i>	

Error Analysis of Automatic Speech Recognition Using Principal Direction Divisive Partitioning	263
<i>David McKoskey and Daniel Boley</i>	
A Study on the Performance of Large Bayes Classifier	271
<i>Dimitris Meretakís, Hongjun Lu and Beat Wüthrich</i>	
Dynamic Discretization of Continuous Values from Time Series	280
<i>Llanos Mora López, Inmaculada Fortes Ruiz, Rafael Morales Bueno and Francisco Triguero Ruiz</i>	
Using a Symbolic Machine Learning Tool to Refine Lexico-syntactic Patterns	292
<i>Emmanuel Morin and Emmanuelle Martienne</i>	
Measuring Performance when Positives Are Rare: Relative Advantage versus Predictive Accuracy – A Biological Case-Study	300
<i>Stephen H. Muggleton, Christopher H. Bryant and Ashwin Srinivasan</i>	
Mining TCP/IP Traffic for Network Intrusion Detection by Using a Distributed Genetic Algorithm	313
<i>Filippo Neri</i>	
Learning Patterns of Behavior by Observing System Events	323
<i>Marlon Núñez</i>	
Dimensionality Reduction through Sub-space Mapping for Nearest Neighbour Algorithms	331
<i>Terry R. Payne and Peter Edwards</i>	
Nonparametric Regularization of Decision Trees	344
<i>Tobias Scheffer</i>	
An Efficient and Effective Procedure for Updating a Competence Model for Case-Based Reasoners	357
<i>Barry Smyth and Elizabeth McKenna</i>	
Layered Learning	369
<i>Peter Stone and Manuela Veloso</i>	
Problem Decomposition for Behavioural Cloning	382
<i>Dorian Šuc and Ivan Bratko</i>	
Dynamic Feature Selection in Incremental Hierarchical Clustering	392
<i>Luis Talavera</i>	
On the Boosting Pruning Problem	404
<i>Christino Tamon and Jie Xiang</i>	
An Empirical Study of MetaCost Using Boosting Algorithms	413
<i>Kai Ming Ting</i>	
Clustered Partial Linear Regression	426
<i>Luís Torgo and Joaquim Pinto da Costa</i>	

Knowledge Discovery from Very Large Databases Using
Frequent Concept Lattices 437
Kitsana Waiyamai and Lotfi Lakhal

Some Improvements on Event-Sequence Temporal Region Methods 446
Wei Zhang

Author Index 459

Beyond Occam's Razor: Process-Oriented Evaluation

Pedro Domingos

Department of Computer Science and Engineering, University of Washington
Box 352350, Seattle, WA 98195, U.S.A.

pedrod@cs.washington.edu

<http://www.cs.washington.edu/homes/pedrod>

Abstract. Overfitting is often considered the central problem in machine learning and data mining. When good performance on training data is not enough to reliably predict good generalization, researchers and practitioners often invoke "Occam's razor" to select among hypotheses: prefer the simplest hypothesis consistent with the data. Occam's razor has a long history in science, but a mass of recent evidence suggests that in most cases it is outperformed by methods that deliberately produce more complex models. The poor performance of Occam's razor can be largely traced to its failure to account for the search process by which hypotheses are obtained: by effectively assuming that the hypothesis space is exhaustively searched, complexity-based methods tend to over-penalize large spaces. This talk describes how information about the search process can be taken into account when evaluating hypotheses. The expected generalization error of a hypothesis is computed as a function of the search steps leading to it. Two variations of this "process-oriented" approach have yielded significant improvements in the accuracy of a rule learner. Process-oriented evaluation leads to the seemingly paradoxical conclusion that the same hypothesis will have different expected generalization errors depending on how it was generated. I believe that this is as it should be, and that a corresponding shift in our way of thinking about inductive learning is required.

References

1. P. Domingos. Occam's two razors: The sharp and the blunt. *Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining*, pp. 37–43, New York, NY, 1998. AAAI Press.
2. P. Domingos. A process-oriented heuristic for model selection. *Proc. 15th Intl. Conf. on Machine Learning*, pp. 127–135, Madison, WI, 1998. Morgan Kaufmann.
3. P. Domingos. Process-oriented estimation of generalization error. *Proc. 16th Intl. Joint Conf. on Artificial Intelligence*, pp. 714–719, Stockholm, Sweden, 1999. Morgan Kaufmann.
4. P. Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.

The Representation Race – Preprocessing for Handling Time Phenomena

Katharina Morik

Univ. Dortmund, Computer Science VIII,
D-44221 Dortmund, Germany
morik@ls8.cs.uni-dortmund.de
<http://www-ai.cs.uni-dortmund.de>

Abstract. Designing the representation languages for the input, L_E , and output, L_H , of a learning algorithm is the hardest task within machine learning applications. This paper emphasizes the importance of constructing an appropriate representation L_E for knowledge discovery applications using the example of time related phenomena. Given the same raw data – most frequently a database with time-stamped data – rather different representations have to be produced for the learning methods that handle time. In this paper, a set of learning tasks dealing with time is given together with the input required by learning methods which solve the tasks. Transformations from raw data to the desired representation are illustrated by three case studies.

1 Introduction

Designing the representation languages for the input and output of a learning algorithm is the hardest task within machine learning applications. The “no free lunch theorem” actually implies that if a hard learning task becomes easy because of choosing appropriate representations, the choice of or the transformation into the appropriate representation must be hard [38]. The importance of L_H , the representation of the output of learning, is well acknowledged. Finding the hypothesis space with most easily learnable concepts, which contains the solution, has been supported by systems with declarative language bias [18], [11], [7] or representation adjustment capabilities [35], [39]. It is also the key idea of *structural* risk minimization, where the trade-off between complexity and accuracy of a hypothesis guides the learning process [37].

The importance of L_E , the representation of the input of learning, has received some attention only recently. Transforming the given representation of observations into a well-suited language L_E may ease learning such that a simple and efficient learning algorithm can solve the learning problem. For instance, first order logic examples and hypothesis space are transformed into propositional logic in order to apply attribute-value learning algorithms [23], [22]. Of course (and in accordance with the “no free lunch theorem”), the transformed set of examples might become exponentially larger than the given one. Only if some restrictions can be applied, the transformation plus the transformed

learning problem are indeed easier than the original learning problem on the original representation. The central issue is to find appropriate restrictions and corresponding transformations for a given task [19].

The problem of designing L_E is not limited to the representation formalism but includes the selection or construction of appropriate features within a formalism [24]. The problem has become particularly urgent, since knowledge discovery confronts machine learning with databases that have been acquired and designed for processes different from learning. Given mature learning algorithms and the knowledge of their properties, the challenge is now to develop transformations from raw data L_E to suitable $L_{E'}$. The transformation can be a learning step itself so that L_{E_1} delivers $L_{H_1} = L_{E_2}$, or it can be another aggregation or inferential step. In general, we consider a series of transformations from the given raw data L_{E_1} to the input of the data mining step, L_{E_n} ¹. The technical term of “preprocessing” seems euphemistic when considering the effort spent on this transformation sequence in comparison to the effort spent on the data mining step. Rather we might view the exploration and design of transformations a *representation race* where the winner leads to the most efficient and accurate learning of the interesting concept, rules, or subgroups. The new European project MININGMART aims at supporting end-users in winning the representation race.

This paper emphasizes the importance of transforming given data into a form appropriate for (further) learning. The MININGMART approach to supporting a user in this difficult task is illustrated by learning tasks which refer to time phenomena. First, the project is briefly described. Since it has just begun, only the main idea and the goals are reported. Second, time phenomena are discussed. Handling time is an excellent example of how data sets can be transformed in diverse ways according to diverse learning tasks and algorithms that solve them. Different views of time phenomena are elaborated and an overview of existing methods is given. Third, preprocessing operators for handling time phenomena are discussed on the basis of three case studies.

2 The MiningMart Approach

The MININGMART will be a system supporting knowledge discovery in databases. A set of transformation tools/operators will be developed in order to construct appropriate representations $L_{E'}$. Machine learning operators are not restricted to the data mining step within knowledge discovery. Instead, they are seen as preprocessing operators that summarize, discretize, and enhance given data. This view offers a variety of learning tasks that are not as well investigated as is learning classifiers. For instance, an important task is to acquire events and their duration (i.e. a time interval) on the basis of time series (i.e. measurements at time points). The tools improve the quality of data with respect to redundancy and noise, they assist the user in selecting appropriate samples, in discretizing

¹ This relates the issue of preprocessing closely to multistrategy learning [26].

numeric data and provide means for the reduction of the dimensionality of data for further processing. Making data transformations available includes the development of an SQL query generator for given data transformations and the execution of SQL queries for querying the database.

The main problem is, that nobody has yet been able to identify reliable rules predicting when one algorithm should be superior to others. Beginning with the MLT-CONSULTANT [34] there was the idea of having a knowledge-based system support the selection of a machine learning method for an application. The MLT-CONSULTANT succeeded in differentiating the nine MLT learning methods with respect to specific syntactic properties of the input and output languages of the methods. However, there was little success in describing and differentiating the methods on an application level that went beyond the well known classification of machine learning systems into classification learning, rule learning, and clustering. Also, the European STATLOG-Project [27], which systematically applied classification learning systems to various domains, did not succeed in establishing criteria for the selection of the best classification learning system. It was concluded that some systems have generally acceptable performance. In order to select the best system for a certain purpose, they must each be applied to the task and the best selected through a test-method such as cross-validation. Theusinger and Lindner [36] are in the process of re-applying this idea of searching for statistical dataset characteristics necessary for the successful applications of tools. An even more demanding approach was started by Engels [13]. This approach not only attempts to support the selection of data mining tools, but to build a knowledge-based process planning support for the entire knowledge discovery process. To date this work has not led to a usable system [14]. The European project METAL now aims at learning how to combine learning algorithms and datasets [8]. At least until today, there is not enough knowledge available in order to propose the correct combination of preprocessing operations for a given dataset and task.

The other extreme of the top-down knowledge-based approach to finding appropriate transformation sequences is the bottom-up exploration of the space of preprocessing chains. Ideally, the system would evaluate all possible transformations in parallel, and propose the most successful sequence of preprocessing steps to the user. This is, however, computationally infeasible. Therefore, the MININGMART follows a third way. It allows each user to store entire chains of preprocessing and analysis steps for later re-use in a case-base (for example, a case of preprocessing for mailing-actions, or a case of preprocessing for business reports). Cases are represented in terms of meta-data about operators and data, are presented to the users in business terms, and are made operational by SQL query generators and learning tools. The case-base of preprocessing and analysis tasks will not only assist the inexperienced user through the exploitation of experienced guidance from past successful applications, but will also allow any user to improve his or her skill for future discovery tasks by learning from the best-practice discovery cases.

3 Handling Time Phenomena

Most data contain time information in one way or another. Think, for instance, of a database storing warranty cases. Among data about the sold item including its production date, there would be data about the sale including the selling date, data about the warranty case together with the date of the claim, the expiration time of warranty, and the payment. Time stamps are natural attributes to all objects described in the database. Depending on the learning task, the same raw data are transformed into rather different example sets. Some of these simply ignore the time stamps, but others take particular care of time phenomena. In this section, first an overall view of time phenomena is presented. This is a necessary step towards a meta-level description of learning tasks related with time. Algorithms that solve one such task are briefly presented in the following subsections. This section concludes with a list of $L_{E'}$ required by the learning methods.

3.1 Structuring Time Phenomena

For the overall view, we may structure time phenomena by two aspects, linear precedence and immediate dominance. These terms have been defined in natural language theory [15]. *Linear precedence* refers to the ordering of elements in a sequence. It is the relation between elements occurring along the time axis, horizontally depicted in Figure 1. Most statistical approaches are restricted to this aspect of time. *Immediate dominance* refers to categories of the time-dependent elements. Categories summarize observations to events of increasingly abstract levels². The linear precedence relation between most abstract categories is propagated to the lowest level of interest, the actually observable actions or events. Sequencing rules often refer to categories (events) instead of their elements.

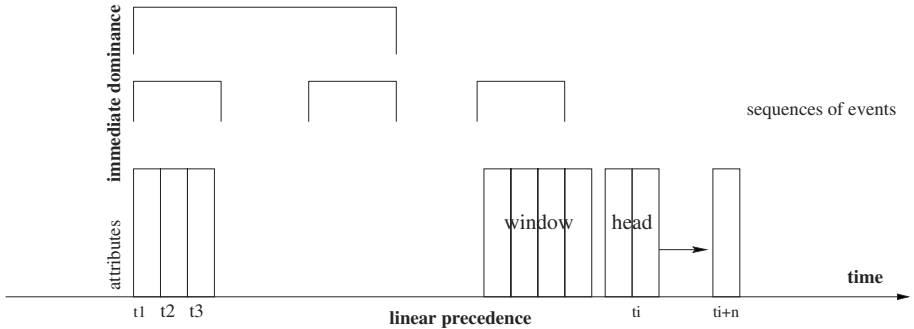


Fig. 1. The overall view of time phenomena

² Mannila and Toivonen name the basic observations *events* and the higher level categories *episodes*.

Learning tasks concerning linear precedence are:

Prediction: Given a sequence of elements until time point t_i , predict the element that will occur at time point t_{i+n} . We call n the horizon.

Characterization: Characterize a time ordered sequence of elements by its trend (i.e. the elements are increasingly or decreasingly ordered over time), a seasonal increasing or decreasing peak, or a cyclic ordering of elements. The cyclic ordering can be described by a function (e.g., sinus, cosinus, wavelet).

Time regions: Given time gaps between occurrences of elements, predict a time interval in which an element is to be expected.

Level changes: Detect time points in a sequence of elements, where the elements are no longer homogenous according to some measure.

Clustering: Given subsequences in a sequence of events find clusters of similar sequences.

Note, that methods about linear precedence can be used to solve the problem of forming (basic) categories. It is evident, that finding trends, seasons, cycles, level changes, and clusters can be used to discretize time series. Hence, these learning tasks can be considered as preprocessing for the learning tasks concerning immediate dominance. They are valuable tasks in their own right, though.

Learning tasks concerning immediate dominance are:

Frequent Sequences: Given sequences of events, learn the precedence relation between sets of events. The sets of events in precedence relation have also been called *episodes*.

Non-determinate sequence prediction: Given a sequence of observations and the background knowledge about characterizations and categories of the basic observations, learn a set of rules that is capable of producing legal sequences.

Relations: Given events and their duration (i.e. a time interval), learn sequences of events in terms of relations between time intervals. Time relations are the ones defined in Allen's time calculus [4,5]: overlap, inclusion, (direct) precedence, ...

Higher-level categories: Given events and their duration together with a classification in terms of a category c of the next higher level, learn the definition of c .

Non-determinate sequence prediction has been solved by [25] and has currently received attention in the context of biochemical analyses [31]. It is also the task that has to be solved for language learning. Since the datasets for sequence prediction do not include any explicit time stamp, we do exclude this very interesting issue here.

A more detailed structure of time phenomena distinguishes between handling abstract and actual time. Consider, for instance, the action of sweetening tea. This category summarizes the actions of putting sugar into the tea and stirring. These categories, in turn, can be instantiated by various alternative observable actions (e.g., using a spoon or pouring the sugar into the cup). It is not at all

important, how long after putting the sugar in, one has to start stirring. Nor is the actual time in seconds interesting for the duration of stirring – stirring is performed as long as the sugar is not yet dissolved. This illustrates abstract time. Consider, in contrast, the duration of drawing the tea. Here, the actual time of 3 minutes is important. In principle, all the tasks listed above could be solved with respect to abstract or to actual time. However, learning tasks concerning linear precedence are typically solved using actual time.

An important choice when describing actual time is the scale. We may use seconds, minutes, days, months or even milleniums. Moreover, even for the same granularity, we may choose different scales of referene. For instance, measurements of vital signs in intensive care units are recorded on a minute to minute basis. Counting the minutes does not start at midnight (day time), but when the patient is connected to the monitoring machines (duration of stay). Transforming the data from one scale to the other allows to discover different regularities. The morning visit, for instance, explains why the therapy is adjusted in a time interval where the patient’s state it not worse than, say, at 2 o’clock in the night. Other therapeutical interventions can better be explained using the scale referring to the duration of stay. Hence, the description of L_E should indicate the scale.

3.2 Statistical Approaches

Statistical approaches view time series as observing a process where a measurement depends on previous measurements³. In principle, the time axis is structured into three areas: the relevant past, the current observations and the observation to be predicted. Diverse functions are chosen to compute a value for the measurements of the relevant past: the average (in simple moving average procedures), the weighted average, where more recent measurements are multiplied by a higher weight than the ones that occured longer ago (weighted moving average), or the weights are such that weights for the relevant past and the weight for the current observation sum up to 1 (exponential moving average), smoothing algorithms use the median for values of the relevant past. Another algorithm uses the gradient [30]. Autocorrelation procedures (ARMA) consider whether past values and current value show the same ($r = 1$) or opposite direction ($r = -1$) and possibly use r^2 . Choices regarding moving average models refer to noise models, the number of observations in the window (lag) of the relevant past, and whether more than one current observation is considered.

Filtering approaches consider the function over time and filter out the peaks (high pass) or the slow move (low pass). Possibly, Fourier analysis is applied decomposing the original curve into a set of sinus and cosinus curves. This is, of course, only possible, if measurements are not received on-line, but the curve is given in total.

³ Since this overview of statistical approaches corresponds to textbooks, I do not give references, if the particular method will not be used in succeeding sections. Focusing on data mining, [32] explains statistical approaches comprehensively.

Multivariate time series analysis is capable of considering the dynamics over time of up to about 5 attributes. Frequently, a multivariate time series is decomposed into a set of univariate time series, thus disregarding the dependencies of different attributes.

In addition to the learning task of predicting the next measurement, the detection of trends, cycles, and seasons is investigated. For abstracting the time point view of linear precedence into time intervals of actual time, the detection of level changes can be used. An interesting recent approach is to transform the time series into a phase space [6]. The visualisation clearly shows regularities that cannot be recognized in the original form. The summary of time intervals according to a level can be seen as a first step towards immediate dominance.

The task of clustering subsequences has been solved in order to obtain categories as input to finding frequent sequences [9]. All subsequences of window length w are formed and similar subsequences are clustered together. The clusters are labeled. The original sequence is transformed into the sequence of labels. The categories apply to overlapping sections of the original curve. This has to be taken into account, when using clustering as preprocessing for rules discovery.

The notion of *examples* becomes difficult when investigating time series. For instance, all minutely measurements of n attributes of a process are just one example for an n -variate time series. The prediction task is solved for one example, although by the technique of moving windows, many subseries are obtained and exploited for learning. The learning result is then applied to the very same process. If the process is something like the stock market or the wheather, there is, in fact, no other similar process available. We do not want to generalise the American, the Japanese, and the German stock market, if they are not (yet) observations of the same global economical process. Nor do we want to generalise the “wheather” of different planets. Hence, time series analysis really differs from the well established paradigm of empirical risk minimization, which assumes many independent observations of different individuals (processes). Let us look at other time stamped data. If warranty claims are analysed, the recall of mailing actions, or christmas sales, the aim is to generalize over sets of customers. This is in accordance with the principle of risk minimization. In order to apply time series methods, we have to perform the generalization step in advance. This can easily be done, for instance, by summing up the sales data of all shops. The result is one time series. It looks exactly like the time series of one process. However, it makes a difference in that less observations from the past are needed, because the present “observation” is already empirically based.

3.3 Frequent Groups

The discovery of subgroups is one of the most common tasks of knowledge discovery. Originated in the database field, there is no assumption about the process producing the data. The typical database stores a huge amount of independent elements (e.g., contracts, sales, warranty cases). Frequent patterns in the data generalize over masses of time series. Association rules describe that some elements frequently occur together (frequent item sets). According to the confidence

measure, the set is divided into an indicator set and an expected set. Although presented for basket analysis, the well-known APRIORI algorithm exploits an ordering of the items [1,2]. Hence, it can easily be applied to sequences [3]. We only need to interpret the ordering as the time attribute. The confidence measure requires some adjustment. Rules of the form $A \rightarrow_T B$ state that if A occurs, then B occurs within time T . The frequency $F(A, B, T)$ is determined by counting, how often A precedes B , given a window of size T . The confidence for the rule can be defined as $\frac{F(A, B, T)}{F(A)}$, where $F(A)$ denotes the frequency of A [9]. A variety of algorithms concerning the discovery of frequent sequences exist, ranging from just testing a user-specified sequence pattern [16] to relational approaches [10]. Frequent subsequences can be detected using actual or abstract time. The algorithms can be applied directly to the time stamped data, or a categorisation step is performed in advance.

3.4 Relational Learning

Often, it is interesting to find relations between durations of diverse events or categories. We might be interested in dependencies between events that are produced by different processes or abstract relations such as “as long as” or “directly after” or “in parallel”. A natural way to represent time relations are rules with time points as chaining arguments, chain rules [12], [33]. Although chain rules are not restricted to time as the chaining arguments, they are particularly well suited for modeling time phenomena in both aspects, linear precedence and immediate dominance.

General chain rule: Let S be a literal or a set of literals. Let $args(S)$ be a function that returns the Datalog arguments of S . A normal clause is a general chain rule, iff its body literals can be arranged in a sequence $B_0 \leftarrow B_1, B_2, \dots, B_k, B_{k+1}$ such that there exist Datalog terms $Begin, End \in args(B_0)$, $Begin, T_1 \in args(B_1)$, $T_1, T_2 \in args(B_2)$, $\dots, T_{k-1}, T_k \in args(B_k)$, and $T_k, End \in args(B_{k+1})$.

Chain rules can express relations between time intervals, form higher-level categories, and dependencies between different multivariate time series. They require facts as input that include two arguments referring to time points that mark the begin and the end of the time interval. Most algorithms of inductive logic programming are capable of learning chain rules. Hence, they solve the learning tasks related with immediate dominance. Either actual or abstract time may be used. However, they are weak in numerical processing. Therefore, time series with numerical attributes should be discretized beforehand.

3.5 Required L_E

The input formats for the selection of methods presented, are now listed. We write attributes A_j and their values a_j , abstract time points T_i and actual time points t_i , the class that is described by the attributes I_l and an instance i_l . This

notation is meant to be close to the one of database theory. Note, however, that the semantic notion of the class being described can be mapped to the database relation or its key or even to one of the attributes of the database relation.

L_{E_1} **multivariate time series:** From a vector with measurements of attributes $A_1, \dots, A_k$, i.e. $i_l : t_1 a_{1_1} \dots a_{1_k}, \dots, t_i a_{i_1} \dots a_{i_k}$ methods solving the prediction task output $i_l : t_{i+n} a_{i+n_1} \dots a_{i+n_k}$, methods characterizing the time series output a label for a trend (e.g., increasing), a time interval for a season, or a function for the cycle. Note, that a_{i_j} are numerical values.

$L_{E_1'}$ **univariate time series:** The vector of measurements here only contains one numerical attribute: $i_l : t_1 a_1, \dots, t_i a_i$. The output for prediction, trend, season, or cycle is similar to the one of multivariate series. Methods for detecting level changes deliver $i_l : t_m, t_n a_{mn}$, where a_{mn} is a computed value, e.g., the average. Clustering delivers a sequence of $Label_j[t_i, t_{i+w}]$, where w is the window size and the label is some computed summary of the attribute values $a_i \dots a_{i+w}$.

L_{E_2} **nominal valued time series:** From a vector of nominal attribute values that can already be considered events, i.e. $i_l : t_1 a_{1_1} \dots a_{1_k}, \dots, t_i a_{i_1} \dots a_{i_k}$, the time region approach [40] outputs a set of rules of the form

$I : a_u, \dots, a_v \rightarrow_{[t_b, t_e]} a_z$, where $u, v, z \in [1, k]$ and $b, e \in [1, i]$. The rule states that within the time interval $[t_b, t_e]$ the event a_z is to be expected if $a_u, \dots, a_v$ have been observed.

L_{E_3} **sequence vectors:** A large set of vectors with nominal or numerical attribute values is the input to finding frequent sequences. The scheme of the vectors is similar to univariate time series, but the example set always consists of a large number of individuals that are described by the attribute. The time span is fixed to the given number of fields in the vector. The scheme $I : T_1 A_1, \dots, T_i A_i$ is instantiated by all individuals about which data are stored in the database. The time points can vary from instance to instance, but the ordering is fixed. Rules learned are of the form $I : a_u, \dots, a_v \rightarrow_{[t_e]} a_z$ with the meaning introduced for nominal valued time series.

L_{E_4} **facts:** A set of facts possibly concerning individuals of different classes, indicating a time interval of abstract or actual time ($[T_b, T_e]$) for an event given by several attributes are the input to relational learning. The number and type of attributes may vary for different predicates p that instantiate P . The facts have the form $P(I_1, T_b, T_e, A_r, \dots, A_s)$, where some attribute A can denote another class.

If the learning task is to define higher-level categories, a classifying fact must be given for each example that is represented by a set of facts of the above form. The classifying fact has at least a time interval as arguments and the predicate denotes the higher-level category.

We have now developed a set of frequently used representations for the input of (time related) learning. In addition, many methods require the parameters window size w , the number of current observations *head*, and the prediction horizon n . The time scale has to be indicated by the granularity and the starting point of reference. Whereas L_{E_1} and L_{E_2} internally produce a large set of

data for one example by moving windows, L_{E_3} and L_{E_4} are representations for sets of examples (independent observations). L_{E_4} in addition possibly combines different classes of individuals within one example. The notation for the examples already covers some semantical aspects. This is important in order to preprocess data appropriately, namely, to distinguish between attributes that refer to time, to a class, to features of an individual, or to relations between individuals of different classes.

4 Preprocessing for Time Phenomena

In order to discuss the transformations into the desired formats, let us now look at typical cases of raw data. We illustrate the representations by three cases that each stands for a large range of applications. The first case is a typical database with time-stamped database tuples, the second is a set of robot traces, where the measurements of 24 sensors are recorded over time, the third is a database of intensive care patients with their vital signs and infusions measured every minute.

The most frequent representation of raw data is

L_{EDB1} **database table:** a set of individuals and a set of time points is given according to the scheme $I : T_1 A_1 \dots A_k, \dots, T_i A_1 \dots A_k$. It is a large set of multivariate time series with nominal or numerical values for the attributes.

Let us now look at the options for transforming the data into appropriate representations for learning.

4.1 The Shop Application – Representing Time Implicitly

In our first example, I is instantiated by shops, $i = 104$ denotes the weeks of two years, A_j is an item, and a_j its sale. In our application, the task was to predict the sales for an item in a time horizon n , that varies from $n = 4$ to $n = 13$. The prediction is necessary for optimizing the storage of goods. Of course, seasonal effects are present. They are already stored as binary flags within the database. The learning method was the regression mode of the support vector machine (SVM) [37]. The SVM requires input vectors of fixed length with numerical values. The method does not handle time explicitly. It is a rather common approach to compile time phenomena into attributes that are then handled by the learning method as any other attribute. The most frequently used choices are:

Multivariate to univariate transformation: For each attribute A_1 to A_k , store a vector for the corresponding univariate time series: $I : t_1 a_1, \dots, t_i a_i$. The result are k vectors for all $i \in I$. In our example, where the sales of 50 items were analyzed, 50 vectors were stored for each of the 20 shops.

Sliding windows: Choose a window size of w consecutive time points, store the vector $i : t_1 a_{1_1} \dots a_{1_k}, \dots, t_w a_{w_1} \dots a_{w_k}$, move the starting point by m steps and repeat, until $t_w = t_i$ (in our example $t_i = 104$). The result is a set of $i - w$ vectors for one time series (in our case, window sizes 3, 4, 5 were tried yielding 103 to 105 vectors).

Summarizing: Attribute values within a window of past observations are summarized by some function $f(a_{i_j}, \dots, a_{i+w_j})$ (e.g., average, gradient, variance). The original time series is replaced by the discretized one:

$$i : [t_1, t_w] f(a_{1_j}, \dots, a_{w_j}), [t_m, t_{m+w}] f(a_{m_j}, \dots, a_{m+w_j}), \dots$$

Some approaches do not fix the window size, but find it in a data-driven fashion [9], [30]. Whereas most approaches deliver overlapping time intervals, [30] deliver a discretized time series with consecutive time intervals. Summarizing time windows is also viewed as a method of feature construction. In our shop example, we did not summarize the time series.

Multiple learning: Instead of handling diverse individuals in one learning run, a learning run can be started for each individual. The learning result of this run is used for the prediction concerning this individual only. In our shop example, for each shop (20) and each item (50), a separate learning of support vectors was started. The results are then used to predict the sales of this item in this particular shop.

Aggregation: Aggregating the shops by summing up the sales made in all shops did not perform well in our application. In principle, however, this aggregation is a common transformation. It constructs $i' \in I$ and a'_j and hence produces one time series for all individuals.

The resulting representation for the SVM that proved successful by cross validation was: $i : t_{i+w} a_{i+w_j} \text{season}, \dots, t_i a_{i_j}$. This is a quite common combination of transformations if we follow a statistic-oriented approach: multivariate to univariate, sliding windows, and multiple learning.

4.2 The Applications in Intensive Care

The second example, records of patients in an intensive care unit, offers raw data of the L_{EDB} form. The difference is that the length of the time series is not determined once for all patients but denotes the length of the patient's stay in the intensive care unit. Hence, the database table is organised as consecutive parts of the time series.

L_{EDB2} **tuples for time points:** The database no longer stores all measurements of one individual in one row, but only the measurements at one point in time: $I : T A_1 \dots A_k$. There are several rows for one individual. The number of measurements needs not be equal for different individuals.

$$i_1 : t_1 a_{1_1} \dots a_{1_k} \dots i_1 : t_i a_{i_1} \dots a_{i_k}$$

...

$$i_z : t_l a_{l_1} \dots a_{l_k}$$

We explored a variety of learning tasks within this application. The learning when and how to change the dosage of drugs is – with respect to preprocessing and learning – similar to the shop example, but we had to combine different rows of the table first.

Chaining database rows: Select all rows concerning the same individual and group its attributes by the time points. Output a vector of the length of the time series of this individual. The result is a multivariate time series.

Again, we used the SVM, now in the classification mode [17]. We experimented with many different features that were formed by sliding windows and different summarization methods. However, the past did not contribute to learning the decision rule. Hence, we learned from patients’ state at t_i whether and how to intervene at t_{i+1} [29]. Therefore, in the end we could use the original data. The real difference to the shop application is that the decision rule is learned from a large training set of different patients and then applied to previously unseen patients and their states.

For a different learning task in the intensive care application, a method for time series analysis was applied. The learning task was to find outliers and detect level changes. A new statistical method was used [6]. It transforms measurements of one vital sign of the patient such that the length of the window is interpreted as dimensions of Euclidian space. Choosing $w = 2$, the measurements of two consecutive time points are depicted as one point in a two-dimensional coordinate system. It turns out, that outliers leave the ellipse of homogeneous measurements, and a level change can be seen as a new ellipse in another region of the space. The method is a special case of sliding windows.

In the intensive care application, current work now uses the detected level changes as input to a relational learning algorithm. This allows to combine various time series and detect dependencies among parameters, deviations from a stable (healthy) state, and therapeutical interventions. The learning task is to find time relations that express therapy protocols, in other words effective sequences of interventions.

4.3 The Application in Robot Navigation

The third application to be presented here, is about sensor measurements of a mobile robot. The raw data are of the L_{EDB2} type, where I denotes mission or path, from which the measurements are taken, k is the number of sensors (in our case 24), and the only attribute is the measured distance to some (unknown) object. The learning tasks are higher-level concepts that can be used for navigation planning and execution [21]. Using chain rules with abstract time arguments allows to apply the learned knowledge to different environments⁴. However, relational learners that are capable of learning them, require facts as input, where the predicate indicates the summary of measurements and two arguments indicate the time interval in which it is valid. The requirements were

⁴ The post-processing of learned rules into real-time control is summarized in [28].

further that the transformation can be applied on-line, i.e. purely incrementally, and the time intervals do not overlap. This excludes the standard methods from statistics as well as the approach of [9]. Hence, we developed our own method that closes a time interval if the gradient of the current summary and the current measurement varies more than a given threshold [30]. Predicate symbols denote classes of gradients, e.g. **increase**, **decrease**, **peak**. The first step of preprocessing was to chain database rows in order to acquire a 24-variate time series for each mission. The second was to transform each one into 24 univariate time series. To these our variant of summarizing was applied.

Input to relational learning was first the set of all summarized univariate time series of all missions, together with the classification of the higher-level category (e.g., **sensor_along_wall**). The learned rules describe sequences of summarized sensor measurements that define the higher category. A classification corresponding to the placement of the sonar sensor at the robot was then used to combine sequences of several sensors. This led to the learned definition of sensor group features. In a bootstrap manner a hierarchical logic program was learned, that integrates all 24 sensors. Moreover, irrelevant relations between summarized measurements and their time intervals are filtered out by relational learning. The low accuracy of 27.1% (for **sensor_along_wall**) and 74.7% (for **sensor_through_door**) at the lowest level increased to 54.3% (**along_wall**) and 93.8% (**through_door**) at the highest level where all perceptions and actions are integrated [20]. This is surprising, because the learned rules of the lower level produced the examples for learning at the next higher level. It clearly shows the importance of taking into account the aspect of immediate dominance when handling time. Handling time with respect to linear precedence alone is unable to discover dependencies between 24 time series in several missions. It also illustrates the power of preprocessing: we could well consider all learning steps at lower levels as a chain of transformations that allow the highest-level data mining step.

5 Conclusion

In this paper, nine time-related learning tasks were presented, together with classes of algorithms that solve them. Five input languages for the methods were distinguished. Given two standard representations of time-stamped data in databases, it was shown, how they can be transformed into the desired languages for learning. All the transformations proved their value in many applications – not only the ones named in the paper. However, a uniform description of data, learning tasks, methods and L_E transformations was missing. The description shown in this paper can now be made operational as meta-data and transformation tools. I am certain, that the lists of tasks and transformations is not complete and that new publications will contribute more tasks and methods. This is not a counter argument, though. In contrast, it emphasizes the need for a preprocessing library.

Since we do not know which representation will turn out to be the best for a learning task, we have to try out several representations in order to determine the winner. This is a tedious and time consuming process. It is the goal of the MININGMART project to supply users with a workbench offering preprocessing tools in a unified manner. Moreover, a case base will present for several applications the winners of the representation race.

Acknowledgements

This work has been partially funded by the European Commission, IST-11993 (MiningMart). I particularly thank those partners with whom I enjoyed fruitful discussions along proposal writing.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, D. C., may 1993. 11
2. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthrusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI Press/The MIT Press, Cambridge Massachusetts, London England, 1996. 11
3. Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *International Conference on Data Engineering*, Taipei, Taiwan, mar 1995. 11
4. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984. 8
5. James F. Allen. Maintaining knowledge about temporal intervals. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, chapter VII, pages 509–523. Morgan Kaufman, Los Altos, CA, 1985. 8
6. M. Bauer, U. Gather, and M. Imhoff. Analysis of high dimensional data from intensive care medicine. Technical Report 13/1998, Sonderforschungsbereich 475, Universität Dortmund, 1998. 10, 15
7. Francesco Bergadano and Daniele Gunetti. *Inductive logic programming: from machine learning to software engineering*. The MIT Press, Cambridge, Mass., 1996. 4
8. Prael Brazdil. Data transformation and model selection by experimentation and meta-learning. In C. Giraud-Carrier and M. Hilario, editors, *Workshop Notes – Upgrading Learning to the Meta-Level: Model Selection and Data Transformation*, number CSR-98-02 in Technical Report, pages 11–17. Technical University Chemnitz, April 1998. 6
9. Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery from Time Series. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 16 – 22, New York City, 1998. AAAI Press. 10, 11, 14, 16

10. Luc Dehaspe and Hannu Toivonen. Discovery of Frequent DATALOG Patterns. *Data Mining and Knowledge Discovery*, 3(1):7 – 36, 1999. 11
11. Luc DeRaedt. *Interactive Theory Revision: an Inductive Logic Programming Approach*. Acad. Press, London [u.a.], 1992. 4
12. G. Dong and S. Ginsburg. On the decomposition of chain datalog programs into p (left-)linear l-rule components. *Logic Programming*, 23:203 – 236, 1995. 11
13. Robert Engels. Planning tasks for knowledge discovery in databases; performing task-oriented user-guidance. In *Proc. of the 2nd Int. Conf. on Knowledge Discovery in Databases*, aug 1996. 6
14. Robert Engels, Guido Lindner, and Rudi Studer. A guided tour through the data mining jungle. In *Proceedings of the 3rd International Conference on Knowledge Discovery in Databases (KDD-97)*, August 14–17 1997. 6
15. Gerald Gazdar and Chris Mellish. *Natural Language Processing in PROLOG*. Addison Wesley, Workingham u.a., 1989. 7
16. Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33 – 42, San Diego, USA, 1999. 11
17. T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11. MIT-Press, 1999. 15
18. J.-U. Kietz and S. Wrobel. Controlling the complexity of learning in logic through syntactic and task-oriented models. In Stephen Muggleton, editor, *Inductive Logic Programming.*, number 38 in The A.P.I.C. Series, chapter 16, pages 335–360. Academic Press, London [u.a.], 1992. 4
19. Jörg-Uwe Kietz and Marcus Lübke. An efficient subsumption algorithm for inductive logic programming. In W. Cohen and H. Hirsh, editors, *Proceedings of the 11th International Conference on Machine Learning IML-94*, San Francisco, CA, 1994. Morgan Kaufmann. 5
20. Volker Klingspor. *Reaktives Planen mit gelernten Begriffen*. PhD thesis, Univ. Dortmund, 1998. 16
21. Volker Klingspor and Katharina Morik. Learning understandable concepts for robot navigation. In K. Morik, V. Klingspor, and M. Kaiser, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning*. Kluwer, 1999. 15
22. S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In D. Page, editor, *Procs. 8th International Workshop on Inductive Logic Programming*, pages 80 – 94. Springer, 1998. 4
23. Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming — Techniques and Applications*. Number 148 in Artificial Intelligence. Ellis Horwood, Hertfortshire, 1994. 4
24. H. Liu and H. Motoda. *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Kluwer, 1998. 5
25. R.S. Michalski and T.G. Dietterich. Learning to predict sequences. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning - An Artificial Intelligence Approach Vol II*, pages 63–106. Tioga Publishing Company, Los Altos, 1986. 8
26. Ryszard Michalski. Inferential learning theory as a basis for multistrategy task-adaptive learning. In Michalski and Tecuci, editors, *Multistrategy Learning*. George Mason University, USA, 1991. 5
27. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York u.a., 1994. 6

28. Katharina Morik. Tailoring representations to different requirements. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory – Procs. 10th Int. Conf. ALT99*, Lecture Notes in Artificial Intelligence, pages 1 – 12. Springer, 1999. 15
29. Katharina Morik, Peter Brockhausen, and Thorsten Joachims. Combining statistical learning with a knowledge-based approach – A case study in intensive care monitoring. In *Proc. 16th Int’l Conf. on Machine Learning (ICML-99)*, Bled, Slowenien, 1999. 15
30. Katharina Morik and Stephanie Wessel. Incremental signal to symbol processing. In K. Morik, M. Kaiser, and V. Klingspor, editors, *Making Robots Smarter – Combining Sensing and Action through Robot Learning*, chapter 11, pages 185 – 198. Kluwer Academic Publ., 1999. 9, 14, 16
31. S. Muggleton, A. Srinivasan, R. King, and M. Sternberg. Biochemical knowledge discovery using inductive logic programming. In Hiroshi Motoda, editor, *Procs. First International Conference on Discovery Science*. Springer, 1998. 8
32. Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers, 1999. 9
33. Anke D. Rieger. *Program Optimization for Temporal Reasoning within a Logic Programming Framework*. PhD thesis, Universität Dortmund, Germany, Dortmund, FRG, 1998. 11
34. D. Sleeman, R. Oehlman, and R. Davidge. Specification of Consultant-0 and a Comparison of Several Learning Algorithms. Deliverable D5.1, Esprit Project P2154, 1989. 6
35. Devika Subramanian. A theory of justified reformulations. In D. Benjamin, Paul, editor, *Change of Representation and Inductive Bias*, pages 147–167. Kluwer, 1990. 4
36. C. Theusinger and G. Lindner. Benutzerunterstützung eines KDD-Prozesses anhand von Datencharakteristiken. In F. Wysotzki, P. Geibel, and K. Schädler, editors, *Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Machinelles Lernen (FGML-98)*, volume 98/11 of *Technical Report*. Technical University Berlin, 1998. 6
37. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995. 4, 13
38. D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fé Institute, Santa Fé, CA., 1995. 4
39. Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, Dordrecht, 1994. 4
40. Wei Zhang. A region-based approach to discovering temporal structures in data. In Ivan Bratko and Saso Dzeroski, editors, *Proc. of 16th Int. Conf. on Machine Learning*, pages 484 – 492. Morgan Kaufmann, 1999. 12

Short-Term Profiling for a Case-Based Reasoning Recommendation System

Esma Aimeur and Mathieu Vézeau

Computer Science Department - University of Montreal
C.P. 6128, Succ. Centre-ville
Montreal, Quebec Canada H3C 3J7

aimeur@iro.umontreal.ca
Phone: 1-514-343-6794
Fax: 1-514-343-5834

Abstract. In this paper, we aim to address a frequent shortcoming of electronic commerce: the lack of customer service. We present an approach to product recommendation using a modified cycle for case-based reasoning in which a new refinement step is introduced. We then use this cycle combined with a heuristic we devised to create a short-term profile of the client. This profile is not stored or reused after the transaction, reducing maintenance. In fact, it allows the client and the system to find an appropriate product to satisfy the client on the basis of available products in a potentially efficient way.

1 Introduction

Electronic commerce on the Internet is steadily gaining importance and promises to revolutionize the way we exchange products and services. However, many problems remain to be solved before we can fully exploit the potential benefits of this new paradigm. One of those problems is the lack of customer service in electronic commerce applications. For now, sales support offered by enterprises to their Internet customers is generally poor, if existent at all. Of course, most sites give their customers the ability to query the available products, by way of catalogs, textual search engines or database interfaces. These tools, however, require the user to expend much effort and can be totally inadequate if the number of products is substantial, if the products are alike or if the consumer does not know the domain very well.

One solution is to use products recommendation systems. Those systems are able to suggest products to clients according to their preferences or specific requirements. This way, those applications contribute to increase customer satisfaction, therefore increasing sales and improving the reputation of enterprises using these systems. One of the promising technologies for the conception of recommendation systems is case-based reasoning (CBR) [1]. The goal of this sub-domain of artificial intelligence is to

conceive knowledge-based systems which, to solve new problems, reuse and adapt solutions to prior similar problems.

In this paper, we propose a heuristic to construct a temporary user profile in CBR-based recommendation systems. This approach is able to fulfill one of the most common deficiencies of these systems, that is to say that they react the same way with all users without regard to their respective preferences.

We first explain the proposed heuristic and then give an example for an application of this heuristic in electronic commerce. Finally, we discuss pros and cons of this method and propose future directions.

2 CBR and Short-Term Profiling

In the context of electronic commerce, the CBR cycle can be interpreted as an iterative search process in the multidimensional space of the products. The initial request is used to find a first solution in the set of possibilities, and we expect the user to iterate progressively toward the ideal solution by formulating successive critiques on the different characteristics of the proposed products. This progression is made possible by adding to the cycle a request refinement step during which the system accounts for the critique by automatically modifying the user request and starting a new search. This interpretation implies that unlike traditional CBR systems, where the cycle is generally executed only once by session (search, adapt, evaluate, retain), recommendation systems for electronic commerce, for their part, are used on a basis of many iterations by session (search, adapt, evaluate, retain, adapt, etc.). By session, we mean a phase where the client interacts with the system to find a particular item. A session may contain an arbitrary number of interactions but must end when the client finds a satisfactory product.

Our approach is based on the following assumption: since the client interacts many times with the system in the same session, this situation is propitious to discovering “short-term” preferences of this client. Here we do not speak of sought product characteristics, those characteristics being already clearly specified in the request. We are instead hinting at the relative significance the user attaches to the various parameters describing the sought product. A client looking for a travel package, for instance, could attach more importance to the price than to the destination itself. In that case, the system should respect the user preferences and rely more on the price in its search. This kind of preference is, in our opinion, temporary, hence the qualifier “short-term”. Indeed, we believe the importance given to concepts in a product search is more associable with temporary interests and to circumstantial causes than with long-term interests. In the preceding example, it is possible that the user wished to travel at this time of the year but the destination was not important in his eyes. Maybe the user had a limited budget at that time, and though he had a destination in mind, he was willing, if needed, to sacrifice that choice for a price within his budget. That kind of motivation oftentimes depends on the moment and is not necessarily representative of the personality of the user. According to our approach, the user profile is thus valid only for a single session. This contrasts with traditional approaches to user modeling,

where profiles mostly represent long-term interests and preferences, and where those profiles are created and maintained on the basis of many sessions with the system.

We make the hypothesis that the various critiques formulated by the client in his search process for the “ideal product” are a good source of information to construct a temporary profile automatically. In fact, in the kind of applications we consider, we expect the client to make heavier use critiques and automatic adjustments of requests as a way to search instead of the explicit formulation of requests. Critiques therefore are an important source of interactions between the individual and the system. We can suppose, for instance, that a client who frequently criticizes a particular aspect such as the price attaches greater importance to this concept and wishes the system considers it for the rest of the session. We can also suppose that the order in which critiques are made is also an indicator of the client’s short term requirements. Instead, we have analyzed the usage of a subtler mechanism that could appear in a more transparent way to the user. This heuristic first supposes that the global similarity of a case with the request is expressed as a combination of local similarities between attributes as well as with weights indicating the relative importance of those attributes. This is the case in many CBR systems, including those using the classical *nearest neighbors* method. It is also the case with the “Case Retrieval Nets” technique [3], popular in electronic commerce CBR applications.

Giving this informal definition is useful at this point: the *deficiency* of an attribute measures to what degree the system considers this attribute to be susceptible to a user critique. The proposed heuristic can then be expressed as such: *if the client does not criticize the most deficient of the considered cases, then the importance of the most deficient aspects must decrease and the importance of the criticized aspect must increase*. The intuitive justification behind this heuristic is that clients have a tendency to first criticize the aspects to which they attach more importance. Therefore, we suppose the user always criticize the most important aspect for him that has not yet been optimized. Hence, if the user does not criticize the aspect the system expects to (i.e., the most deficient), then the relative importance of the concepts as maintained by the system is incorrect and must be updated. In fact, if the relative importance of the attributes more deficient than the one being criticized had been lower than the importance of that one, then the proposed result would have been likelier to satisfy the client request.

We now give a short mathematical formalization of the proposed heuristic, in the context of the nearest neighbors method. According to this method, the *global similarity* between a case c and the current request q is expressed by:

$$S = \sum_{i=1}^K \omega_i S_i(q_i, c_i) \quad \text{where} \quad \omega_i \geq 0 \forall i \quad \text{and} \quad \sum_{i=1}^K \omega_i = 1$$

where S is the global similarity, K is the number of parameters of each case, ω_i is the weight given to the parameter i , and S_i is the measure of the global similarity between the parameter i of the request and the one of the considered case.

Local similarity measures largely depend on the application domain, but they all have the same use: to return an estimate between 0 and 1 indicating the similarity between a particular attribute of a case and its equivalent in the request. Knowing the nature of used local similarity measures is not useful for this analysis, because the

approach is general and does not depend nor influences these measures. These measures are considered to be “scientific” references that indicate the absolute similarity between two attributes. We shall see some examples of local similarity measures in the next section.

Here, the ω_i weights represent the relative importance of the concepts. These are the quantities that we want to modify when the user does not criticize the most deficient attribute. Following a critique, the first task of the system consists of computing the deficiency of each attribute. We introduce a mathematical definition of the *deficiency* D :

$$D_i = \omega_i(1 - S_i(q_i, c_i))$$

The reader will notice that the bigger the weight and the weaker the local similarity, the higher the deficiency of the attribute. Once the deficiency of each parameter i is computed, we cover one by one each parameter whose deficiency is higher than the criticized parameter, which we call the *critical deficiency* D_* . For all those parameters, we reduce their weight by a value equal to their deficiency:

$$\omega_i \leftarrow \omega_i - (D_i - D_*) \quad \forall i \quad D_i > D_*$$

Finally, there had been at least one parameter whose deficiency is higher than the critical deficiency. The weight of the criticized parameter finds itself increased by an amount equivalent to the sum of all reductions subjected by the weights of the parameters that lost importance, such as the sum of all weights stays equal to the unit:

$$\omega_* \leftarrow \omega_* + \sum_i (D_i - D_*) \quad \forall i \quad D_i > D_*$$

The proposed heuristic automatically modifies the weights as the user criticizes the products, so as to obtain an increasingly faithful representation of his short-term profile. This, of course, supposes that at each time the client always criticizes the attribute with which he is the least satisfied. In that case, the method offers a way to accelerate the convergence toward a recommendation that can fulfill the needs of the client. Moreover, the adjustment of weights can be useful when the successive critiques of the client always lead him in the same dead-end, that is in an iteration where the system cannot find any product matching the critiques formulated to date. In that case, the client must explicitly reinitiate a request, but he still profits from the adjusted weights according to his requirements, so that we can expect him to find faster than the first time a subset of interesting products.

In the next section, we will instead see an example of an application of the method explained here.

3 Application Example

In order to illustrate the method proposed with a concrete example, we have implemented a prototype system for the recommendation of travel packages. For this,

we will use the “travel agency” case base, freely available from the AI-CBR site [5]. The cases contained in that base come from a real application: the “Virtual Travel Agency”. The case base consists of 1470 predefined travel packages (i.e., non-configurable), each of them described by about ten attributes such as the type (bathing, skiing, etc.), the price, the duration, the destination and others. For our example, we chose the following five attributes: the type, the region, the month, the duration and the price. For each attribute, we have defined a simple measure of local similarity that produces a value between 0 and 1. The measures for nominal attributes (type, region, month) are tables indicating similarities between all possible combinations. An example can be found in Figure 1.

Query / Case	“bathing”	“city”	“recreation”	“skiing”	...
“bathing”	1.0	0.2	0.5	0.1	...
“city”	0.2	1.0	0.3	0.2	...
“recreation”	0.8	0.3	1.0	0.8	...
“skiing”	0.1	0.2	0.5	1.0	...
...	...	...	...	...	...

Fig. 1. Local similarity measure for the “type” attribute As for numerical quantities such as duration and price, we use standard distance measures (i.e., Euclidean) that we normalize between 0 and 1

Figure 2 illustrates the graphical interface of the application. In the upper-left corner, the user can enter explicit requests used as starting points to the recommendation process. We can specify the desired product characteristics, and we can see at each moment the actual weights. Let us note that in our application, only the automatic weight refinement mechanism can modify those values. In a real application however, it would be interesting to allow the user to override this functionality. The upper-right part of the interface is the location where is displayed the product currently being considered as well as its parameters. It is also from there that the client can formulate its critiques of these parameters, such as “cheaper” for the price, “sooner” or “later” for the month, etc. Finally, the lower part continually displays the ten cases judged the most likely to respond to the client’s needs according to the last request (or critique) made. We note that following a request or a critique, the system displays by default as the candidate product the one that obtains the highest score (the highest global similarity). But the client may select and criticize other products in the pool of possibilities if he so desires. That table exists specifically to give the user a broader choice.

Short-Term Profiling CBR

Querying

Feature

Value

Weight

Type

Language

0.2

Region

Egypt

0.2

Month

March

0.2

Duration

12

0.2

Price

2000

0.2

Reset

Submit

Tweaking

Current case

Tweaks

Case #

11

Not my type

Type

City

Elsewhere

Region

Cairo

Sooner

Month

April

Later

Duration

7

Shorter

Price

1978

Longer

Score

0.890

Cheaper

Results

Case #	Type	Region	Month	Duration	Price	Score
11	City	Cairo	April	7	1 978	0,890
16	City	Cairo	April	7	2 308	0,878
738	Education	Turk. Riviera	May	14	2 698	0,852
12	City	Cairo	May	7	1 898	0,850
720	Education	Turk. Aegean ...	May	14	2 878	0,845
1 005	Wandering	Thuringia	March	14	855	0,840
982	Wandering	Black Forest	March	14	1 033	0,840
324	Education	Wales	June	14	1 808	0,840
293	Education	Crete	May	14	3 038	0,838
309	Education	England	June	14	2 068	0,837

Fig. 2. The application graphical interface

As an example, let us imagine the system is in the state illustrated in Figure 2 and the user wishes to criticize the month: “sooner”. Figure 3. represents the internal state or the system before and after the formulation of this critique.

Attribute	Weight	Local similarity	Deficiency
Type	0.2	0.8	0.04
Region	0.2	1.0	0.0
Month	0.2	0.9	0.02
Duration	0.2	0.75	0.05
Price	0.2	1.0	0.0

Attribute	Weight	Local similarity	Deficiency
Type	0.18	0.8	0.04
Region	0.2	1.0	0.0
Month	0.25	0.9	0.02
Duration	0.17	0.75	0.05
Price	0.2	1.0	0.0

Fig. 3. System state before and after the “sooner” critique

Case #	Type	Region	Month	Duration	Price	Score
1 005	Wandering	Thuringia	March	14	855	0,857
982	Wandering	Black Forest	March	14	1 033	0,857
928	Recreation	Erz Oebirge	March	14	838	0,839
1 001	Wandering	Thuringia	March	7	389	0,831
999	Wandering	Thuringia	March	7	598	0,831
988	Wandering	Thuringia	March	7	1 368	0,831
970	Wandering	Black Forest	March	7	1 257	0,831
1 450	Recreation	Turk. Riviera	March	14	2 238	0,829
1 365	Recreation	Fuerteventura	March	14	2 308	0,827
963	Wandering	Black Forest	March	14	2 768	0,826

Fig. 4. Proposed results after the “sooner” critique

We note that the criticized attribute, the month, was not the most deficient at the time the critique was formulated. Two other attributes had a higher deficiency: the travel type and the duration. Since the user chose to criticize the month, we deduce this parameter is more important to him and we increase its value at the expense of the type and the duration. In Figure 4, we can see what the interface shows after the critique. The results are more in line with the month constraint, not only because the critique had the effect of filtering all months that did not satisfy “sooner”, but also due to the increase of the importance of this concept. The pertinence of the other less important parameters, particularly the type, becomes more arbitrary. Curiously, the duration seem more similar than before yet their importance has diminished. This is only a coincidence due to the fact that travels less similar according to the type happen to be more similar according to the duration.

We have made various trials using this application and we could notice that if, at each moment, we always modified the parameter we considered the most unsatisfactory, then the recommendation system allowed us to find an adequate solution faster with our approach than if the weights were fixed. If we base ourselves on the few thirties of examples we tested, we can roughly estimate a performance gain of 20 to 40 percent on the convergence speed toward an acceptable solution. Of course, that is only an estimate and quantifying the advantages of this heuristic more precisely would require in-depth experiments.

4 Conclusion

In this work, we began by raising the problem of the lack of customer support in electronic commerce applications on the Internet. We then proposed a heuristic

allowing CBR-based recommendation systems to take into account short-term preferences of clients. This heuristic is based on the principle that if a client does not criticize the attribute of a product the system considers the most deficient, then the relative importance of the attributes as represented by the system would benefit from being corrected. Using this method, a CBR system does not react exactly the same way from one session to the other. It can now take into account the fact that according to the person or the situation, the relative importance of the different characteristics may vary. The simplicity, the maintenance-free operation and the automatic nature of the profile creation are three of the principal advantages of the proposed approach. Also, the fact that this heuristic can be combined with long-term modeling methods makes it an ideal candidate for hybridization. However, the method has several limits at this point, which is why a more detailed study would be required before its validity can be definitely ascertained. In particular, better mathematical formalization as well as experimental tests are certainly required.

References

1. Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1), 39-59.
2. Bartsch-Spörl, B., Lenz, M. and Hübner, A. (1999). Case-Based Reasoning – Survey and Future Directions. In *Proceedings of the 5th Biannual German Conference on Knowledge-Based Systems (XPS-99)*, Springer-Verlag.
3. Lenz, M. and Burkhard, H.D. (1996). Case Retrieval Nets : Basic Ideas and Extensions. In *Proceedings of the 4th German Workshop on Case-Based Reasoning (GWCBR-96)*, Berlin.
4. Wilke, W., Lenz, M. and Wess, S. (1998). Intelligent Sales Support with CBR. In Lenz, M. et al. (editors), *Case-Based Reasoning Technology – From Foundations to Applications*, Springer-Verlag.

URL links

5. **AI-CBR** <http://www.ai-cbr.org/theindex.html>

K-SVCR. A Multi-class Support Vector Machine

Cecilio Angulo¹ and Andreu Català²

¹ Dept. of Systems Engineering, Polytechnical University of Catalonia

E-08222 Terrassa, Spain

cangulo@esaii.upc.es

² LEA-SICA, European Associated Lab. Intelligent Systems and Advanced Control

E-08800 Vilanova i la Geltrú, Spain

andreu@esaii.upc.es

Abstract. Support Vector Machines for pattern recognition are addressed to binary classification problems. The problem of multi-class classification is typically solved by the combination of 2-class decision functions using voting scheme methods or decision trees. We present a new multi-class classification SVM for the separable case, called K-SVCR. Learning machines operating in a kernel-induced feature space are constructed assigning output +1 or -1 if training patterns belongs to the classes to be separated, and assigning output 0 if patterns have a different label to the formers. This formulation of multi-class classification problem ever assigns a meaningful answer to every input and its architecture is more fault-tolerant than standard methods one.

1 Introduction

The problem of multi-class classification from examples addresses the general problem of finding a decision function f , approximation of an unknown function $\hat{f}$, defined from an input space Ω into an unordered set of classes $\{\theta_1, \dots, \theta_K\}$, given a training set

$$\mathcal{T} = \{(\mathbf{x}_p, y_p = f(\mathbf{x}_p))\}_{p=1}^{\ell} \subset \Omega \times \{\theta_1, \dots, \theta_K\}. \quad (1)$$

Support Vector Machines (SVMs) that learn classification problems - in short SVMC -, are specific to binary classification problems, also called dichotomies. The problem of multi-class classification ($K \geq 2$) is typically solved by the combination of 2-class decision functions.

In this paper we present a new multi-class classification SVM for the separable case, called K-SVCR. When $K \geq 2$, we will construct learning machines assigning output +1 or -1 if training patterns belongs the classes to be separated, and output 0 if patterns belongs a different class to the formers. So, we are forcing the computed separating hyperplane to cover all the '0-label' training patterns. Like in the construction of SVMs, the new method exploits the basic idea of map the data from the input space Ω into some other higher dimension dot product space $\mathcal{F}$, called feature space, via a non linear map and perform the above linear algorithm in $\mathcal{F}$. The associated restricted QP-problem could be

subject to

$$\begin{aligned} \alpha_i &\geq 0, \quad i = 1, \dots, \ell \\ \sum_{i=1}^{\ell} \alpha_i y_i &= 0. \end{aligned} \quad (8)$$

The hyperplane decision function can thus be written as

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^{SV} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right), \quad (9)$$

where b is computed using the Karush-Kuhn-Tucker complementary conditions

$$\alpha_i \cdot [y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - 1] = 0, \quad i = 1, \dots, \ell. \quad (10)$$

Among all the training patterns, only a few of them have an associated weight α_i non-zero in the expansion (9). These elements lie on the margin - some strict constraint in (6) is accomplished - and they are called support vectors.

To generalize the SV algorithm to regression estimation, an analogue of the margin is constructed in the space of the target values - $y \in \mathbb{R}$ - by using Vapnik's ε -insensitive loss function

$$|y - f(\mathbf{x})|_{\varepsilon} \stackrel{\text{def}}{=} \max \{0, |y - f(\mathbf{x})| - \varepsilon\}. \quad (11)$$

For a priori chosen $\varepsilon \geq 0$, the associated constrained optimization problem for the separable case is

$$\arg \min \tau(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_{\mathcal{F}}^2 \quad (12)$$

subject to

$$\begin{aligned} (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - y_i &\leq \varepsilon, \quad i = 1, \dots, \ell \\ y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) &\leq \varepsilon, \quad i = 1, \dots, \ell. \end{aligned} \quad (13)$$

Introducing Lagrange multipliers, we arrive at the constrained optimization problem: find multipliers $\alpha_i, \alpha_i^* \geq 0$ which

$$\begin{aligned} \min W(\alpha, \alpha^*) &= \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}_j) (\alpha_j^* - \alpha_j) + \\ &+ \varepsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) \end{aligned} \quad (14)$$

subject to

$$\alpha_i, \alpha_i^* \geq 0, \quad i = 1, \dots, \ell \quad (15)$$

$$\sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0.$$

The regression estimate takes the form:

$$f(\mathbf{x}) = \sum_{i=1}^{SV} (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}) + b. \quad (16)$$

The solution expands again in terms of a subset of the training patterns, and b is calculated from (13) in strict equal form over the support vectors.

3 Multi-class Support Vector Machines

The standard method of decomposing a general classification problem into dichotomies is to place K binary classifiers in parallel. In the original method [3,10], the i th SVMC is trained with positive labels for all the examples in the i th class, and negative labels for all other examples. We refer to SVMs trained in this way as 1- v - r SVMCs - short for one-versus-rest -. The training time of the standard method scales linearly with K .

The output scale of a SVM is determined so that the separating hyperplane is in canonical form, i.e., support vector output is ± 1 . In [6] is asserted that this scale is not robust, since it depends on just a few points, often including outliers, and different alternatives are proposed to circumvent this problem

Another general method to construct multi-class classifiers is to build all possible binary classifiers - $K \cdot (K-1)/2$ hyperplane decision functions - from a training set of K classes, each classifier being trained on only two out of K classes. We refer to the SVMCs trained with this method like 1- v -1 SVMCs - short for one-versus-one -. The combination of these binary classifiers to determine the label assigned to each new input can be made by different algorithms, for example the voting scheme [4]. The 1- v -1 approach is, in general, preferable to the 1- v - r one [5]. Unfortunately, the size of the 1- v -1 classifier may grow superlinearly with K .

In addition to these two general methodologies, it is possible to construct multi-class classifiers combining 1- v -1 SVMCs with decision trees, that are able to handle many classes. In [8] a learning architecture is presented, the DAGSVM algorithm, which operates in a kernel-induced feature space and uses 2-class maximal margin hyperplanes at each decision-node of the Decision Directed Acyclic Graph (DDAG). The class of functions implemented naturally generalizes the class of decision trees.

In [1] the relationship between SVMC and a family of mathematical programming methods (MPM) are examined and a new method for nonlinear discrimination, the Support Vector Decision Tree (SVDT), is generated. It constructs decision trees in which each decision is a support vector machine. In this sense, the architecture method is similar to the DAGSVM algorithm.

Working in a different way, in [11] the original SVMC constrained optimization problem is redefined and generalized to construct a decision function by

considering all classes at once. The K -SVCR multi-class classification method is also defined in this sense, the constrained QP problem is redefined, but we are not considering the classification of all classes at once. In the other hand, it is possible to make an extension of our algorithm to capture the advantageous properties of the DAGSVM algorithm.

4 K -SVCR Learning Machine

Given the training set $\mathcal{T}$ defined in (1) we would find a decision function f in the form (3) with:

$$\begin{aligned} f(\mathbf{x}_p) &= +1, & p = 1, \dots, \ell_1 \\ &= -1, & p = \ell_1 + 1, \dots, \ell_1 + \ell_2 \\ &= 0, & p = \ell_1 + \ell_2 + 1, \dots, \ell, \end{aligned} \quad (17)$$

where, without loss of generality, we suppose the first $\ell_{12} = \ell_1 + \ell_2$ patterns corresponding to the two classes to be separated, and the other patterns ($\ell_3 = \ell - \ell_{12}$) belonging to any different class - we will label them with 0 -.

Obviously, in general, do not exist any hyperplane accomplishing the constraints (17) in the input space Ω , and hence is useless looking for a linear solution to the problem in this space. But, if we insert this space via a nonlinear map into a feature space with a dimension high enough, the hyperplane capacity to accomplish the constrains increase, and it will be possible to find a solution.

For instance, when we solve the QP problem leading to the SVMC solution it is very usual to formulate the problem with $b = 0$, which is equivalent to require that all hyperplanes contain the origin. This is considered a mild restriction for high dimensional spaces, since it is equivalent to reduce the number of degrees of freedom by one [2].

The requirement of the K -SVCR learning machine is higher. It requires that optimal hyperplane contains all ℓ_3 training patterns with label 0.

We define below the constrained optimization problem associated to K -SVCR method, for the separable case: for $0 \leq \delta < 1$ chosen a priori,

$$\arg \min \tau(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|_{\mathcal{F}}^2 \quad (18)$$

subject to

$$\begin{aligned} y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - 1 &\geq 0, & i = 1, \dots, \ell_{12} \\ \langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b &\leq \delta, & i = \ell_{12} + 1, \dots, \ell \\ \langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b &\geq \delta, & i = \ell_{12} + 1, \dots, \ell, \end{aligned} \quad (19)$$

with a decision function solution similar to (3), defined by

$$\begin{aligned} f(\mathbf{x}) &= +1, & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle_{\mathcal{F}} + b > \delta \\ &= -1, & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle_{\mathcal{F}} + b < \delta \\ &= 0, & \text{otherwise.} \end{aligned} \quad (20)$$

If $\delta = 0$ then decision function (20) is the same as (3) and we are exactly requiring that the separating hyperplane will contains the last ℓ_3 training patterns. Nonetheless, this imposition implies no generalization for the '0-label class', no sparsity in the support vectors set over the training patterns with label 0 [9], and higher computational cost. So, even if our task is learning pattern recognition, it could seems that we make a certain use of the ε -insensitive loss function (11) employed in the SVMR method for the output $y_i = 0$.

A solution for the problem defined in (18) and (19) can be found by locating the saddle point of the Lagrangian

$$\begin{aligned} L(\mathbf{w}, b, \alpha, \beta, \beta^*) = & \frac{1}{2} \|\mathbf{w}\|_{\mathcal{F}}^2 - \sum_{i=1}^{\ell_{12}} \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - 1] + \\ & + \sum_{i=\ell_{12}+1}^{\ell} \beta_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - \delta] - \\ & - \sum_{i=\ell_{12}+1}^{\ell} \beta_i^* [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{F}} + b) - \delta] \end{aligned} \quad (21)$$

with constraints

$$\begin{aligned} \alpha_i &\geq 0, \quad i = 1, \dots, \ell_{12} \\ \beta_i, \beta_i^* &\geq 0, \quad i = \ell_{12} + 1, \dots, \ell. \end{aligned} \quad (22)$$

which has to be maximized with respect to the dual variables α_i and β_i, β_i^* and minimized with respect to the primal variables $\mathbf{w}$ and b . In the saddle point the solution should satisfy the conditions, leading to

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{\ell_{12}} \alpha_i y_i \mathbf{x}_i - \sum_{i=\ell_{12}+1}^{\ell} (\beta_i - \beta_i^*) \mathbf{x}_i \\ 0 &= \sum_{i=1}^{\ell_{12}} \alpha_i y_i - \sum_{i=\ell_{12}+1}^{\ell} (\beta_i - \beta_i^*). \end{aligned} \quad (23)$$

Finally, if we define

$$\begin{aligned} \gamma_i &= \alpha_i y_i, \quad i = 1, \dots, \ell_{12} \\ \gamma_i &= \beta_i, \quad i = \ell_{12} + 1, \dots, \ell \\ \gamma_i &= \beta_{i-\ell_3}^*, \quad i = \ell + 1, \dots, \ell + \ell_3 \end{aligned} \quad (24)$$

the primal variables are eliminated and we arrive at the Wolfe dual of the optimization problem: for $0 \leq \delta < 1$ chosen a priori

$$\arg \min L(\gamma) = \frac{1}{2} \gamma^T \cdot \mathbf{H} \cdot \gamma + \mathbf{c}^T \cdot \gamma \quad (25)$$

with

$$\begin{aligned}\gamma^T &= (\gamma_1, \dots, \gamma_\ell, \gamma_{\ell+1}, \dots, \gamma_{\ell+\ell_3}) \in \mathbb{R}^{\ell_{12}+\ell_3+\ell_3} \\ \mathbf{c}^T &= \left(\frac{-1}{y_1}, \dots, \frac{-1}{y_{\ell_{12}}}, \delta, \dots, \delta \right) \in \mathbb{R}^{\ell_{12}+\ell_3+\ell_3} \\ \mathbf{H} &= \begin{pmatrix} (k(\mathbf{x}_i, \mathbf{x}_j)) & -(k(\mathbf{x}_i, \mathbf{x}_j)) & (k(\mathbf{x}_i, \mathbf{x}_j)) \\ -(k(\mathbf{x}_i, \mathbf{x}_j)) & (k(\mathbf{x}_i, \mathbf{x}_j)) & -(k(\mathbf{x}_i, \mathbf{x}_j)) \\ (k(\mathbf{x}_i, \mathbf{x}_j)) & -(k(\mathbf{x}_i, \mathbf{x}_j)) & (k(\mathbf{x}_i, \mathbf{x}_j)) \end{pmatrix} = \mathbf{H}^T \in \mathcal{S}(\mathbb{R}^{\ell+\ell_3}),\end{aligned}\tag{26}$$

subject to

$$\begin{aligned}\gamma_i \cdot y_i &\geq 0, \quad i = 1, \dots, \ell_{12} \\ \gamma_i &\geq 0, \quad i = \ell_{12}, \dots, \ell + \ell_3 \\ \sum_{i=1}^{\ell_{12}} \gamma_i &= \sum_{i=\ell_{12}+1}^{\ell} \gamma_i - \sum_{i=\ell+1}^{\ell+\ell_3} \gamma_i.\end{aligned}\tag{27}$$

The hyperplane decision function can be written as

$$\begin{aligned}f(\mathbf{x}) &= +1, \quad \text{if } \sum_{i=1}^{SV} \nu_i k(\mathbf{x}_i, \mathbf{x}) + b > \delta \\ &= -1, \quad \text{if } \sum_{i=1}^{SV} \nu_i k(\mathbf{x}_i, \mathbf{x}) + b < \delta \\ &= 0, \quad \text{otherwise}\end{aligned}\tag{28}$$

where

$$\begin{aligned}\nu_i &= \gamma_i, \quad i = 1, \dots, \ell_{12} \\ \nu_i &= \gamma_{i+\ell_3} - \gamma_i, \quad i = \ell_{12} + 1, \dots, \ell,\end{aligned}\tag{29}$$

and b is calculated from (19) in strict equal form over the support vectors in terms of parameters γ_i . We observe that the third constraint in (27) can be written as

$$\sum_{i=1}^{SV} \nu_i = 0.\tag{30}$$

This formulation of multi-class classification problem is more fault-tolerant than the 1- v -r general method, because there exist more redundancy in the answers [7]. On the other hand, all the K -SVCRs answers have sense: each machine classifies any input into a class, the two class implicated in the binary classification or into the 'rest' class (0-label class). The 1- v -1 general classification method is more fault-tolerant than the 1- v -r one, but the classifiers give no sense answers if the evaluated input does not belong to the classes implicated in the binary classification.

5 Conclusions and Further Research

The K -SVCR algorithm, a novel learning machine based in SVMs for multi-class pattern recognition for the separable case is presented. This algorithm construct a decision function to separate two classes containing the patterns of all the others classes. These 1- v -1 SVMs can be combined in an "AND" scheme, in a voting scheme or in a decision tree formulation. Two initial schemes are easily implemented, meanwhile the last formulation is part of our actual study, employing a DDAG architecture to reduce the evaluation time and control the generalization performance.

Further research involves the test of the method on large data sets and a more detailed comparison with other methods over real data benchmarks.

A generalization of the K -SVCR procedure for the non-separable case is being developed in the present, and future work will establish a comparison between the generalized algorithm and a modification over the sensitivity parameter for the present formulation.

References

1. Bennett, K.P.: Combining Support Vector and Mathematical Programming Methods for Classification. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.): *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA (1999) [34](#)
2. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining And Knowledge Discovery* **2** (1998) 1-47 [32](#), [35](#)
3. Cortes, C., Vapnik, V.: Support Vector Networks. *Machine Learning* **20** (1995) 273-297 [32](#), [34](#)
4. Friedman, J.H.: Another Approach to Polychotomous Classification. Tech. Report, Dept. Statistics, Stanford Univ. (1996) [34](#)
5. Kreßel, U.: Pairwise classification and support vector machines. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.): *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA (1999) [34](#)
6. Mayoraz, E., Alpaydin, E.: Support Vector Machines for Multi-class Classification. Proc. 5th IWANN, Alicante, Spain (1999) 833-842 [34](#)
7. Moreira, M., Mayoraz, E.: Improved Pairwise Coupling Classification with Correcting Classifiers. In: Nédellec, C., Rouveirol, C. (eds.): *Lecture Notes in Artificial Intelligence*, Vol. 1398. Springer-Verlag, Berlin Heidelberg New York (1998) 160-171 [37](#)
8. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large Margin DAGs for Multiclass Classification. *Advances in NIPS* **12**, MIT Press, Cambridge, MA (2000) To appear [34](#)
9. Smola, A.J.: Learning with Kernels. Doctoral thesis. TU Berlin (1998) [32](#), [36](#)
10. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer Verlag, Berlin Heidelberg New York (1995) [32](#), [34](#)
11. Weston, J., Watkins, C.: Multi-class Support Vector Machines. Tech. Report CSD-TR-98-04, Royal Holloway, Univ. of London, Egham, UK (1998) [34](#)

Learning Trading Rules with Inductive Logic Programming

Liviu Badea

AI Lab, National Institute for Research and Development in Informatics
8-10 Averescu Blvd., Bucharest, Romania
badea@ici.ro

Abstract. We apply Inductive Logic Programming (ILP) for inducing trading rules formed out of combinations of technical indicators from historical market data. To do this, we first identify ideal trading opportunities in the historical data, and then feed these as examples to an ILP learner, which will try to induce a description of them in terms of a given set of indicators. The main contributions of this paper are twofold. Conceptually, we are learning strategies in a chaotic domain in which learning a predictive model is impossible. Technically, we show a way of dealing with *disjunctive* positive examples, which create significant problems for most inductive learners.

1 Introduction and Motivation

Stock market prices are inherently chaotic and unpredictable. They are generated by a large number of time-dependent processes and are therefore non-stationary. Trying to induce a predictive model of the market evolution is thus bound to failure, mostly because any regularity would be immediately exploited and broken.

As long as the evolution of the market cannot be predicted, we cannot directly apply an inductive learning algorithm on the market data and hope to obtain a predictive model. However, although we are incapable of prediction, we may interact with the market (using suitable trading rules) and still make money.

We should therefore concentrate on inducing trading rules rather than predictive models. It is exactly this aspect that makes learning trading rules a challenging domain. Viewed more abstractly, we are dealing with an agent involved in an interaction (a sort of game) with an environment whose evolution cannot be predicted from past records. The challenge consists in devising (inducing) a winning strategy, despite the fact that the environment is unpredictable. For example, a profitable trading strategy is to buy at (local) minima and sell at (local) maxima. The difficulty consists in detecting such extrema by looking only into the past.

But what is the difference between predicting the next value of the price time-series and finding a profitable strategy? For one thing, the ability to predict (the next value) entails a profitable strategy, as follows. Roughly speaking, if we can predict the next value to go up (down), then it is profitable to BUY (respectively SELL). If we do not expect a significant increase or decrease, we should simply wait.

On the other hand, a strategy saying to buy (sell) will probably predict an increase (decrease). The key difference manifests itself when the strategy says to wait, case in which it either predicts a more or less stationary value, or it is *unable to predict*.

Strategies are therefore *partial* predictive models. They can be profitable although at times they may be unable to predict.

We mentioned the fact that trading at (local) extrema is a profitable strategy. But how do we detect such extrema only from past data? A large number of so called *technical analysis indicators* [1] are usually employed for this purpose. Roughly speaking, there are two categories of such technical indicators: trend-following indicators, as well as indicators employed in choppy and sideways-moving (non-trending) markets. *Trend-following indicators*, such as moving averages, aim at detecting longer (or shorter) term trends in the price time series, usually at the expense of a longer (respectively shorter) response delay. Therefore they are called “lagging indicators”. We can detect local extrema in trending markets by studying the crossings of two moving averages of different averaging lengths. In the case of sideways moving markets, trend-following indicators will usually produce losses. Other indicators, like stochastic oscillators for example, are used instead. If we could discriminate trending markets from non-trending ones, we could apply indicators suited to the specific market conditions. Discriminating between trending and non-trending markets is however difficult. Indicators like the average directional movement index (ADX) are sometimes employed for this purpose.

Using the indicators appropriate for the specific market conditions, possibly as filters, has proved a crucial but extremely difficult task, which has been approached mostly by empirical means. While tuning the numerical parameters of the indicators occurring in trading rules can be done automatically, this is only a first step in the process of adapting a set of indicators to a given market. Finding the most appropriate *combinations* of indicators is in certain ways more interesting, although it is also more complicated due to possible combinatorial explosions.

This paper applies Inductive Logic Programming (ILP) for inducing trading rules formed out of combinations of technical indicators from historical market data. To do this, we first identify buy and sell opportunities in the historical data (by looking not only into the past, but also at future time points).¹ These buy/sell opportunities are then given as examples to an ILP learner which will try to induce a description of these trading opportunities in terms of a given set of technical indicators. For an appropriate set of indicators, the induced trading rules may be profitable in the future as well.

Our main departure from other approaches to the problem of inducing trading rules consists in the fact that whereas other approaches simply test the profitability of syntactically generated strategy variants, we are focusing on recognising the ideal trading opportunities (such as local extrema). Our strategy may prove more reliable, since it may be less influenced by contingent fluctuations in the historical data used for training (because we are explicitly concentrating on the key aspects of a successful strategy, such as recognising extrema). On the other hand, looking just at the profit of a particular candidate strategy on the historical data without analysing its behaviour in more detail may not represent a guarantee for its profitability in the future.

2 Identifying Positive and Negative Examples

Most approaches to learning trading rules guide the syntactical generation of candidate rule sets by global performance criteria, such as profit or risk. Our approach, on the other

¹ If, at a given time point we knew not only the past, but also the future, then we could easily make money by buying at minima and selling at maxima. The future is normally unpredictable and therefore we cannot detect extrema without a delay. But we can easily do this for a given historical time series.

hand is more selective: we first label the historical time series with ideal BUY/SELL opportunities by taking into account both the past *and the future*. These will be subsequently given as examples to an inductive learner. This approach allows us to have a tighter control on the performance of the rules and to avoid selecting rules just because they are profitable on the historical data. Their profitability may depend on contingencies of the historical data and may not guarantee the profitability in the future. Trying to recognise ideal trading opportunities (determined in advance by the initial labelling process) may be more selective, and therefore have better chances to generalise to unseen data. Let us describe the labelling process in more detail.

2.1 Positive Examples

Since we cannot predict *global* extrema just from past data, we shall consider the *local* extrema as ideal trading opportunities. But unfortunately, even these can be too difficult targets for trading strategies based on predetermined sets of technical indicators. Small deviations from the local extrema (in terms of price and time) are not critical from the point of view of profitability and can be tolerated. Therefore we shall consider the points “around” the local extrema as positive examples of trading opportunities. Additionally, we need to avoid declaring the local extrema at the finest time-scales as potential learning targets. We could deal with eliminating such small-scale fluctuations by not distinguishing among the points in a price band of a given width ϵ . More precisely, we define the ϵ -band of a given time point t to be the set of time points preceding t that all stay within a band of height ϵ : $\epsilon\text{-band}(t) = \{t' \leq t \mid \forall t'', t' \leq t'' \leq t \rightarrow |y(t') - y(t'')| < \epsilon\}$.

Instead of targeting the precise local extremum t_{extr} , we shall target one of the points in its ϵ -band: $t \in \epsilon\text{-band}(t_{extr})$. If one such t is covered, then no other time point of the ϵ -band needs to be covered any more. Thus, instead of a conjunctive set of examples of the form $buy(t_{min}^{(1)}) \wedge sell(t_{max}^{(2)}) \wedge buy(t_{min}^{(3)}) \wedge \dots$ (where $t^{(i)}$ are the *exact* local extrema), we will end up with disjunctive examples like $(buy(t_1^{(1)}) \vee buy(t_2^{(1)}) \vee \dots) \wedge (sell(t_1^{(2)}) \vee sell(t_2^{(2)}) \vee \dots) \wedge \dots$ where $t_j^{(i)}$ are the time points in the ϵ -band of the local extremum $t^{(i)}$.

This *disjunctive* nature of the positive examples creates significant problems for most inductive learners. There seems to be no direct way of dealing with such examples in decision tree learning. ILP has also problems when faced with such examples, unless we are learning general clausal theories (i.e. we are not confined to Horn clauses).

The greater flexibility of ILP (as compared to ID3) allows us to represent such examples as $do(buy, [t_1^{(i)}, t_2^{(i)}, \dots])$ and to look for candidate hypotheses of the form

$$\begin{aligned} do(buy, TimeList) :- & \text{member}(T1, TimeList), \text{indicator1}(T1), \\ & \text{member}(T2, TimeList), \text{indicator2}(T2), \dots \end{aligned} \quad (1)$$

where *TimeList* is an ϵ -band of the current time point. Such a rule is unfortunately useless by itself when it comes to suggesting the next action at a given time point T , because it does not explicitly refer T . Fortunately, we can easily compute the ϵ -band of a given time point T (since the ϵ -band refers only past data) and use it as an argument to the above *do/2* predicate:

$$do\_at(Action, T) :- \text{epsilon\_band}(T, TimeList), do(Action, TimeList).$$

There is another subtle point regarding the precise form of the candidate hypotheses (1). Roughly speaking, they activate a signal if a *combination* of indicators gets activated in the ϵ -band of the current time point. The indicators need *not* be activated all at *exactly* the same time point T , as they are in the following more restrictive rule:

$$do(buy, TimeList) :- member(T, TimeList), indicator1(T), indicator2(T), \dots \quad (2)$$

For example it may be that a certain combination of two crossover systems most reliably signals a trading opportunity. It is however extremely unlikely that those crossovers happen both *exactly at the same time point* T . For all practical purposes, it is sufficient if they both happen in the same ϵ -band, even if not at exactly the same time.

The algorithm for computing the ϵ -bands of the local local extrema (which will be used as positive examples) dynamically maintains an ϵ -band. When a "break-out above" occurs, then the current ϵ -band is marked as a minimum band only if we have entered it from above (i.e. from a previous maximum). Otherwise, it will be treated as an intermediate band (see Figure 1). "Break-outs below" are treated in a similar fashion.

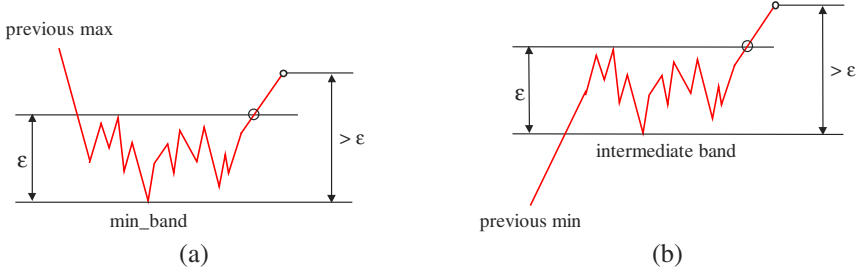


Fig. 1. An ϵ -band "break-out above" can lead to a minimum (a) or an intermediate band (b) depending on the type of the previous extremum: maximum (a) or minimum (b)

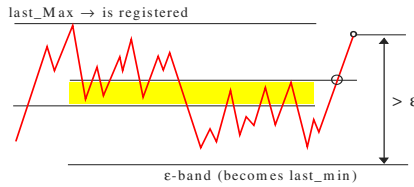


Fig. 2. A break-out above from a minimum-band partially overlapping the previous max-band. The max-band is registered (as a positive example for SELL), but the current min-band is not, because of potential overlaps with the next band (not shown)

Since successive extrema-band price ranges can overlap (leading to potential BUY operations at prices higher than some adjacent SELL), we need to eliminate such overlaps before registering the corresponding extrema bands as positive examples. We also need to delay the registration process of an extremum band $B1$ until the next extremum band $B2$ is computed, because $B1$'s overlap with $B2$ cannot be computed before computing $B2$ (see Figure 2). Additionally, we avoid classifying the very first ϵ -band as either a minimum, or a maximum band.

If the next price $y(t)$ stays within the current ϵ -band (there is no "break-out"), then we simply add it to the current ϵ -band.

2.2 Negative Examples

Learning from positive examples only [4] does not prevent the learner from obtaining bad candidate rules, which can produce significant losses. We therefore also need negative examples, to avoid obtaining overly general rules that might produce such failures. Adopting all intermediate points² as negative examples may be too strict³: a point very close to a maximum band should maybe not be considered a negative example for SELL (unless it is also very close to an adjacent minimum-band). It should however be considered a negative example for BUY, since we are too close to the SELL points in the maximum band (see Figure 3). Although such a point will not be necessarily considered a negative example for SELL (thereby tolerating an induced rule that covers it), the point is not a positive example either, so it will not *trigger* the induction of a rule covering it.

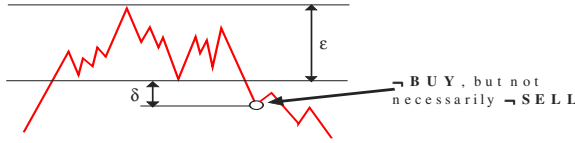


Fig. 3. Points close to a max-band should be negative examples for BUY, but not necessarily for SELL

We generate negative examples according to the following rules

- $\neg \text{do}(\text{buy}, [t])$ if $\min(y[\text{Max\_band}]) - y(t) < \delta$
- $\neg \text{do}(\text{sell}, [t])$ if $y(t) - \min(y[\text{Max\_band}]) < \delta$

for some parameter δ (see also Figure 3). Note that we can have "grey areas" for BUY (SELL), i.e. areas in which we have neither BUY (SELL) nor $\neg \text{BUY}$ (respectively $\neg \text{SELL}$).

Besides the above "normal" situation in which the δ -bands are separated, we can have overlapping δ -bands (when the ϵ -bands are close). In such cases, the intermediate points are negative examples for both BUY and SELL. Of course, we can have not only overlapping δ -bands, but also overlapping ϵ -bands. As previously discussed in more detail, such overlaps are also considered negative examples for both BUY and SELL.

3 Using ILP for Learning from Disjunctive Examples

The first, 'labelling' phase of our algorithm has set up a learning problem with positive examples of the form

$$\begin{aligned} \text{do}(\text{buy}, [3,4,6,7]). & \dots & (3) \\ \text{do}(\text{sell}, [10,11,14]). & \dots \end{aligned}$$

negative examples like

$$\begin{aligned} \neg \text{do}(\text{sell}, [8]). & \dots \\ \neg \text{do}(\text{buy}, [15]). & \dots \end{aligned}$$

² i.e. all points that are neither in minimum, nor in maximum bands.

³ in the same way in which marking the local *exact* extrema as positive examples (targets) for buy/sell actions was too strict. Indeed, it is unlikely not only that an induced rule will cover the local extrema *exactly*, but also that it will succeed avoiding *all* intermediate points.

and a "background theory" containing the historical price time series, as well as predicates for computing various technical indicators and trading strategies. In our experiments we have used the following *technical indicators* on daily closing prices of the USD-DEM exchange rate: moving averages, the Relative Strength Index (RSI), the Average Directional Movement Index (ADX), the stochastic oscillators SlowK and SlowD with an internal averaging period of 3 days, all of these computed for time windows of 5, 10, 15, 25, 40, 65 trading days. These indicators have been used for implementing the following *trading strategies*, which are to be used as "building blocks" in inductive hypotheses:

- a moving average crossover system
- an RSI oscillator system (with buy and sell zones)
- an ADX system triggered by crossing over a given threshold after two successive up-movements
- a stochastic SlowK-SlowD crossover system
- a stochastic oscillator system (with buy and sell zones).

Practically all existing propositional learning algorithms cannot directly deal with disjunctive examples of the form (3). Being a first-order learning technique, Inductive Logic Programming (ILP) can deal with such examples by searching for hypotheses of the form

```
do(buy, TimeList) :- member(T1, TimeList), indicator1(T1),
                    member(T2, TimeList), indicator2(T2), ...
```

As previously mentioned, the various indicators occurring in such a hypothesis need not all refer the same time point T . (Since *TimeList* is the ε -band of the current time point, they can be triggered at different time points of the ε -band.)

The following simplified Progol description⁴ illustrates our approach to learning trading rules using ILP.

```
% Mode declarations
:- modeh(*, do(buy,+time_list))?           :- modeh(*, do(sell,+time_list))?
:- modeb(*, member(-time,+time_list))?     :- modeb(*, member(+time,-time_list))?
:- modeb(*, indicator_sell1(+time))?        :- modeb(*, indicator_sell2(+time))?
:- modeb(*, indicator_buy1(+time))?         :- modeb(*, indicator_buy2(+time))?

% Type declarations
time(T) :- number(T).
time_list([]).      time_list([X|L]) :- time(X), time_list(L).

% Background knowledge
member(X, [X|_]).   member(X, [_|L]) :- member(X, L).

indicator_sell1(2). indicator_sell1(6). indicator_sell1(11). indicator_sell1(16).
indicator_sell2(3). indicator_sell2(7). indicator_sell2(13). indicator_sell2(17).
indicator_buy1(5).  indicator_buy1(9).  indicator_buy1(15). indicator_buy1(20).
indicator_sell1(21). indicator_sell2(22).

% Positive examples
do(sell, [1,2,3]). do(sell, [6,7]). do(sell, [11,12,13]). do(sell, [16,17]).
do(buy, [4,5]). do(buy, [8,9,10]). do(buy, [14,15]). do(buy, [18,19,20]).

% Negative examples
:- do(_, [21]). :- do(_, [22]). :- do(_, [23]).
```

The *modeh* declarations describe the atoms allowed in the head of the induced rules, while *modeb* describe the atoms allowed in the body. It is essential that the recall of the member body atom be unlimited (*), because the lengths of the time lists in the positive examples

⁴ the simplification amounts to explicitly enumerating the indicators at the various time points instead of computing them using background clauses.

can be arbitrary. (If the recall would be set to 1, then only the indicators triggered at the *first* time point of each time-list (of each positive example) would be included in the most specific clause and the search would be incomplete.)

Running Progol 4.4 [3] on this example produces the following rules:

```
do(sell, TimeList) :- member(T1, TimeList), indicator_sell1(T1),           (5)
                    member(T2, TimeList), indicator_sell2(T2).
do(buy, TimeList) :- member(T, TimeList), indicator_buy1(T).
do(buy, TimeList) :- member(T, TimeList), indicator_buy2(T).
```

Note that we have learned a (single) rule for SELL involving a combination of two indicators, although each of these indicators, taken separately, covers at least a negative example. Thus, since none of these indicators has enough discrimination power to avoid negative examples, the simpler rule

```
do(sell, TimeList) :- member(T, TimeList), indicator_sell1(T).
```

will not work (similarly for *indicator_sell2*). Also note that the rule (5) holds although the two indicators are never triggered simultaneously - they are just triggered in the same ϵ -band. Insisting that both indicators be triggered simultaneously would produce no results at all.

The main limitations of the ILP approach is the sheer size of the hypotheses space. For each positive example, such as *do(buy, [t₁, t₂, ..., t_k])*, Progol constructs a Most Specific Clause (MSC) which bounds from below the search space of hypotheses. For each time point *t_i* in the example, the MSC will contain a *member/2* literal, as well as an indicator literal for every indicator that is triggered at *t_i*. Since hypotheses roughly correspond to subsets of the MSC, the size of the search space will be exponential in the number of MSC literals.

Running Progol on the USD-DEM historical data⁵ has produced trading rules like the following:

```
do(buy, TimeList) :- member(T, TimeList),
                    mov_avg_xover(T, 10, 15, buy), stochastic_xover(T, 15, buy).
do(buy, TimeList) :- member(T, TimeList), stochastic_xover(T, 5, buy),
                    stochastic_xover(T, 10, buy), stochastic_xover(T, 15, buy).
```

The first rule is particularly interesting since it combines a lagging indicator (moving average crossover) with a stochastic indicator in a single strategy. The second is also interesting since it generates a buy signal only when 3 stochastic crossover systems of increasing lengths agree.

On a different run, Progol 4.2.1 also obtained the following *buy* rules:

```
do(buy, TimeList) :- member(T1, TimeList), member(T2, TimeList),
                    mov_avg_xover(T2, 10, 15, buy), stochastic_xover(T1, 15, buy).
do(buy, TimeList) :- member(T1, TimeList), member(T2, TimeList), member(T3, TimeList),
                    mov_avg_xover(T3, 5, 10, buy), stochastic_xover(T1, 5, buy),
                    stochastic_xover(T2, 25, buy).
```

The first is very similar to the first rule of the previous run. The difference is that it doesn't require the two indicators (moving average crossover and stochastic crossover) to be

⁵ The noise parameter of Progol has been set to 100% since we are dealing with an extremely noisy domain. The number of nodes explored per seed example was limited to 1000.

triggered at exactly the same time point. As argued before, this makes the rule more general and thereby applicable in more situations than the previous one. The second rule activates a buy signal only if a moving average and two stochastic crossover systems all agree - at least in the same ϵ -band.

The profitability of our ILP generated trading rules is about 80% of the profitability of an over-optimized moving average crossover system⁶, which is encouraging for an initial experiment. (The profitability of such over-optimized strategies doesn't usually extrapolate to unseen time series data.) The percentage of actual trades as compared to the total number of ideal trading opportunities was about 25-30%, which is also encouraging considering the chaotic nature of such financial time series.

4 Conclusions

We present an original approach to the problem of inducing symbolic trading rules consisting in first labelling the historical data with ideal trading opportunities, and then using ILP to induce rules based on a given set of technical indicators.⁷

As opposed to other approaches which essentially produce "black-box strategies", ours produces understandable rules, especially since the technical indicators used as "building blocks" have an intuitive reading for the human trader.

Most existing approaches to learning trading strategies are confined to the optimisation of the various numerical parameters of a fixed strategy. Our approach goes beyond these simple approaches by automatically discovering significant combinations of indicators that are capable of recognising ideal trading opportunities. Compared to simple profit-driven rule discovery, our rules have better chances for generalising to unseen data (since the profit of rules evolved on a particular data series can be due to certain accidental features in the data and may not extrapolate to new data). The main utility of our approach is in the case of atypical markets when a trader would like to test his intuition that a certain set of indicators may be helpful, without being able to pinpoint the exact combinations.

References

- [1] S.B. Achelis. *Technical Analysis from A to Z*, Irwin Professional Publishing, 2nd ed., 1995.
- [2] Dietterich T.G., R.H. Lathrop, T. Lozano-Perez, Solving the multiple-instance problem with axis-parallel rectangles, *Artificial Intelligence*, 89 (1-2), pp. 31-71, 1997.
- [3] S. Muggleton. *Inverse Entailment and Progol*, *New Generation Computing Journal*, 13:245-286, 1995.
- [4] S. Muggleton. *Learning from Positive Data*, Proc. of the 6th Int. workshop on Inductive Logic programming ILP-96, Stockholm, 1996.
- [5] Omega Research. *Easy Language User's Manual*, 1996.

⁶ obtained using the Omega TradeStation [5].

⁷ As far as we know, there are no other approaches to learning trading rules using ILP. The main advantage of ILP w.r.t. other methods in this area is the capability of dealing with *disjunctive* examples. I am grateful to the anonymous reviewers who pointed out the connection with multiple-instance learning [2].

Improving Knowledge Discovery Using Domain Knowledge in Unsupervised Learning

Javier Béjar*

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

c/ Jordi Girona 1-3. 08034 Barcelona, Spain.

Phone: 34 3 4015653, Fax: 34 3 4017014

bejar@lsi.upc.es

Abstract. Using domain knowledge in unsupervised learning has shown to be a useful strategy when the set of examples of a given domain has not an evident structure or presents some level of noise. This background knowledge can be expressed as a set of classification rules and introduced as a *semantic bias* during the learning process.

In this work we present some experiments on the use of partial domain knowledge in conceptual clustering. The domain knowledge (or domain theory) is used to select a set of examples that will be used to start the learning process, this knowledge has not to be complete neither consistent. This bias will increase the quality of the final groups and reduce the effect of the order of the examples. Some measures of stability of classification are used as evaluation method.

The improvement of the acquired concepts can be used to improve and correct the domain knowledge. A set of heuristics to revise the original domain theory has been experimented, yielding to some interesting results.

1 Introduction

The use of unsupervised learning to discover useful concepts in sets of non classified examples allow to ease the labour of data analyst in data mining tasks or any other task that involves the discovery of useful descriptions from data. Tools that help to this labour and that increase the quality of the knowledge obtained are very desirable.

In this work we present the methodology used by LINNEO⁺ [1,2,3], that has been extended to use domain knowledge in order to *semantically bias* a conceptual clustering algorithm [7,5]. This knowledge helps to obtain more stable classifications and more meaningful concepts from unclassified observations.

It is shown, also, that little knowledge can produce considerable gain, despite of the ambiguity or the partial incorrectness of the knowledge. This ambiguity can be also solved using the improved classifications, performing specializations or generalizations that correct the domain knowledge.

* This work has been financed by UPC grup precompetitiu PR99-09

2 LINNEO⁺

LINNEO⁺ [1] is a tool oriented to discover probabilistic description of concepts from unclassified data. It uses an unsupervised learning strategy and incrementally discover a classification scheme from the data. The expert has to define dataset of observations to model the domain and also defines a set of attributes relevant to the classification goal intended. The expert is allowed to represent attributes by means of quantitative and qualitative attributes.

The strategy of classification is based on a distance measure and a criterion of membership. The algorithm that it uses is a variation of the nearest neighbour algorithm [6] augmented by the use of probabilistic prototypes in order to describe the discovered clusters [5].

The similarity function used is a generalization of hamming distance usually used by other conceptual clustering algorithms [7]. The aggregation algorithm builds clusters of similar objects given a initial parameter that we call *radius* that selects the level of generality of the induced concepts. This radius is selected heuristically by trial and error. A detailed description of the algorithm can be found in [1,3]

This methodology has been successfully applied to some real domains as mental illnesses [11], marine sponge classification [1] and discovery and characterization of fault diagnose in wastewater treatment plants [12]

3 Using a Domain Theory

In unsupervised learning the description of the observations usually is not enough to build a set of concepts. The noise of the observations, the existence of irrelevant descriptors or the non homogeneity of the sampling of observations can deviate the learning process from a meaningful result. It is desirable, thus, a *guide* from a higher level of knowledge to assure the success of the acquisition.

In our methodology, we allow the expert to define as Domain Theory (DT) as a group of constraints guiding the inductive process. Therefore, the DT semantically *biases* the set of possible classes. This DT acts just as a guide; it does not need to be complete. It could be very interesting for the experts to play with several definitions of DT as they could model several levels of *expertise* or to obtain different classifications using different points of view or bias. The expert is allowed to express his DT in terms of rules that determine the definition of a part of the definition of classes he already knows to exist. A rule is composed by an identifier and some constraints, a set of conditions that elements must fulfill in order to belong to the defined identifier. These conditions are expressed by simple selectors including conditions as =, >, < or membership to a range of values for an specific attribute.

3.1 Biasing with a Domain Theory

If the expert is able to build a DT, it is possible to use this knowledge to *bias* the classification using the constraints as a guide to preprocessing the dataset.

Even in poor defined domains the expert knows that to ignore some attributes for certain classes can be useful, because those attributes are not relevant predicting class membership. In the same way, the expert, knows that there are other attributes, or their conjunction, that could be used, with a certain degree of confidence, to try to predict class membership. The idea is to create a partition of the dataset using the rules defined by the expert in meaningful parts, the objects with some knowledge about its relation (those described by the rules). Those objects that not fulfill none of the rules are treated as without Domain Theory.

The treatment of the dataset previous to the classification is as follows:

- All the objects that satisfies a rule (R_i) are grouped together ($\mathcal{S}_{R_i}$) (the expert could give more than one rule for the same class).
- If the rules are too general, two or more rules could select the same object, in this case a *special* set is created for this objects. The objects are tagged with the conflicting rules.
- All the objects that do not accomplish any rule are grouped in a *residual* set.

After this process at maximum, it is generated $r + 2$ sets of objects, where r is the number of sets that the expert has constrained.

Each one of these sets, except for the *special* and the *residual*, is classified separately and, eventually, it is created at least a class for each. Then a new classification process begins with the centers of these classes as seeds of the new classification and the rest of objects. In this process new classes can be formed corresponding to classes not described by the rules.

The *bias* is obtained by the reordering and previous grouping of the observations in a meaningful scheme, rather than the random order of the unbiased process. This yields a more meaningful set of classes, more in the idea that the expert has of his domain structure. This avoids also the instability induced by the ordering of the observations.

4 Experiments with a Domain Theory

In order to test the effect of a domain theory in the process of classification, we have written a small set of rules for the *Soya bean* domain [3] (11 rules, see table 1 for example rules) to bias the resulting classes. These rules have been built by hand, inspecting the prototypes of the classes of a unbiased classification, extracting the attributes more relevant. This set of rules is neither complete nor consistent, because we just want to show that only a small piece of domain knowledge is enough to improve the stability, and therefore the quality, of a classification. These rules select 130 observations from a total of 307.

The experiment was carried out by comparing two sets of 20 random ordered classification using LINNEO⁺ of the *Soya bean* dataset [8] obtained from the UCI Repository of Machine Learning Databases and Domain Theories [9]. The

Table 1. A Soya Bean Domain Theory

((= (diseased) fruit-pods)	((= (norm) fruit-pods)
(= (colored) fruit-spots)	(= (tan) canker-lesion)
(= (norm) seed)	(= (lt-norm norm) precip)
-> frog-eye-leaf-spot)	-> charcoal-and-brown-stem-rot)
((= (abnorm) seed)	((= (lower-surf) leaf-mild)
(= (tan) canker-lesion)	-> downy-mildew)
-> purple-seed-stain)	

first without use of domain theory, the second using our set of rules as domain theory.

In order to compare the resulting classifications we have developed an algorithm that provides a measure of the differences between two classifications [1,2,4]. This measure, that we call *structural coincidence*, is used to provide a value for the stability of each set of classifications as the mean of the difference of each pair of classifications in the set. Among these differences, it is taken in account the coincidence of objects in the same group and the number of classes of each classification.

Another measure of stability that is used in the comparison is based on the coincidence of the pairs of associations of observations between two partitions described in [4], this measure decreases with the similarity.

The stability of a classification of the Soya Bean dataset without the DT is 77.6% for the first measure and -1013.4 for the second. The stability using the DT increases to 91% for the first measure and -4285.6 for the second. A cross comparison between the two sets of classification yields a value of 79.9% for the structural coincidence. This value has been calculated comparing each class resulting from each method with all the others and averaging. The interpretation of this value is that the classifications using the domain theory are similar to those created without using a *bias* but much more stable. Applying this technique to other datasets yields similar results [3].

In the light of these results, we can say that the use of domain knowledge in unsupervised learning reduces the problem of obtaining meaningless groupings and also reduces the instability induced by an improper input order.

A similar technique for biasing an unsupervised algorithm has been applied in order to build concepts hierarchies successfully [13]. This encourages to apply a similar strategy to bias other incremental conceptual clustering algorithms.

5 Domain Theory Revision

Due to that the domain theory that the expert gives for the *biasing* process could be inconsistent or incomplete, it is worth to improve it in some automatic way. Some EBL systems try to improve incomplete or incorrect domain theories using labeled examples in order to fix the errors [10]. Our system is unsupervised, so we have to trust the classes formed during the classification process and the

source of detected errors only can be from the use of the DT previous to the classification.

We have been experimenting with some heuristics for theory revision. These heuristics are very conservative, they only try to discover the minimum set of changes that improve the selectivity of the rule and maintaining the consistency with the obtained clusters. The heuristics only can revise the clauses applied to one attribute with the operators $=$, $\neq$, $>$, $<$ and **range**.

This revision has two parts. When a dataset is classified using the domain theory two kind of rules may appear if we observe the consistency of the resulting partitions. There is a set of rules that selects a definite set of objects that no other rule selects, we call this set *non collision rules*. There is another set of rules whose sets of objects intersect among them. These are *ambiguous rules* and the multiple selected objects can not be assigned to a definite set. So, the revision can be done separately for each set of rules. First, to improve the non collision rules trying to generalize them or by deleting superfluous conditions. Second, to correct the ambiguous rules, trying to specialize them in order that no object is selected by more than one rule. A more extended description of this process can be found in [1]

5.1 Revision of Non Collision Rules

These rules can be treated separately, because all of them have their own set of examples, classified in one or more groups. The objective of this process is to fit the rules with the groups but not to select objects from other groups.

This improvement has two phases. Firstly, the phase of specializing. Some rules can have an extension so broad, or excessive disjunctive conditions, that can prevent a later generalization. So, some of these conditions can be restricted or dropped in order to be consistent with the values of the objects in the classes selected by the rules. This can be done for example eliminating modalities that do not appear in the values of a class from an equal ($=$) clause, or by changing the $<$ and $>$ clauses to the upper and lower bound of the attribute in the prototype of the class respectively.

The second phase is generalizing. Not all the objects from a class are selected by the rules that had generated it. It is desirable that the rules cover the maximum number of objects of this class in order to be more descriptive of the class. To achieve this we generalize the conditions of the class extending its ranges or dropping conjunctions, only if these changes are consistent with the rest of classes of the dataset. This generalization can be done for example by introducing more modalities in a equal ($=$) condition, modalities that appear in the class and have not been used by the expert or to change the clause to a **range** clause with the bounds of the attribute in the prototype. Also, it is possible to test the effect of eliminate each one of the conditions of the rule.

The corrected rules can help the expert to refine his knowledge.

Table 2. An ambiguous Soya Bean Domain Theory

((= (lt-normal) plant-stand) (= (severe) severity) (= (none) int-discolor) -> phytophthora-and-rhizoctonia-root-rot)	((= (lt-80%) germination) (= (norm) plant-growth) -> bacterial-pustule)
((= (no) lodging) (= (tan) canker-lesion) -> herbicide-injury)	((= (lower-surf) leaf-mild) -> downy-mildew)

5.2 Revision of Ambiguous Rules

This set corresponds to rules too general or to classes where the expert can not differentiate accurately. To treat these rules it is necessary to calculate what groups of rules are in conflict and what objects are the conflictive ones.

As information to correct those rules, it is taken in account the classes that group the conflictive objects and the rule that has formed this group. It is a logical assumption to assign the conflictive objects to the rule that has formed the class the objects belong to.

The objective is to specialize each conflictive rule using as constraints the observations that it has not to select. To do this a rule is specialized constraining its conditions or adding new conditions that exclude the conflicting observations.

The selection of the conditions is done by choosing the attributes from the classes (of the rule) that have values not present in the non desired observations. With these attributes, it is possible to construct new clauses in order to specialize the rule. If the attribute is quantitative, a clause that selects only the values between the range present in the classes can be constructed. If the attribute is qualitative, a clause that test that the modalities are only the present in the classes can be constructed.

The specialization process is done by selecting some of the candidate clauses. Each clause is tested with the clauses from the rule. The clauses selected are those that reduce the most the selection of non desirable observations and maintain the selection of correct observations.

After the specialization process a generalization can be done by testing if some of the original conditions of the rule are unnecessary because the new conditions.

6 Evaluating the Revised Domain Theory

The same dataset has been used in order to evaluate the heuristics, but and artificially ambiguous DT has been constructed [3] (see table 2 for example rules). There are, in this case, 12 rules (some of them grouping more than one category) that select 178 objects (33 in the special class) from a total of 307.

Concretely the rule number 9 has the following collisions: rule 1 (6 observations), rule 2 (2 observations), rule 5 (2 observations), rule 6 (1 observation),

Table 3. The corrected rules

((= (lt-normal) plant-stand) (= (severe) severity) (= (brown dk-brown-blk) canker-lesion) -> phytophthora-and-rhizoctonia-root-rot)	
((= (lt-80%) germination) (= (norm) plant-growth) (= (absent) leaf-mild) (= (absent brown-w/blk-specks) fruit-spots) (= (norm) seed-size) -> bacterial-pustule)	((= (no) lodging) (= (w-s-marg no-w-s-marg) leafspots-marg) (= (90-100%) germination) -> herbicide-injury)

Table 4. Number of objects selected

Rule	Before	After	Rule	Before	After	Rule	Before	After
Num 1	18	24	Num 2	17	17	Num 3	10	17
Num 4	31	31	Num 5	8	8	Num 6	5	6
Num 7	6	6	Num 8	2	2	Num 9	41	41
Num 10	11	14	Num 11	7	10	Num 12	6	10

rule 10 (5 observations), rule 11 (3 observations), rule 12 (4 observations); the rule number 8 has the following collisions: rule 3 (8 observations), rule 10 (2 observations). The total number of conflicting objects is 33. After the correcting process the number of collisions has been reduced to 7 objects, specializing 3 rules as can be seen in table 3. The number of objects selected by each rule after and before the correction can be seen in table 4.

The structural coincidence with the ambiguous rules is 88.9% With the corrected rules it has been increased slightly. The value for the structural coincidence has been increased to 89.6%. It is not expected a great increase of stability because the number of selected objects by the domain theory has not been increased, but the gain of stability is maintained.

Applying this technique to other datasets with expert build domain theories yields similar results, a slightly increase of stability is obtained, but the selectivity of the rules is increased ([1]).

7 Conclusions

It has been shown that the use of domain knowledge as semantic bias in a unsupervised learning algorithm increases the quality of the result. The domain knowledge has not to be perfect, can have some ambiguities or inconsistencies, an increase of stability of the results could still be achieved.

The approximation used to bias learning is enough algorithm independent to be exported to other incremental conceptual clustering algorithms.

The fix and revision of the domain knowledge can also be done, obtaining a benefit from the better classification. Ambiguities can be detected and corrected observing the nature of the obtained groups and generalizing and specializing the knowledge in order to fit the description of the concepts.

References

1. J. Béjar. *Adquisición de conocimiento en dominios poco estructurados*. PhD thesis, Dep. of Software. Universitat Politècnica de Catalunya, 1995. 47, 48, 50, 51, 53
2. J. Béjar, U. Cortés, and R. Sangüesa. *Experiments with Domain Knowledge in Knowledge Discovery* 1st International Conference on the Practical Application of Knowledge Discovery and Data Mining, London, 1997. 47, 50
3. J. Béjar, U. Cortés. *Experiments with Domain Knowledge in Unsupervised Learning: Using and Revising Theories*. Revista Iberoamericana de Computación. Computación y Sistemas 1 (3). pages 136-144. 1998. 47, 48, 49, 50, 52
4. D. Faith, L. Belbin. *Comparison of classifications using measures intermediate between metric dissimilarity and consensus similarity*. *Journal of Classification*, 3:257-280, 1986. 50
5. D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139-172, 1987. 47, 48
6. A. K. Jain, R. C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1989. 48
7. R. Michalski, R. E. Steep. *Learning from observation: Conceptual Clustering, Machine Learning an A.I. Perspective*, pages 331-363. Ed. Tioga, Palo Alto, California, 1983. 47, 48
8. R. S. Michalski, R. L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 1980. 49
9. P. M. Murphy, D. W. Aha. UCI repository of machine learning databases. Irvine, University of California, Department of Information and Computer Science, 1994. 49
10. D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 1993. 50
11. E. Rojo. *Aplicación del software LINNEO a la clasificación de trastornos mentales*. PhD thesis, Facultat de Medicina. Universitat de Barcelona, 1993. 48
12. M. Sánchez, U. Cortés, J. Béjar, J. de Gracia, J. Lafuente, and M. Poch. Concept formation in WWWTP by means of classification techniques: A compared study. *Applied Intelligence* 7, pages 147-165, 1997. 48
13. L. Talavera, J. Bejar. Integrating Declarative Knowledge in Hierarchical Clustering Tasks. Third Symposium on Intelligent Data Analysis. Pages 211-222, Amsterdam, The Netherlands. LNCS vol. 1642. Springer Verlag, 1999. 50

Exploiting Classifier Combination for Early Melanoma Diagnosis Support

E. Blanzieri, C. Eccher, S. Forti, and A. Sboner

ITC-irst

Via Sommarive, 18, 38100 – Trento, Italy
{blanzier,sboner}@itc.it

Abstract. Melanoma is the most dangerous skin cancer and early diagnosis is the main factor for its successful treatment. Experienced dermatologists with specific training make the diagnosis by clinical inspection and they reach 80% level of both sensitivity and specificity. In this paper, we present a multi-classifiers system for supporting the early diagnosis of melanoma. The system acquires a digital image of the skin lesion and extracts a set of geometric and colorimetric features. The diagnosis is performed on the vector of features by integrating with a voting schema the diagnostic outputs of three different classifiers: discriminant analysis, k-nearest neighbor and decision tree. The system is build and validated on a set of 152 skin images acquired via D-ELM. The results are comparable or better of the diagnostic response of a group of expert dermatologists.

1 Introduction

Combination and integration of different models has been a very popular area in recent years and techniques such *bagging* and *boosting* have gained a lot of attention (see for references Chan et al. 1999). In such schemata training of the same algorithm on different data generates different models. However, as noted by Merz (1999), the combination of different kinds of algorithms proved to be effective in increasing accuracy.

In real-world applications, namely when a user is involved in the development (following the definition of Saitta and Neri, 1998), accuracy is not the main issue. Comprehension and readability of the results are particularly relevant when the final user is a skilled physician. Presenting their experience in a medical application, Morik et al. (1999) emphasized the relevance of understandability and embeddedness of the learning component into the overall application system. Finally, in a medical diagnosis application, sensitivity and specificity are far more relevant of accuracy, as noted by Kukar et al. (1999) cost-sensitive algorithms should be used.

In this work we claim that combination of different algorithms can be also useful in order to solve problems that arises in a real application in a sensitive domain such as cancer diagnosis. In particular we present MEDS (Melanoma Diagnosis System) a system for early melanoma diagnosis support developed by ITC-IRST in collaboration with the Department of Dermatology of Santa Chiara Hospital, Trento.

The main goal of MEDS is to provide support to a physician for early diagnosis of melanoma, the most dangerous skin cancer. Although it is diagnosed in about 5% of the overall skin cancers, melanoma is responsible of the 91% of the deaths and its incidence in Europe is increasing of 3%-5% yearly. The early diagnosis of melanoma is the principal factor for the prognosis of this disease. The diagnosis is difficult and requires a well-trained physician, because the early lesion looks like a benign one. Several studies have shown that the diagnostic accuracy of a specialist is about 69% for early melanomas, and it reduces to 12% for non-specialists (Clemente et al. 1998). One of the digital techniques that had considerable success in clinical practice is *digital epi-luminescence microscopy (D-ELM)* (see for a review Zsolt, 1997). It allows the determination of several morphological and structural characteristics of skin lesions without remove them. Several automatic systems were proposed for the early diagnosis of melanoma (Shindewolf 1992, Green 1994, Ercal 1994, Binder 1994, Takiwaki 1995, Seidenari 1998) and recently D-ELM has been exploited by Bischof et al. (1999).

MEDS make the diagnosis on a D-ELM image, so it faces the problem of processing the image for feature extraction. A second problem is that collecting data on melanoma is difficult: melanoma cases are not common and characteristics of the D-ELM images depend on the type of acquisition system chosen. Therefore, the application has to been built with small data sets and unbalanced classes. A third and major problem arises when loss functions are considered. Sensitivity shows the ability of the system to recognize the malign lesion, while specificity describes how the system recognizes the benign lesion. Depending on the application (screening by a general practitioner or diagnosis support of an expert dermatologist) very different levels of sensitivity and specificity are required and the system should provide a tuning mechanism. Finally, another critical issue in order to build a usable system is gaining the trust of the user, comprehensibility of the results are one of the major issue.

MEDS elaborate D-ELM images extracting features that could be meaningful for the expert dermatologist, following the so-called ABCD Rule (Nachbar 1994) in order to improve comprehensibility. The features are the input of three different classifiers, namely Discriminant Analysis, Decision Tree and k-Nearest Neighbor, which MEDS integrates by means of voting schemata. The classifiers permit different explanations of the results. The combination improves the performance in terms of sensitivity and specificity given the small number of data. Finally, the simple voting mechanism is clear and well understood by the user and it is possible to use the voting schema as a tuning parameter comprehensible by the expert.

The paper is organized as follows: section 2 describes the system, technical and clinical validations are presented in section 3 and 4 respectively, section 5 is devoted to related works and finally, conclusions are drawn in section 6.

2 System Description

MEDS architecture has three main components: the D-ELM Image Acquisition component whose goal is to acquire the image of the pigmented skin lesion, the Image Processing component that elaborates the digital image producing the vector of

features, the Multi-classifier that applies and combines Discriminant Analysis, Decision Tree and k-Nearest Neighbor. The system functional architecture is shown in Fig.1. Physically, the three main components run on different machines and we transfer information by means of files. We are working to integrate the components in a client/server application. The image processing and the multi-classifier components will reside on a centralized server, while the D-ELM image acquisition component will be deployed to several clients.

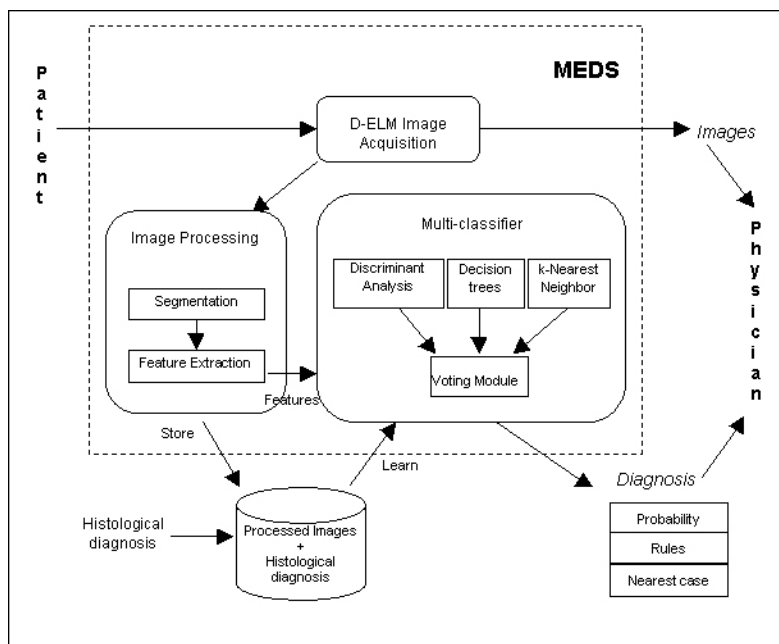


Fig. 1. Overview of the overall system

For the D-ELM Image Acquisition, we used a stereomicroscope Leica WILD M-650, with a color camera SONY 3CCD DXC-930P. The camera is linked with an acquisition board AT-Vista Videographics, which allows digitizing the analog image of the microscope. The software for the image acquisition is DBDERMO MIPS (Dell'Eva/Burroni Studio, Florence/Siena, Italy). For image processing we used the morphometer Leica Q570. The colored image is usually divided in three gray images, which represent the red, the green and the blue component respectively (RGB color space). During the image processing, the image is converted from the usual RGB colors space to the hue, saturation and value (HSV colors space). The HSV colors space is particularly useful because it reflects the human perception of colors.

The Image Processing component performs two functions: segmentation and feature extraction. The purpose of segmentation is defining the border of the lesion, separating it from the rest of the skin. We exploit the HSV images, because, in this case, the normal skin and the lesion present marked differences, especially for the hue and the saturation images. The feature extraction module produces numerical features: geometric and colorimetric ones. The geometric parameters measure the dimension of the lesion (area and perimeter) and its symmetric characteristics (roundness, aspect

ratio and full ratio). The colorimetric features quantitatively reflect concepts as the presence and symmetric or asymmetric distribution of the colors, the granularity of the colors, the irregularity of the pigmentation on the border of the lesion, etc. The extracted features reflect the ABCD rule used by the dermatologist to diagnose a skin lesion (Nachbar 1994). In fact, with this rule, the physician evaluates the Area of the lesion, the irregularity of the Border, the presence and the distribution of the Color and the presence of Differential structures.

The Multi-classifier module produces a diagnosis based on the features extracted from the D-ELM images. It used three kinds of classifiers: discriminant analysis, decision tree and k-nearest neighbor, and combine them by means of different voting schemata. The classifiers were chosen in order to permits different explanation of the results (probability, rules and a nearest similar cases respectively).

In this application, the relevant gain functions for the classifiers are sensitivity, defined as $TP / (TP + FN)$, and specificity, defined as $TN / (TN + FP)$. TP, TN, FP, FN are the number of melanomas correctly classified (True Positives), the number of nevi correctly classified (True Negatives), the number of nevi classified as melanomas (False Positives) and the number of melanomas classified as nevi (False Negatives), respectively. Sensitivity depends on FN and it is the most critical parameter.

In order to improve sensitivity we altered the prior probabilities of the classes as described in Kukar et al. (1999). We adopted different strategies for the three classifiers. The prior probabilities for the linear discriminant analysis were considered equal for each class. Discriminant analysis was performed via a multivariate analysis on features selected by means of a univariate analysis. For the decision tree, we adopted C4.5 and we performed a pre-processing on the data increasing the weight of malignant melanomas as described in Breiman (1984) and reported by Kukar et al. (1999). Finally, we adopted the Euclidean metric for the k-nearest neighbor (*k-NN*). To improve sensitivity, we also used a particular form of the nearest neighbor algorithm (*k-NN-Uni*), whose output is “melanoma” if at least one of the k-nearest-neighbors is a melanoma.

Comprehensibility is preserved by combining discriminant analysis, decision tree and k-nearest-neighbor with simple rules. If the combination involved k-NN-Uni, we adopted the majority rule (schema “2/3”). Otherwise, we required a total agreement on benign lesions for classify the new case as a mole (schema “1/3”).

3 System Validation

We analyzed 152 skin images acquired by D-ELM at the Department of Dermatology of Santa Chiara Hospital, Trento. The images were classified histologically by the pathologist as 42 melanomas and 110 nevi. Breslow thickness was evaluated for 42 malign lesions. This parameter is linked to the prognosis of the disease and it can be determined only by the histological analysis. The average Breslow thickness for our lesions is 1.0 ± 0.7 mm, and the 90% of them are thinner than 1.70 mm. This fact confirms the earliness of the involved melanomas. We evaluate sensitivity and specificity using *10-fold cross-validation* and performed experiments with the single classifiers and their combinations.

Table 1 reports the results for the single classifiers, both sensitivity and specificity with their standard deviation are shown. Discriminant analysis, decision tree and k-NN have poor sensitivity values, while specificity range from 0.83 to 0.97. This fact is due to the major number of benign lesions relative to the melanomas. Instead, for the k-NN-Uni sensitivity is very high (from 0.69 to 1.00), while specificity significantly decreases as k become greater (from 0.75 to 0.20). This fact agrees with the adopted modified decision rule for the k-NN-Uni, which strongly promotes sensitivity.

Table 2 reports the results for the 3-Classifiers systems. The combination of three classifiers has a general improvement of sensitivity without a strong decrease of specificity. Comparison of the 3-classifiers with the single systems (in particular for the voting schema “1/3”) shows a statistical improvement of sensitivity, while specificity shows no statistical differences.

Table 1. Single classifier systems – For each classifier, sensitivity and specificity, with the standard deviation, are shown

CLASSIFIER	Sens.	SD	Spec.	SD	CLASSIFIER	Sens.	SD	Spec.	SD
<i>Discriminant analysis</i>					<i>k-NN-Uni</i>				
DiscrAn	0.65	0.30	0.83	0.11	2-Uni	0.69	0.30	0.75	0.10
<i>Decision trees</i>					3-Uni	0.81	0.19	0.61	0.08
C4.5	0.64	0.28	0.84	0.05	4-Uni	0.86	0.19	0.53	0.16
<i>k-Nearest-Neighbor</i>					5-Uni	0.88	0.16	0.45	0.17
1-NN	0.68	0.30	0.90	0.10	6-Uni	0.99	0.05	0.35	0.16
3-NN	0.49	0.36	0.91	0.07	7-Uni	1.00	0.00	0.29	0.18
5-NN	0.46	0.34	0.97	0.06	8-Uni	1.00	0.00	0.28	0.18
7-NN	0.35	0.23	0.97	0.04	9-Uni	1.00	0.00	0.20	0.18
9-NN	0.41	0.25	0.96	0.04					

Table 2. 3-Classifiers systems – The columns “Comparison [vs. single] {+, =, -}” represent the comparison of the combined classifier with each single component: combined classifier better than the single (+), combined worst than single (-) and no significant differences (=). The statistical analysis is based on the paired Wilcoxon test with a p-value of 0.05

				Sensitivity						Specificity					
						Comparison						Comparison			
						[vs. single]						[vs. single]			
Voting Schema	Combined Classifiers			Value	SD	+	=	-	Value	SD	+	=	-		
1/3	DiscrAn	C4.5	1-NN	0.86	0.32	ααα			0.64	0.11	ααα				
	DiscrAn	C4.5	3-NN	0.84	0.32	ααα			0.65	0.11	ααα				
	DiscrAn	C4.5	5-NN	0.84	0.32	ααα			0.68	0.12	ααα				
	DiscrAn	C4.5	7-NN	0.84	0.32	ααα			0.68	0.10	ααα				
	DiscrAn	C4.5	9-NN	0.85	0.32	ααα			0.68	0.11	ααα				
2/3	DiscrAn	C4.5	2-Uni	0.75	0.31	ααα			0.89	0.11	ααα				
	DiscrAn	C4.5	3-Uni	0.75	0.31	ααα			0.84	0.09	ααα				
	DiscrAn	C4.5	4-Uni	0.77	0.31	ααα			0.81	0.11	ααα				
	DiscrAn	C4.5	5-Uni	0.77	0.31	ααα			0.81	0.11	α	αα			
	DiscrAn	C4.5	6-Uni	0.82	0.31	αα	α		0.78	0.10	α	αα			
	DiscrAn	C4.5	7-Uni	0.84	0.32	αα	α		0.75	0.09	α	αα			
	DiscrAn	C4.5	8-Uni	0.84	0.32	αα	α		0.75	0.09	α	αα			
	DiscrAn	C4.5	9-Uni	0.84	0.32	αα	α		0.71	0.12	α	αα			

4 Clinical Validation

We involved in this study a group of eight dermatologists in order to compare the performances of the system with those of clinicians. Part of the group of the dermatologist was experienced in digital epiluminescence. The average sensitivity and specificity of the dermatologist were 0.83 and 0.66 respectively and the diagnosis were performed only on a video device, reproducing a teledermatology setting.

Table 3 and Table 4 show the results obtained comparing the dermatologists to the classifiers: for single classifiers and 3-Classifiers systems respectively. Each table shows the number of physicians (among the eight dermatologists involved in this study) that performed better, equal or worse than the classifier. We used the paired Wilcoxon test to measure statistical significance. The p-value considered for discriminate the results was 0.05. Table 3 shows that the single classifier systems do not reach useful performances for the early diagnosis of melanoma. When they perform better for one parameter, for example sensitivity, the physicians perform better for the other. Table 4 shows that the 3-Classifiers systems perform as well as the eight dermatologists for what concern sensitivity, while they perform better for what concern specificity for, at least, one physician. Moreover, these systems never have poor performances compared with each dermatologist for both sensitivity and specificity, and the mis-classified melanomas are different from those of the physicians. This fact confirms the possibility to use these systems as a diagnosis support, also for the well-trained dermatologist.

Table 3. Dermatologists vs. single classifier systems. (+ means that the classifier is better than the physician, = means that there is non statistical difference between the classifiers and the physician and – means that the physician is better than the classifier.)

<i>Dermatologists</i> <i>vs.</i>			<i>Sens.</i>			<i>Spec.</i>			<i>Dermatologists</i> <i>vs.</i>			<i>Sens.</i>			<i>Spec.</i>		
			+	=	-	+	=	-				+	=	-	+	=	-
DiscrAn			0	4	4	5	3	0	2-Uni			0	8	0	3	5	0
C4.5			0	7	1	6	2	0	3-Uni			0	8	0	2	4	2
1-NN			0	8	0	7	1	0	4-Uni			0	8	0	0	4	4
3-NN			0	3	5	7	1	0	5-Uni			0	8	0	0	3	5
5-NN			0	2	6	8	0	0	6-Uni			3	5	0	0	1	7
7-NN			0	1	7	8	0	0	7-Uni			3	5	0	0	1	7
9-NN			0	1	7	8	0	0	8-Uni			3	5	0	0	1	7
									9-Uni			3	5	0	0	0	8

Table 4. Dermatologists vs. 3-classifiers systems

<i>Dermatologists vs.</i> <i>Cl. 1 Cl. 2 Cl. 3</i>			<i>Sens.</i>			<i>Spec.</i>			<i>Dermatologists vs.</i> <i>Cl. 1 Cl. 2 Cl. 3</i>			<i>Sens.</i>			<i>Spec.</i>		
			+	=	-	+	=	-				+	=	-	+	=	-
DiscrAn	C4.5	1-NN	0	8	0	2	6	0	DiscrAn	C4.5	2-Uni	0	8	0	1	7	0
DiscrAn	C4.5	3-NN	0	8	0	2	6	0	DiscrAn	C4.5	3-Uni	0	8	0	1	7	0
DiscrAn	C4.5	5-NN	0	8	0	2	6	0	DiscrAn	C4.5	4-Uni	0	8	0	2	6	0
DiscrAn	C4.5	7-NN	0	8	0	2	6	0	DiscrAn	C4.5	5-Uni	0	8	0	2	6	0
DiscrAn	C4.5	9-NN	0	8	0	2	6	0	DiscrAn	C4.5	6-Uni	0	8	0	2	6	0
									DiscrAn	C4.5	7-Uni	0	8	0	2	6	0
									DiscrAn	C4.5	8-Uni	0	8	0	2	6	0
									DiscrAn	C4.5	9-Uni	0	8	0	2	6	0

5 Related Works

In the recent years, several systems for the diagnosis of melanoma were proposed. Shindewolf et al. (1992) used a decision tree to classify digital images of skin lesions. The images were acquired by a photo-camera, and then scanned by a color TV camera and digitized. They showed results based on a resubstitution evaluation technique. Green et al. (1994) used a discriminant analysis as classification system. In this case, the obtained results seem to refer to all the cases, without any evaluation of the prediction performance. Ercal et al. (1994) applied an artificial neural network on feature extracted by photographic images with different films. As the color is the most significant parameter for the diagnosis of early melanoma it is difficult to compare images from different films. They obtained a sensitivity of 0.86 and a specificity of 0.85 as best result. Binder et al. (1994) applied an artificial neural network to classify dermatological images. They used as inputs of the neural network the ABCDE parameters that were predefined by a physician. This is a semi-automated method, which strongly relies upon the dermatologist. Takiwaki et al. (1995) used a decision tree to discriminate among the lesions. The reported results show only the generated tree, describing the most significant features.

Some recent works are more related to MEDS. Seidenari et al. (1998) applied a discriminant analysis describing the most significant features but it is not clear the evaluation procedure they adopted. Finally, Bischof et al. (1999) used a decision tree to classify images from a D-ELM system. Their results show a cross-validated sensitivity of 0.89 and a specificity of 0.80. Using their methodology, namely training only a decision tree, we did not succeed in reaching such good results (see in Table 1), probably for the different characteristics of the data.

6 Conclusions

In this paper, we have presented MEDS, a system for early diagnosis support of melanoma. MEDS uses a combination of classifiers for solving some of the typical problems that are present in melanoma diagnosis applications. By combination of standard learning algorithms it is possible to improve sensitivity and specificity for reaching the performance of skilled dermatologists solving the problems related to small data sets and unbalanced classes. Different combinations can be useful in order to select the level of sensitivity or specificity required by applications like screening or support of expert dermatologists. The algorithms selected are able to suggest an explanation for the diagnosis: a probability in case of discriminant analysis, a rule in case of decision tree and a similar case using the k-nearest neighbor. Comprehensibility is improved by extracting features related to the clinical practice, and in particular to the digital epiluminescence methodology.

MEDS is integrated with D-ELM that represents the state of the art of clinical practice in pigmented skin lesion diagnosis, and we plan to test the system in a clinical setting.

Acknowledgments

We thank M. Cristofolini and P. Bauer of the Department of Dermatology of the S.Chiara Hospital – Trento, for their significant collaboration in this study, and F. Ricci for participating to the initial phase of the project. We thank the reviewers for insightful comments, which helped to improve the paper. This work was partially supported by the Lega Italiana per la lotta contro i tumori – Sezione Trentina

References

1. Binder M., Steiner A., Scharz M., Knollmayer S., Wolff K., Pehamberger H., Application of an artificial neural network in epiluminescence microscopy pattern analysis of pigmented skin lesions: a pilot study, *British Journal of Dermatology* (1994) 130, 460-465.
2. Bischof L., Talbot H., Breen E., Lovell D., Chan D., Stone G., Menzies S.W., Gutenev A., Caffin R., An Automated Melanoma Diagnosis System, in Pham B., Braun M., Maeder A. J. and Eckert M.P., editors, *New Approaches in Medical Image Analysis*, volume 3474, pp. 130-41, SPIE (1999).
3. Breiman L., Friedman J.H., Ohlsen R.A., Stone C.J., *Classification and Regression Trees*, Monterey, CA, Wadsworth (1984).
4. Chan P.K, Stolfo S.J., Wolpert D., (Eds) *Machine Learning, Special Issue on Integrating Multiple Models for Improving and Scaling Machine Learning Algorithms*. 36 (1999).
5. Clemente C.G., Mihm M.C., Cainelli T., Cristofolini M., Cascinelli N., *Melanoma e Nevi*, Effetti, Milano (1998).
6. Ercal F., Chawla A., Stoecker W.V., Lee H., Moss R.H., Neural network diagnosis of malignant melanoma from color images, *IEEE Transaction on Biomedical Engineering*, vol. 4, n. 9, Sept. 1994.
7. Green A., Martin N., Pfitzner J., O'Rourke M., Knight N., Computer image analysis in the diagnosis of melanoma, *J Am Acad Dermatol* (1994), 31:958-64.
8. Kukar M., Kononenko I., Groselj C., Krali K., Fetic J., Analysing and improving the diagnosis of ischaemic heart disease with machine learning, *Artif Intell Med*, vol. 16 (1999) 25-50.
9. Merz C., Using Correspondence Analysis to Combine Classifiers, *Machine Learning* 36, 33-58, (1999).
10. Morik K., Brockhausen P., Joachims T., Combining statistical learning with a knowledge-based approach – A case study in intensive case monitoring, *ICML99*, 268-277, (1999).
11. Nachbar F., Stolz W., Merkle T., Cognetta A.B., Vogt T., Landthaler M., Bilek P., Braun-Falco O., Plewig G., *The ABCD rule of Dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions*, *J Am Acad Dermatol* (1994) 30:551-9.
12. Saitta L., Neri F., Learning in the "Real-World", *Machine Learning* 30, 133-136 (1998).
13. Seidenari S., Pellacani G., Pepe P., Digital videomicroscopy improves diagnostic accuracy for melanoma, *J Am Acad Dermatol* (1998) 39:175-81.
14. Shindewolf T., Stolz W., Albert R., Abmayr W., Harms H., Classification of Melanocytic Lesions with Colour and Texture Analyses Using Digital Image Processing, *The international Academy of Cytology*, (1992).
15. Takiwaki H., Shirai S., Watanabe Y., Nakagawa K., Arase S., A rudimentary system for automatic discrimination among basic skin lesions on the basis of color analysis of video images, *J Am Acad Dermatol* (1995) 32:600-3.
16. Zsolt B.A. Dermoscopy (Epiluminescence Microscopy) of pigmented skin lesions, *Dermatologic clinics*, Vol. 15, n.1 Jan 1997.

A Comparison of Ranking Methods for Classification Algorithm Selection

Pavel B. Brazdil and Carlos Soares

LIACC/Faculty of Economics, University of Porto
R. Campo Alegre, 823, 4150-800 Porto, Portugal
{pbrazdil,csoares}@ncc.up.pt

Abstract. We investigate the problem of using past performance information to select an algorithm for a given classification problem. We present three ranking methods for that purpose: average ranks, success rate ratios and significant wins. We also analyze the problem of evaluating and comparing these methods. The evaluation technique used is based on a leave-one-out procedure. On each iteration, the method generates a ranking using the results obtained by the algorithms on the training datasets. This ranking is then evaluated by calculating its distance from the ideal ranking built using the performance information on the test dataset. The distance measure adopted here, average correlation, is based on Spearman's rank correlation coefficient. To compare ranking methods, a combination of Friedman's test and Dunn's multiple comparison procedure is adopted. When applied to the methods presented here, these tests indicate that the success rate ratios and average ranks methods perform better than significant wins.

Keywords: classifier selection, ranking, ranking evaluation

1 Introduction

The selection of the most adequate algorithm for a new problem is a difficult task. This is an important issue, because many different classification algorithms are available. These algorithms originate from different areas like Statistics, Machine Learning and Neural Networks and their performance may vary considerably [12]. Recent interest in combination of methods like bagging, boosting, stacking and cascading has resulted in many new additional methods. We could reduce the problem of algorithm selection to the problem of algorithm performance comparison by trying all the algorithms on the problem at hand. In practice this is not feasible in many situations, because there are too many algorithms to try out, some of which may be quite slow., especially with large amounts of data, as it is common in Data Mining. An alternative solution would be to try to identify the single best algorithm, which could be used in all situations. However, the *No Free Lunch* (NFL) theorem [19] states that if algorithm A outperforms algorithm B on some cost functions, then there must exist exactly as many other functions where B outperforms A.

All this implies that, according to the problem at hand, specific recommendation should be given concerning which algorithm(s) should be used or tried out. Brachman et al. [3] describe algorithm selection as an exploratory process, highly dependent on the analyst's knowledge of the algorithms and of the problem domain, thus something which lies somewhere on the border between engineering and art.

As it is usually difficult to identify a single best algorithm reliably, we believe that a good alternative is to provide a ranking. In this paper we are concerned with ranking methods. These methods use experimental results obtained by a set of algorithms on a set of datasets to generate an ordering of those algorithms. The ranking generated can be used to select one or more suitable algorithms for a new, previously unseen problem. In such a situation, only the top algorithm, i.e. the algorithm expected to achieve the best performance, may be tried out or, depending on the available resources, the tests may be extended to the first few algorithms in the ranking.

Considering the NFL theorem we cannot expect that a single best ranking of algorithms could be found and be valid for all datasets. We address this issue by dividing the process into two distinct phases. In the first one, we identify a subset of relevant datasets that should be taken into account later. In the second phase, we proceed to construct a ranking on the basis of the datasets identified. In this paper we restrict our attention to the second phase only. Whatever method we use to identify the relevant datasets, we still need to resolve the issue concerning which ranking method is the best one.

Our aim is to examine three ranking methods and evaluate their ability to generate rankings which are consistent with the actual performance information of the algorithms on an unseen dataset. We also investigate the issue whether there are significant differences between them, and, if there are, which method is preferable to the others.

2 Ranking Methods

The ranking methods presented here are: *average ranks* (AR), *success rate ratios* (SRR) and *significant wins* (SW). The first method, AR, uses, as the name suggests, individual rankings to derive an overall ranking. The next method, SRR, ranks algorithms according to the relative advantage/disadvantage they have over the other algorithms. A parallel can be established between the ratios underlying SRR and performance scatter plots that have been used in some empirical studies to compare pairs of algorithms [14]. Finally, SW is based on pairwise comparisons of the algorithms using statistical tests. This kind of tests is often used in comparative studies of classification algorithms.

Before presenting the ranking methods, we describe the experimental setting. We have used three decision tree classifiers, C5.0, C5.0 with boosting [15] and Ltree, which is a decision tree which can introduce oblique decision surfaces [9]. We have also used an instance based classifier, TiMBL [6], a linear discriminant and a naive bayes classifier [12]. We will refer to these algo-

gorithms as `c5`, `c5boost`, `ltree`, `timbl`, `discrim` and `nbayes`, respectively. We ran these algorithms on 16 datasets. Seven of those (`australian`, `diabetes`, `german`, `heart`, `letter`, `segment` and `vehicle`) are from the StatLog repository¹ and the rest (`balance-scale`, `breast-cancer-wisconsin`, `glass`, `hepatitis`, `house-votes-84`, `ionosphere`, `iris`, `waveform` and `wine`) are from the UCI repository² [2]. The error rate was estimated using 10-fold cross-validation.

2.1 Average Ranks Ranking Method

This is a simple ranking method, inspired by Friedman’s M statistic [13]. For each dataset we order the algorithms according to the measured error rates³ and assign ranks accordingly. The best algorithm will be assigned rank 1, the runner-up, 2, and so on. Let r_j^i be the rank of algorithm j on dataset i . We calculate the *average rank* for each algorithm $\bar{r}_j = (\sum_i r_j^i) / n$, where n is the number of datasets. The final ranking is obtained by ordering the average ranks and assigning ranks to the algorithms accordingly. The average ranks based on all the datasets considered in this study and the corresponding ranking are presented in Table 1.

Table 1. Rankings generated by the three methods on the basis of their accuracy on all datasets

Algorithm (j)	AR		SRR		SW	
	$\bar{r}_j$	Rank	SRR_j	Rank	pw_j	Rank
c5	3.9	4	1.017	4	0.225	4
ltree	2.2	1	1.068	2	0.425	2
timbl	5.4	6	0.899	6	0.063	6
discrim	2.9	3	1.039	3	0.388	3
nbayes	4.1	5	0.969	5	0.188	5
c5boost	2.6	2	1.073	1	0.438	1

2.2 Success Rate Ratios Ranking Method

As the name suggests this method employs ratios of success rates between pairs of algorithms. We start by creating a *success rate ratio table* for each of the datasets. Each slot of this table is filled with $SRR_{j,k}^i = (1 - ER_j^i) / (1 - ER_k^i)$, where ER_j^i is the measured error rate of algorithm j on dataset i . For example,

¹ See <http://www.liacc.up.pt/ML/statlog/>.

² Some preparation was necessary in some cases, so some of the datasets may not be exactly the same as the ones used in other experimental work.

³ The measured error rate refers to the average of the error rates on all the folds of the cross-validation procedure.

on the **australian** dataset, the error rates of **timbl** and **discrim** are 19.13% and 14.06%, respectively, so $SRR_{\text{timbl}, \text{discrim}}^{\text{australian}} = (1 - 0.1913)/(1 - 0.1406) = 0.941$,

indicating that **discrim** has advantage over **timbl** on this dataset. Next, we calculate a *pairwise mean success rate ratio*, $SRR_{j,k} = (\sum_i SRR_{j,k}^i)/n$, for each pair of algorithms j and k , where n is the number of datasets. This is an estimate of the general advantage/disadvantage of algorithm j over algorithm k . Finally, we derive the *overall mean success rate ratio* for each algorithm, $SRR_j = (\sum_k SRR_{j,k})/(m - 1)$ where m is the number of algorithms (Table 1). The ranking is derived directly from this measure. In the current setting, the ranking obtained is quite similar to the one generated with AR, except for **c5boost** and **1tree**, which have swapped positions.

2.3 Significant Wins Ranking Method

This method builds a ranking on the basis of results of pairwise hypothesis tests concerning the performance of pairs of algorithms. We start by testing the significance of the differences in performance between each pair of algorithms. This is done for all datasets. In this study we have used paired t tests with a significance level of 5%. We have opted for this significance level because we wanted the test to be relatively sensitive to differences but, at the same time, as reliable as possible. A little less than 2/3 (138/240) of the hypothesis tests carried out detected a significant difference. We denote the fact that algorithm j is significantly better than algorithm k on dataset i as $ER_j^i \ll ER_k^i$. Then, we construct a *win table* for each of the datasets as follows. The value of each cell, $W_{j,k}^i$, indicates whether algorithm j wins over algorithm k on dataset i at a given significance level and is determined in the following way:

$$W_{j,k}^i = \begin{cases} 1 & \text{iff } ER_j^i \ll ER_k^i \\ -1 & \text{iff } ER_k^i \ll ER_j^i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Note that $W_{j,k}^i = -W_{k,j}^i$ by definition. Next, we calculate the *pairwise estimate of the probability of winning* for each pair of algorithms, $pw_{j,k}$. This is calculated by dividing the number of datasets where algorithm j is significantly better than algorithm k by the number of datasets, n . This value estimates the probability that algorithm j is significantly better than algorithm k . For instance, **1tree** is significantly better than **c5** on 5 out of the 16 datasets used in this study, thus $pw_{\text{1tree}, \text{c5}} = 5/16 = 0.313$. Finally, we calculate the *overall estimate of the probability of winning* for each algorithm, $pw_j = (\sum_k pw_{j,k})/(m - 1)$ where m is the number of algorithms (Table 1). The values obtained are used as a basis for constructing the overall ranking. In our example, $pw_{\text{c5boost}} = 0.438$, which is the largest one and, thus, **c5boost** appears first in the ranking, closely followed by **1tree**, as happened in the ranking generated with SRR.

3 Evaluation

Having considered three ranking methods, we would like to know whether their performances differ, and, if they do, which is the best one. For that purpose we use a leave-one-out procedure. For each dataset (*test dataset*), we do the following:

1. Build a *recommended ranking* by applying the ranking method under evaluation to all but the test dataset (*training datasets*).
2. Build an *ideal ranking* for the test dataset.
3. Calculate the distance between the two rankings using an appropriate measure.

The score of each of the ranking methods is expressed in terms of the mean distance.

The ideal ranking represents the correct ordering of the algorithms on a test dataset, and it is constructed on the basis of their performance (measured error rate) on that dataset. Therefore, the distance between the recommended ranking and the ideal ranking for some dataset is a measure of the quality of the former and thus also of the ranking method that generated it.

Creating an ideal ranking is not a simple task, however. Given that only a sample of the population is known, rather than the whole population, we can only *estimate* the error rate of algorithms. These estimates have confidence intervals which may overlap. Therefore, the ideal ranking obtained simply by ordering the estimates may often be quite meaningless. For instance, Table 2 shows one ranking for the **glass** dataset, where **c5** and **1tree** are ranked in 2nd and 3rd, respectively. The performance of these algorithms on this dataset is, however, not significantly different, according to a paired *t* test at a 5% significance level. Thus, we would not consider a ranking where the position of **c5** and **1tree** is interchanged worse than the one we show. In such a situation, these algorithms often swap positions in different folds of the *N*-fold cross-validation procedure (Table 2). Therefore, we use *N* orderings to represent an ideal ordering.

To calculate the distance between the recommended ranking and each of the *N* orderings that represent the ideal ranking, we use Spearman's rank correlation coefficient [13]. The score of the recommended ranking is expressed as the average of the *N* correlation coefficients. This measure is referred here as *average correlation*, $\bar{C}$.

To illustrate this performance measure, we evaluate the ranking recommended by SW for the **glass** dataset, focusing on the first fold (Table 2). Note that **c5** and **c5boost** share the first place in the ordering obtained in this fold, so they are both assigned rank $\frac{1+2}{2} = 1.5$, following the method in [13]. A similar situation occurs with **c5** and **nbayes** in the recommended ranking⁴. To calculate Spearman's rank correlation coefficient we first calculate $D^2 = \sum D_i^2$, where D_i is the difference between the recommended and the ideal rank for algorithm *i*. The correlation coefficient is $r_s = 1 - \frac{6D^2}{n^3 - n}$, where *n* is the number of datasets.

⁴ The same reasoning is applied when more than two algorithms are tied.

In our example, $D^2 = 17.5$ and $r_s = 0.5$, where n is the number of algorithms. These calculations are repeated for all the folds, permitting to calculate the score of the recommended ranking, $\bar{C}$, as the average of the individual coefficients.

Table 2. Some steps in the calculation of the correlation coefficient between recommended and ideal ranking for the `glass` dataset

		Average		Fold 1			Fold 5		
Algorithm (<i>i</i>)	Rec. rank	ER (%)	rank	ER (%)	rank	D_i^2	ER (%)	rank	D_i^2
c5	4.5	29.9	2	28.6	1.5	9	47.6	3.5	1
ltree	1	31.8	3	31.8	3	4	42.9	2	1
timbl	6	45.2	5	50.0	5	1	52.4	5	1
discrim	3	36.9	4	36.4	4	1	47.6	3.5	0.25
nbayes	4.5	48.7	6	59.1	6	2.25	71.4	6	2.25
c5boost	2	23.8	1	28.6	1.5	0.25	23.8	1	1

Table 3 presents the results of the evaluation of the three ranking methods presented earlier. These results indicate that AR is the best method as the mean $\bar{C}$ has the highest value (0.426). It is followed by SRR (0.411) and SW (0.387). However, when looking at the standard deviations, the differences do not seem to be too significant. A comparison using an appropriate statistical test needs to be carried out. It is described in the next section.

4 Comparison

To test whether the ranking methods have significantly different performance we have used a distribution-free hypothesis test on the difference between more than two population means, Friedman’s test [13]. This hypothesis test was used because we have no information about the distribution of the correlation coefficient in the population of datasets, the number of samples is larger than 2 and also because the samples are related, i.e. for each ranking method the correlation coefficients are calculated for the same part of each dataset. According to Neave and Worthington [13] not many methods can compete with Friedman’s test with regard to both power and ease of computation.

Here, the hypotheses are:

H_0 : There is no difference in the mean average correlation coefficients for the three ranking methods.

H_1 : There are some differences in the mean average correlation coefficients for the three ranking methods.

We will use results for fold 1 on datasets `australian` and `ionosphere` to illustrate how this test is applied (Table 4). First, we rank the correlation coefficients of all the ranking methods for each fold on each dataset. We thus obtain $R_j^{d,f}$,

Table 3. Average correlation scores for the three ranking methods

Test dataset	AR	SRR	SW
australian	0.417	0.503	0.494
balance-scale	0.514	0.440	0.651
breast-cancer-wisconsin	0.146	0.123	0.123
diabetes	0.330	0.421	0.421
german	0.460	0.403	0.403
glass	0.573	0.573	0.413
heart	0.324	0.339	0.339
hepatitis	0.051	0.049	0.049
house-votes-84	0.339	0.307	0.307
ionosphere	0.326	0.326	0.120
iris	0.270	0.167	0.167
letter	0.086	0.086	-0.086
segment	0.804	0.853	0.804
vehicle	0.800	0.731	0.731
waveform	0.714	0.663	0.663
wine	0.621	0.587	0.587
Mean C	0.426	0.411	0.387
StdDv	0.235	0.235	0.262

representing the rank of the correlation obtained by ranking method j on fold f of dataset d , when compared to the corresponding correlations obtained by the other methods. Next, we calculate the mean rank for each method, $\bar{R}_j$, and the overall mean rank across all methods, $\bar{R}$. As each method is ranked from 1 to k , where k is the number of methods being compared (3 in the present case), we know that $\bar{R} = \frac{k+1}{2} = 2$. Then we calculate the sum of the squared differences between the mean rank for each method and the overall mean rank, $S = \sum_{j=1}^k (\bar{R}_j - \bar{R})^2$. Finally, we calculate Friedman’s statistic, $M = \frac{12nS}{k(k+1)}$, where n is the number of points being compared, which in this case is the total number of folds. In this simple example where $n = 2$, $S = 0.5$ and $M = 1$. The critical region for this test has the form $M \geq \text{critical value}$, where the critical value is obtained from the appropriate table, given the number of methods (k) and the number of points (n).

Table 4. Some steps in the application of Friedman’s test and Dunn’s Multiple Comparison procedure on folds 1 of the `australian` and `ionosphere` datasets

	australian		ionosphere				
Method (j)	r_s	$R_j^{\text{australian},1}$	r_s	$R_j^{\text{ionosphere},1}$	$\bar{R}_j$	$(\bar{R}_j - \bar{R})^2$	$\sum_{d,f} R_j^{d,f}$
SW	0.357	2	-0.371	3	2.5	0.25	5
SRR	0.314	3	-0.086	1	2	0	4
AR	0.371	1	-0.214	2	1.5	0.25	3

Dealing with Ties. When applying this test ties may occur, meaning that two ranking methods have the same correlation coefficient on a given fold of a given dataset. In that case, the average rank value is assigned to all the methods involved, as explained earlier for Spearman’s correlation coefficient. When the number of ties is significant, the M statistic must be corrected [13]. First, we calculate Friedman’s statistic as before, M . Then, for each fold of each dataset, we calculate $t^* = t^3 - t$, where t is the number of methods contributing to a tie. Next, we obtain T by adding up all t^* s. The correction factor is $C = 1 - \frac{T}{n(k^3 - k)}$ where k and n are the number of methods and the number of points, as before. The modified statistic is $M^* = M/C$. The critical values for M^* are the same as for M . More details can be found in [13,16].

Results. With the full set of results available, $\bar{R}_{AR} = 1.950$, $\bar{R}_{SRR} = 1.872$ and $\bar{R}_{SW} = 2.178$. Given that the number of ties is high (55%), the statistic is appropriately corrected, yielding $M^* = 13.39$. The critical value for the number of methods being compared ($k = 3$) and the number of points in each ($n = \#datasets * \#folds = 160$) is 9.210 for a significance level of 1%⁵. As $M^* > 9.210$, we are 99% confident that there are some differences in the $\bar{C}$ scores for the three ranking methods, contrary to what could be expected.

Which Method is Better? Naturally, we must now determine which methods are different from one another. To answer this question we use Dunn’s multiple comparison technique [13]. Using this method we test $p = \frac{1}{2}k(k-1)$ hypotheses of the form:

$H_o^{(i,j)}$: There is no difference in the mean average correlation coefficients between methods i and j .

$H_1^{(i,j)}$: There is some difference in the mean average correlation coefficients between methods i and j .

We use again the results for fold 1 on datasets **australian** and **ionosphere** to illustrate how this procedure is applied (Table 4). First, we calculate the rank sums for each method. Then we calculate $T_{i,j} = D_{i,j}/stdev$ for each pair of ranking methods, where $D_{i,j}$ is the difference in the rank sums of methods i and j , and, $stdev = \sqrt{\frac{nk(k+1)}{6}}$. As before, k is the number of methods and n is the number of points in each. In our simple example, where $n = 2$ and $k = 3$, $stdev = 2$, $D_{SRR,AR} = D_{SW,SRR} = 1$ and $D_{SW,AR} = 2$, and then $|T_{SW,SRR}| = |T_{SRR,AR}| = 0.5$ and $|T_{SW,AR}| = 1$.

The values of $|T_{i,j}|$, which follow a normal distribution, are used to reject or accept the corresponding null hypothesis at an appropriate confidence level. As we are doing multiple comparisons, we have to carry out the Bonferroni adjustment to the chosen overall significance level. Neave and Worthington [13] suggest a rather high overall significance level (between 10% and 25%) so that we

⁵ We have used the critical value for $n = \infty$, which does not affect the result of the test.

could detect any differences at all. The use of high significance levels naturally carries the risk of obtaining false significant differences. However, the risk is somewhat reduced thanks to the previous application of the Friedman’s test, which concluded that there exist differences in the methods compared. Here we use an overall significance level of 25%. Applying the Bonferroni adjustment, we obtain $\alpha = \text{overall } \alpha / k(k - 1) = 4.17\%$ where $k = 3$, as before. Consulting the appropriate table we obtain the corresponding critical value, $z = 1.731$. If $|T_{i,j}| \geq z$ then the methods i and j are significantly different.

Given that three methods are being compared, the number of hypotheses being tested is, $p = 3$. We obtain $|T_{\text{SRR},\text{SW}}| = 1.76$, $|T_{\text{AR},\text{SW}}| = 3.19$ and $|T_{\text{SRR},\text{AR}}| = 1.42$. As $|T_{\text{SRR},\text{SW}}| > 1.731$ and $|T_{\text{AR},\text{SW}}| > 1.731$, we conclude that both SRR and AR are significantly better than SW.

5 Discussion

Considering the variance of the obtained $\bar{C}$ scores, the conclusion that the SRR and AR are both significantly better than SW is somewhat surprising.

We have observed that the three methods generated quite similar rankings with the performance information on all the datasets used (Table 1). However, if we compare the rankings generated using the leave-one-out procedure, we observe that the number of differently assigned ranks is not negligible. In a total of 96 assigned ranks, there are 33 differences between AR and SRR, 8 between SRR and SW, and 27 between SW and AR.

Next, we analyze the ranking methods according to how well they exploit the available information and present some considerations concerning sensitivity and robustness.

Exploitation of Information. The aggregation methods underlying both SRR and AR exploit to some degree the magnitude of the difference in performance of the algorithms. The ratios used by the method SRR indicate not only which algorithm performs better, but also exploit the magnitude of the difference. To a smaller extent, the difference in ranks used in the AR method, does the same thing. However, in SW, the method is restricted to whether the algorithms have different performance or not, therefore exploiting no information about the magnitude of the difference. Therefore, it seems that methods that exploit more information generate better rankings.

Sensitivity to the Significance of Differences. One potential drawback of the AR method is that it is based on rankings which may be quite meaningless. Two algorithms j and k may have different error rates, thus being assigned different ranks, despite the fact that the error rates may differ only slightly. If we were to conduct a significance test on the difference of two averages, it could show they are *not* significantly different.

With the SRR method the ratio of the success rates of two algorithms which are not significantly different is close to 1, thus, we expect that this problem

has small impact. The same problem should not happen with SW, although the statistical tests on which it is based are liable to commit errors [7].

The results obtained indicate that none of the methods seem to be influenced by this problem. However, it should be noted that the $\bar{C}$ measure used to evaluate the ranking methods equally does not take the significance of the differences into account, although, as was shown in [17], the problem does not seem to affect the overall outcome.

Robustness. Taking the magnitude of the difference in performance of two algorithms into account makes SRR liable to be affected by outliers, i.e. datasets where the algorithms have unusual error rates. We, thus, expect this method to be sensitive to small differences in the pool of the training datasets. Consider, for example, algorithm `ltree` on the `glass` dataset. The error rate obtained by `ltree` is higher than usual. As expected, the inclusion of this dataset affects the rankings generated by the method, namely, the relative positions of `ltree` and `c5boost` are swapped.

This sensitivity does not seem to significantly affect the rankings generated, however. We observe that identical rankings were generated by SRR in 13 experiments of the leave-one-out procedure. In the remaining 3, the positions of two algorithms (`ltree` and `c5boost`) were interchanged. Contrary to what could be expected, the other two methods show an apparently less stable behavior: AR has 4 variations on 4 datasets and SW has 13 across 5 datasets.

6 Related Work

The interest in the problem of algorithm selection based on past performance is growing⁶. Most recent approaches exploited *Meta-knowledge* concerning the performance of algorithms. This knowledge can be either theoretical or of experimental origin, or a mixture of both. The rules described by Brodley [5] captured the knowledge of experts concerning the applicability of certain classification algorithms. Most often, the meta-knowledge is of experimental origin [1,4,10,11,18]. In the analysis of the results of project StatLog [12], the objective of the meta-knowledge is to capture certain relationships between the measured dataset characteristics (such as the number of attributes and cases, skew, etc.) and the performance of the algorithms. This knowledge was obtained by *meta-learning* on past performance information of the algorithms. In [4] the meta-learning algorithm used was `c4.5`. In [10] several meta-learning algorithms were used and evaluated, including rule models generated with `c4.5`, IBL, regression and piecewise linear models. In [11] the authors used IBL and in [18], an ILP framework was applied.

⁶ Recently, an ESPRIT project, METAL, involving several research groups and companies has started (<http://www.cs.bris.ac.uk/~cgcm/METAL>).

7 Conclusions and Future Work

We have presented three methods to generate rankings of classification algorithms based on their past performance. We have also evaluated and compared them. Unexpectedly, the statistical tests have shown that the methods have different performance and that SRR and AR are better than SW.

The evaluation of the scores obtained does not allow us to conclude that the ranking methods produce satisfactory results. One possibility is to use the statistical properties of Spearman's correlation coefficient to assess the quality of those results. This issue should be further investigated.

The algorithms and datasets used in this study were selected according to no particular criterion. We expect that, in particular, the small number of datasets used has contributed to the sensitivity to outliers observed. We are planning to extend this work to other datasets and algorithms.

Several improvements can be made to the ranking methods presented. In particular paired t tests, which are used in SW, have been shown to be inadequate for pairwise comparisons of classification algorithms [7].

Also, the evaluation measure needs further investigation. One important issue is the difference in importance between higher and lower ranks into account, which is addressed by the Average Weighted Correlation measure [16,17].

The fact that some particular classification algorithm is generally better than another on a given dataset, does not guarantee that the same relationship holds on a new dataset in question. Hence datasets need to be characterized and some metric adopted when generalizing from past results to new situations. One possibility is to use an instance based/nearest neighbor metric to determine a subset of relevant datasets that should be taken into account, following the approach described in [10]. This opinion is consistent with the NFL theorem [19] which implies that there may be *subsets of all possible applications* where the the same ranking of algorithms holds.

In the work presented here, we have concentrated on accuracy. Recently we have extended this study to two criteria — accuracy and time — with rather promising results [16]. Other important evaluation criteria that could be considered are the simplicity of its use [12] and also some knowledge-related criteria, like novelty, usefulness and understandability [8].

Acknowledgements

We would like to thank Joaquim Costa, Joerg Keller, Reza Nakhaeizadeh and Iain Paterson for useful discussions, and the reviewers for their comments. Also to João Gama for providing his implementations of Linear Discriminant and Naive Bayes, and to Rui Pereira for implementing an important part of the methods. The financial support from PRAXIS XXI project ECO, ESPRIT project METAL, Ministry of Science and Technology (plurianual support) and Faculty of Economics is gratefully acknowledged.

References

1. D.W. Aha. Generalizing from case studies: A case study. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML92)*, pages 1–10. Morgan Kaufmann, 1992. 72
2. C. Blake, E. Keogh, and C.J. Merz. Repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mlearn/MLRepository.html>. 65
3. R.J. Brachman and T. Anand. The process of knowledge discovery in databases. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 2, pages 37–57. AAAI Press/The MIT Press, 1996. 64
4. P. Brazdil, J. Gama, and B. Henery. Characterizing the applicability of classification algorithms using meta-level learning. In F. Bergadano and L. de Raedt, editors, *Proceedings of the European Conference on Machine Learning (ECML-94)*, pages 83–102. Springer-Verlag, 1994. 72
5. C.E. Brodley. Addressing the selective superiority problem: Automatic Algorithm/Model class selection. In P. Utgoff, editor, *Proceedings of the 10th International Conference on Machine Learning*, pages 17–24. Morgan Kaufmann, 1993. 72
6. W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van Den Bosch. TiMBL: Tilburg memory based learner. Technical Report 99-01, ILK, 1999. 64
7. T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998. <ftp://ftp.cs.orst.edu/pub/tgd/papers/nc-stats.ps.gz>. 72, 73
8. U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI Press/The MIT Press, 1996. 73
9. J. Gama. Probabilistic linear tree. In D. Fisher, editor, *Proceedings of the 14th International Machine Learning Conference (ICML97)*, Morgan Kaufmann, 1997. 64
10. J. Gama and P. Brazdil. Characterization of classification algorithms. In C. Pinto-Ferreira and N.J. Mamede, editors, *Progress in Artificial Intelligence*, pages 189–200. Springer-Verlag, 1995. 72, 73
11. A. Kalousis and T. Theoharis. NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337, November 1999. 72
12. D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994. 63, 64, 72, 73
13. H.R. Neave and P.L. Worthington. *Distribution-Free Tests*. Routledge, 1992. 65, 67, 68, 70
14. F. Provost and D. Jensen. Evaluating knowledge discovery and data mining. Tutorial Notes, Fourth International Conference on Knowledge Discovery and Data Mining, 1998. 64
15. R. Quinlan. *C5.0: An Informal Tutorial*. RuleQuest, 1998. 64
16. C. Soares. Ranking classification algorithms on past performance. Master’s thesis, Faculty of Economics, University of Porto, 1999. http://www.ncc.up.pt/~csoares/miac/thesis_revised.zip. 70, 73
17. C. Soares, P. Brazdil, and J. Costa. Measures to compare rankings of classification algorithms. In *Proceedings of the 7th IFCS*, 2000. 72, 73

18. L. Todorovski and S. Dzeroski. Experiments in meta-level learning with ILP. In *Proceedings of PKDD99*, 1999. 72
19. D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute, 1996. 63, 73

Hidden Markov Models with Patterns and Their Application to Integrated Circuit Testing

Laurent Bréhélin, Olivier Gascuel, and Gilles Caraux

LIRMM, Université Montpellier II
161 rue Ada, 34392 Montpellier Cedex 5, France
{brehelin,gascuel,caraux}@lirmm.fr

Abstract. We present a new model, derived from classical Hidden Markov Models (HMMs), to learn sequences of *large* Boolean vectors. Our model – *Hidden Markov Model with Patterns*, or *HMMP* – differs from HMM by the fact that it uses patterns to define the emission probability distributions attached to the states. We also present an efficient state merging algorithm to learn this model from training vector sequences. This model and our algorithm are applied to learn Boolean vector sequences used to test integrated circuits. The learned HMMPs are used as test sequence generators. They achieve very high fault coverage, despite their reduced size, which demonstrates the effectiveness of our approach.

1 Introduction

The Hidden Markov Model (or HMM) was introduced by Baum and colleagues in the late 1960s [1]. This model is closely related to probabilistic automata (PAs) [2]. A probabilistic automaton is defined by its structure, made up of states and transitions, and by probability distributions over the transitions. Moreover, each transition is associated with a letter from a finite alphabet that is generated each time the transition is ran over. An HMM is also defined by its structure, composed of states and transitions, and by probability distributions over the transitions. The difference with respect to PAs is that the letter generation is attached to the states. Each state is associated with a probability distribution over the alphabet that expresses the probability for each letter to be generated when the state is encountered.

When the structure is known, the HMM learning (or training) problem is reduced to estimating the value of its parameters – transition and generation probabilities – from a sample of sequences. A well-known approach is the *Baum-Welch* algorithm [3], which complies with the maximum likelihood principle and is a special case of the *Expectation-Maximization* (EM) algorithm [4]. This is an iterative re-estimation algorithm that ensures convergence to a local optimum. Abe and Warmuth [5] studied the training problem from a Computational Learning Theory perspective. They proved that the PA class is not polynomially trainable unless $RP=NP$, while, to the best of our knowledge, for the HMMs the question remains open. However, we can reasonably assume that the problem is not easier, and that heuristics have to be used.

In many applications, it is not possible to infer the structure of the HMM from the *a priori* knowledge we have about the problem under investigation. In this case, the HMM learning problem becomes even more difficult. We have to estimate the parameters of the structure, and also infer this structure from the learning sample. Various authors have proposed a heuristic approach derived from automata theory. It involves generalizing an initial specific automaton that accurately represents the learning sample, by iteratively merging "similar" states until a "convenient" (e.g. sufficiently general or small) structure is obtained. This principle has been successfully applied to non-probabilistic automata [6] as well as to probabilistic ones [7], and to HMMs [8].

HMMs have been used as models for sequences from various domains, such as speech signals (e.g. [9]), handwritten text (e.g. [10]) and biological sequences (e.g. [11]). In these applications, the usual approach involves using an HMM for each word or character to be recognized. Typically HMMs have a pre-determined left-to-right structure with fixed size, and they are trained using the Baum-Welch method. Moreover, these applications all learn and use HMMs for recognition purposes. In this article, we present a new application: the *Built-in Self Test for integrated circuits*. This application differs in that we use HMMs for generation purposes: an HMM is learned from a sample of sequences, which involves building a convenient structure and estimating its parameters. This HMM is then used to generate sequences similar (eventually identical) to the learning sequences. Another difference is that the alphabet in this application can be extremely large, e.g. 2^{100} , so the emission probabilities cannot be easily defined in the usual way. This led us to develop a new class of HMM that we called *HMMP* (*Hidden Markov Model with Patterns*).

The organization of this paper is as follows. In Section 2, we present the integrated circuit test; we indicate the main features of the manipulated data and explain how this problem can naturally be dealt with using HMMs. In Section 3, we define HMMPs, and in Section 4 we present an HMMP learning algorithm which uses the state merging principle. Section 5 provides experimental results of our method with classical benchmark circuits of the test community.

2 Integrated Circuit Testing

An integrated circuit manipulates Boolean values (0 or 1). It is made of *inputs*, *outputs* and internal elements that compose the body of the circuit. It is possible to apply values to the inputs and read the output values, but it is not possible to access the internal elements. These elements may be affected by various physical *faults*. We can infer the set of potential faults of a circuit because we know its logic structure. The test of a potential fault of a given circuit is achieved by using an appropriate sequence – or *test sequence* – of Boolean vectors. The vectors are sequentially applied to the inputs; when the outputs are identical to those logically expected the fault is not present, and conversely, when the fault is present the outputs are erroneous. In a test sequence, the vector application order is as important as the vectors themselves. One sequence can test several

faults, and one fault may be tested by several sequences. Note that the sequence length varies according to the faults (between two and hundreds of vectors) and that we can deduce, by simulation, all the faults detected by a given sequence.

The research of a test sequence, for a circuit and a given fault, is NP-hard [12]. *Automatic Test Pattern Generators (ATPGs)* try to circumvent the difficulty by using various heuristics, and usually provide satisfying results. We shall see in Section 5 that the fault coverage – the proportion of faults for which the ATPG finds a test sequence – is usually above 80%.

ATPGs provide sequences of patterns $\in \{0, 1, *\}^k$ (where k is the number of inputs) rather than sequences of vectors. The $*$ character means that the bit value is unimportant, i.e. if the fault occurs, it is detected regardless of the value of this bit. A Boolean pattern defines a set of vectors; a pattern with n $*$ bits represents 2^n vectors. In the same way, a sequence of patterns defines a set of vector sequences. We present below an example of three test sequences similar to those generated by an ATPG.

Sequence 1	Sequence 2	Sequence 3
****0	****0	****0
1*001	1*001	1**01
0*101	0*101	0*001
10*01	11*01	1*101
0**01		

When test sequences with high fault coverage have been obtained, we have to position the test procedure. The classical approach involves using an external tester. Due to the price of these testers and, sometimes, the fact that there is no physical access to the circuit, we often prefer the method of *built-in self test* or *BIST*. The BIST principle is to incorporate a supplementary test structure in the circuit. This structure should be able to generate test sequences and analyse the circuit responses. Response analysis is a task that has efficient solutions. This is not the case for test sequence generation. The problem is to find a generator of sequences that combines high fault coverage and small size (in terms of silicon area). Indeed, we can not physically stock all the ATPG sequences on ground of silicon cost. On the other side, we can build small generators of pseudo-random sequences, but they have low fault coverage.

Our approach consists of building, with a learning algorithm, an instance of a new class of HMM (called HMMP) that generates ATPG sequences or similar sequences with sufficiently high probability. We shall see (Section 5) that relatively small HMMPs effectively generate sequences with fault coverage equivalent to that of the ATPG.

3 The HMMP Class

The HMMs that we want to infer are intended to generate Boolean vector sequences. Then the size of the manipulated alphabet is 2^k (for our test problem, k is the number of inputs of the circuit), which is potentially very large (e.g., there

are circuits with $k > 100$). Common HMMs do not fit the modeling of such sequences. We define here a new class of HMMs, named *Hidden Markov Model with Patterns* (or *HMMPs*), specially adapted to deal with this problem. Owing to the specificity of the test sequences (we learn from patterns, i.e. from sets of vectors and not from simple vectors; moreover, we perform generation and not recognition), we use symbols and operators specific to these data and to the problem. However, HMMPs can also be used to model more conventional vector sequences (Boolean or not). This point is discussed in Section 6.

3.1 Presentation

An HMMP H is defined by a triplet $\langle S, P, M \rangle$, where

- S is a finite set of states; S contains two special states *start* and *end* which are used to initiate and conclude a sequence respectively. Each state of S , except *start* and *end*, is labeled by a pattern from P .
- $P = \{p_s, s \in S - \{\text{start}, \text{end}\}\}$ is the set of patterns associated with the states; p_s is the pattern associated with state s .
- $M : S - \{\text{end}\} \times S - \{\text{start}\} \rightarrow [0, 1]$ is the matrix that contains transition probabilities between states. M defines the probability distributions associated with states of H . We have: $\forall s, t, M(s \rightarrow t) \geq 0$, and $\forall s, \sum_{t \in S} M(s \rightarrow t) = 1$.

The *structure* of an HMMP is the set of its states and of its non-zero transitions. Figure 1 gives an example of HMMP with six states and seven transitions.

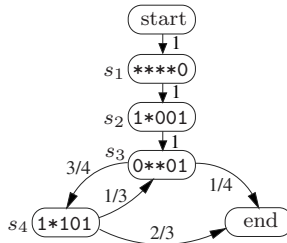


Fig. 1. Example of HMMP: each state is labelled by its name and associated pattern; transitions are labeled by their transition probability

3.2 Generating Vector Sequences with an HMMP

The procedure involves beginning on the *start* state, running over the transitions and generating a test vector for each state encountered by using the pattern associated with that state. The test vectors generated by a given pattern are consistent with this pattern and, moreover, the $*$ has equal probability of generating a 0 and a 1. For example, pattern $*1*$ generates, with probability $1/4$, each of

the 4 vectors 010, 011, 110 and 111. Once a test vector has been generated, we choose, according to the associated probabilities, one transition starting from the current state, and then go to the targeted state. This procedure is continued until the *end* is reached. A sequence of vectors is thus generated, and another sequence can eventually be generated by going onto the *start* again.

3.3 Generation Probability of a Set of Sequences

The pattern p is said to be *compatible* with the pattern p_s if its fixed bits (those with value 0 or 1) have the same value or the value $*$ in p_s . For example, $p = 11*$ is compatible with $p_s = 1*0$, while $p = 11*$ is not compatible with $p_s = 100$. The probability is zero that the state s will generate a pattern p that is not compatible with p_s . The generation probability by a state s of a pattern p compatible with p_s depends on the number of bits which are fixed in p but have the value $*$ in p_s . Let $*_{p_s}^p$ denote this number. For example, if $p = 10**$ and $p_s = 1***$ then $*_{p_s}^p = 1$. Since $*$ has equiprobability of generating a 0 or a 1, the probability of generating the compatible pattern p on state s is given by the formula:

$$P(p|s) = \left(\frac{1}{2}\right)^{*_{p_s}^p}. \quad (1)$$

For example, if $p = 1*01*$ and $p_s = 1*010$, then $P(p|s) = 1$. On the other hand, if $p = 1*010*$ and $p_s = ***10*$, then $P(p|s) = \frac{1}{4}$.

Let $x = p_1 p_2 \dots p_l$ be a sequence of patterns. A common method for computing the generation probability of x by an HMM H is to make the *Viterbi* assumption [13] that x can only be generated by a unique path (or sequence of states) through H . In other words, all paths except the most likely are assumed to have a negligible (or null) probability of generating x . This path is called the *Viterbi path* of x . For example, the Viterbi path of the first sequence of Section 2 in the HMMP of Figure 1 is $start - s_1 - s_2 - s_3 - s_4 - s_3 - end$. Moreover, this is the only path that can generate this sequence – which often occurs in practice –, and the Viterbi assumption holds in this case.

Let $V_x = v_{p_0} \dots v_{p_{l+1}}$ (with $v_{p_0} = start$ and $v_{p_{l+1}} = end$) be the Viterbi path of the sequence $x = p_1 \dots p_l$. Then, under the Viterbi assumption, the generation probability of x by H is:

$$P(x|H) = \left(\prod_{i=0}^{l-1} M(v_{p_i} \rightarrow v_{p_{i+1}}) P(p_{i+1}|v_{p_{i+1}}) \right) M(v_{p_l} \rightarrow v_{p_{l+1}}).$$

For the above sequence, we have: $M(start \rightarrow s_1) = 1$, $P(p_1|s_1) = 1$, $M(s_1 \rightarrow s_2) = 1$, $P(p_2|s_2) = 1$, $M(s_2 \rightarrow s_3) = 1$, $P(p_3|s_3) = 1/2$, $M(s_3 \rightarrow s_4) = 3/4$, $P(p_4|s_4) = 1/2$, $M(s_4 \rightarrow s_3) = 1/3$, $P(p_5|s_3) = 1$, $M(s_3 \rightarrow end) = 1/4$. It follows that the probability of HMMP of Figure 1 generating this sequence is equal to: $1/2 \times 3/4 \times 1/2 \times 1/3 \times 1/4 = 1/64$.

Let X be a set of sequences and $V = \{V_x, x \in X\}$ the set of Viterbi paths associated with the sequences of X . The probability $P(X|H)$ of generating, with $|X|$ trials, the set X using H , is obtained (under the same assumption) by the following formula:

$$P(X|H) = |X|! \prod_{x \in X} P(x|H),$$

that can be rewritten as:

$$P(X|H) = |X|! \prod_{s \in S} \left(\prod_{p \in P_{X,s}} P(p|s)^{n_s^p} \prod_{t \in Out(s)} M(s \rightarrow t)^{n_{s \rightarrow t}} \right), \quad (2)$$

where $P_{X,s}$ is the set of patterns generated by s (in V), n_s^p is the number of times s generates the pattern p ($\in P_{X,s}$), $n_{s \rightarrow t}$ is the number of times the transition $s \rightarrow t$ is used, and $Out(s)$ is the set of states t for which there is a transition $s \rightarrow t$.

4 Learning HMMPs

Let X be a set of pattern sequences (for example obtained from an ATPG). Our aim is to build an HMMP of low size (i.e. with a low number of states and transitions), and that generates X with probability as high as possible. We designed a learning algorithm for this purpose which is based on the state merging generalization method.

4.1 Main Algorithm

The main algorithm – HMMPLEARNING – of the learning procedure proceeds in a greedy ascending way. First, it builds with the INITIALHMMPBUILDING procedure (Line 1) an initial specific HMMP that represents the sequences of X . Next, at each step of the algorithm, the BESTSTATEPAIR procedure (Line 2) selects the state pair that, when merged, involves the lowest loss of probability of generating X . If several pairs have the same probability loss, it chooses the pair that involves the nearest patterns. The selected state pair is merged with the MERGE procedure (Line 3), which modifies the structure of the HMMP and updates its parameters. The algorithm iterates this procedure until the desired number of states (N) is reached.

Figure 2 details six steps of the algorithm when applied to the three sequences of Section 2.

4.2 Building the Initial HMMP

The initial HMMP H_0 is obtained by building the *prefix tree* of X . In such a tree, each path from the root to a leaf corresponds to a sequence of X , and the common prefixes are not repeated but represented by a unique path starting

Algorithm 1: HMMPLEARNING(X, N)

```

Data :  $X, N$ 
Result :  $H$ 
1  $H \leftarrow \text{INITIALHMMPBUILDING}(X)$ ;
  while Number of states of  $H > N$  do
2    $(s_1, s_2) \leftarrow \text{BESTSTATEPAIR}(H)$ ;
3    $H \leftarrow \text{MERGE}(H, s_1, s_2)$ ;
  return  $H$ ;

```

from the tree root. In our case, the root represents the *start* state. Next, to each state s we attach its pattern p_s (except for the *start* with which no pattern is associated) and the number of times (n_s) it is used in the Viterbi paths. In H_0 , Viterbi paths are naturally described by sequences and the Viterbi assumption holds. Therefore, n_s is equal to the number of leaves of the sub-tree with root s , and, in the same way, $n_{s \rightarrow t}$ is equal to the number of leaves of the sub-tree with root t . Values of both parameters (n_s and $n_{s \rightarrow t}$) are stored. Moreover, we set $P_{X,s} = \{p_s\}$ and $n_s^{p_s} = n_s$.

According to the maximum likelihood principle, the values of the transition probabilities associated with the edges of H_0 are estimated by maximizing $P(X|H_0)$. Maximizing Expression (2) is equivalent to maximizing each of its sub-products. Therefore, we estimate the transition probabilities by maximizing the expressions $\prod_{t \in \text{Out}(s)} M(s \rightarrow t)^{n_{s \rightarrow t}}$. Each of these expressions is identical to the probability distribution of a multinomial law and is maximized by

$$M(s \rightarrow t) = \frac{n_{s \rightarrow t}}{n_s}. \quad (3)$$

Finally, we create the *end* state to which every leave is linked with transition probability 1. The HMMP obtained is the most specific, in that it describes all sequences of X , but only these sequences.

The HMMP H_0 of Figure 2 is the initial HMMP obtained from the sequence set of Section 2. Each sequence has probability $1/3$ of being generated by H_0 . Therefore, $P(X|H_0) = 3! \times 1/3 \times 1/3 \times 1/3 = 6/27$.

4.3 State Merging

When two states s_1 and s_2 have been selected (the criterion used is described in Section 4.4), they are merged. States s_1 and s_2 are deleted and replaced by a new state s . The *in* and *out* edges from s_1 and s_2 are connected to s , and the potential double transitions (e.g., $t \rightarrow s_1$ and $t \rightarrow s_2$) are also merged. An example of state merging is provided in Figure 3.

The structure of the HMMP is modified and its parameters have to be updated. We assume (as usual) that the Viterbi paths are not altered by merging, and the new Viterbi paths are inferred from the previous ones by replacing s_1 and s_2 by s . This assumption provides an efficient way of updating parameters associated with the new state and to its adjacent edges. We have: $n_s =$

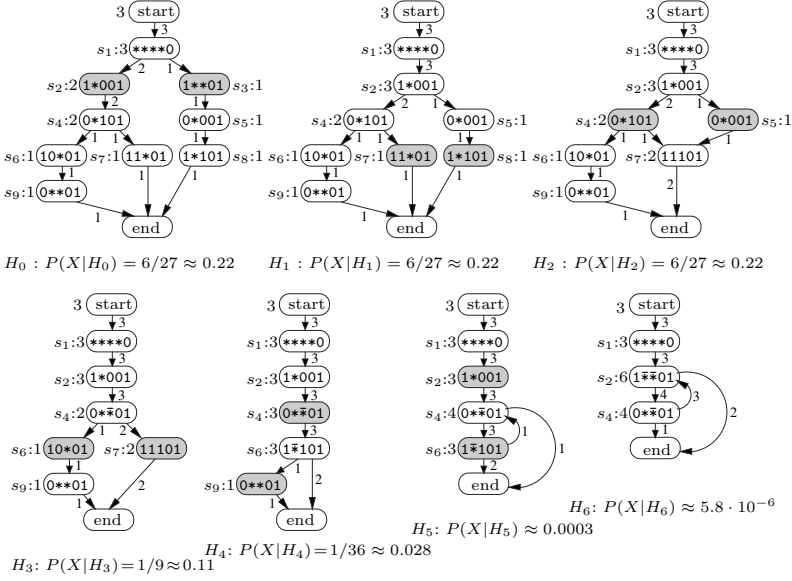


Fig. 2. Generalization achieved by six state mergings. The HMMPs are obtained by merging the grey nodes. For each HMMP, we indicate the probability of generating the learning sequences. Each state is labeled by its name and the number of times n_s it is used in the Viterbi paths. Each edge is also labeled by the number of times $n_{s \rightarrow t}$ it is ran over. The transition probability associated with $s \rightarrow t$ is equal to the ratio $n_{s \rightarrow t}/n_s$ (*c.f.* Formula (3))

$n_{s_1} + n_{s_2}$, $P_{X,s} = P_{X,s_1} \cup P_{X,s_2}$, and $\forall p \in P_{X,s}$, $n_s^p = n_{s_1}^p + n_{s_2}^p$. Moreover, for edges adjacent to s , we have: $n_{s \rightarrow t} = n_{s_1 \rightarrow t} + n_{s_2 \rightarrow t}$, $n_{t \rightarrow s} = n_{t \rightarrow s_1} + n_{t \rightarrow s_2}$ and $n_{s \rightarrow s} = n_{s_1 \rightarrow s_1} + n_{s_1 \rightarrow s_2} + n_{s_2 \rightarrow s_2} + n_{s_2 \rightarrow s_1}$. Note that only parameters associated with the new state and its adjacent edges need to be updated; the merging has no effect on the other states and transitions. Finally, the transition probabilities attached to the updated edges are computed using Formula (3).

The pattern associated with the new state must generate, with the highest possible probability, all patterns of $P_{X,s}$. It can be computed by merging all these patterns bit after bit, but a more efficient method is to merge bit after bit

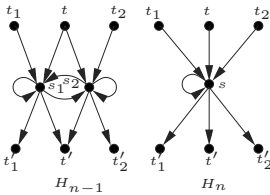


Fig. 3. States and transitions before and after the merging of s_1 and s_2

γ	1	0	*	$\bar{*}$
1	1	$\bar{*}$	1	$\bar{*}$
0	$\bar{*}$	0	0	$\bar{*}$
*	1	0	*	$\bar{*}$
$\bar{*}$	$\bar{*}$	$\bar{*}$	$\bar{*}$	$\bar{*}$

Fig. 4. Response table of the bit merging operator

the patterns p_{s_1} and p_{s_2} attached to s_1 and s_2 . Let γ denote the bit merging operator. The character $*$ means that the value of the bit is not important. Therefore, $\gamma(*, 0) = 0$ and $\gamma(*, 1) = 1$. But two patterns are not always mutually compatible. In this case, some of their bits differ and the merged pattern must generate 0 or 1 with equal probability on these bits. These bits take the value $*$ in the merged pattern and are marked to store the fact that their value results from the merging of a 0 and a 1. Let $\bar{*}$ denote the marked $*$. During further steps of the learning algorithm, this information is needed so that we do not set this bit at value 0 or 1. The merging operator takes this into account, and we have: $\gamma(0, 1) = \bar{*}$, and during the further steps, $\gamma(\bar{*}, 0) = \gamma(\bar{*}, 1) = \gamma(\bar{*}, *) = \bar{*}$. Figure 4 summarizes the result of operator γ .

4.4 Selection of the Best State Pair

The aim of the learning algorithm is to reduce the HMMP structure while keeping (as much as possible) a high probability (given by Formula (2)) of generating X . At each step, the algorithm chooses the state pair which, when merged, involves the lowest loss of probability of generating X . Nevertheless, sometimes (especially at the beginning of the process) many pairs agree with this criterion. Then, we choose among these pairs that for which pattern merging involves fixing the lowest number of bits. The number of bits fixed by the merging of two patterns p and p' is obtained by formula

$$\varphi(p, p') = \min(*_p^{\gamma(p, p')}, *_{p'}^{\gamma(p, p')}). \quad (4)$$

For example, if $p = 1*11*$ and $p' = *0\bar{*}**$, then $\gamma(p, p') = 10\bar{*}1*$; we have $*_{\gamma(p, p')}^p = 1$, $*_{\gamma(p, p')}^{p'} = 2$ and then $\varphi(p, p') = 1$. Note that if p is more general than p' , then $\varphi(p, p') = 0$. Using Formula (4) has two justifications. First, at the beginning of the learning procedure, this avoids fixing bits in the patterns too soon, which would make further mergings more difficult (in terms of likelihood). Second, we can reasonably assume that similar patterns quite likely play the same part in sequences.

In Figure 2, two state pairs of H_0 involve a null loss of generation probability: (s_2, s_3) and (s_7, s_8) . However, $\varphi(p_{s_2}, p_{s_3}) = 0$, while $\varphi(p_{s_7}, p_{s_8}) = 1$. Therefore, (s_2, s_3) is selected and merged to obtain H_1 .

At each step of the algorithm, this selection criterion involves computing the loss of generation probability for every state pair, i.e. calculating $O(n^2)$ times Formula (2), where n is the number of states in the current HMMP. When the initial HMMP contains numerous states (i.e. when the learning sequence set is large) this yields prohibitive computing time. A more efficient algorithm is proposed in [14]. It make use of the fact that merging has no effect on the states and transitions which are not adjacent to the merged states. It follows that the performance of a pair is not really affected, unless one of its states is adjacent to the merged states. Therefore, we initially compute the $O(n^2)$ criterion values for all pairs, and further we only update the values of the adjacent pairs, that is $O(nb)$, where b is the branching factor of the HMMP. Moreover, b remains relatively low

during the learning process. Indeed, our selection criterion (minimizing the loss of $P(X|H)$) tends to lower the number of out transitions (*c.f.* Formula (2)). For example, with $|Out(s)| = 1$ we have $\prod_{t \in Out(s)} M(s \rightarrow t)^{n_{s \rightarrow t}} = 1$, while with $|Out(s)| = 2$ and $n_{s \rightarrow t_1} = n_{s \rightarrow t_2}$, we have $\prod_{t \in Out(s)} M(s \rightarrow t)^{n_{s \rightarrow t}} = (1/2)^{n_s}$.

5 Experimental Results

After the learning phase, the HMMP can be physically implemented as test sequence generator. We do not describe this procedure (the interested reader can consult [15]). It consists in a natural microelectronic translation of the HMMP structure; the size of the implementation (an crucial factor for the validity of our approach) is strongly connected to the size of the HMMP (in terms of number of states and transitions).

The performances of our method were tested on the classical benchmark¹ circuits of the electronic community. The results are reported in Table 1. After the name of the circuit, the number of its inputs (#I.) and potential faults (#F.), we provide the fault coverage and the total length (in thousands of patterns) of ATPG sequences. For comparison purposes, we include the fault coverage achieved by a *long* Boolean vector sequence generated by a pure random process (only simulating one long sequence is highly justified in the case of a purely random approach, and no improvement is obtained by decomposing this long sequence into many shorter ones).

For every circuit, we computed by simulation the fault coverage of 10 random sequences, and the fault coverage of 10 sets of test sequences generated with the HMMP learned from the ATPG sequences. The T.Len. column provides both the length of the random sequences and the total length of the set of HMMP sequences. This length has been manually tuned for each circuit, in order to obtain a sufficient fault coverage. The following column (ratio) provides the ratio of this length over the total length of ATPG sequences. The %Best columns indicate the best fault coverage achieved in the 10 simulations; the %Av. columns indicate the average of these 10 fault coverages. Columns #S and #E provide the number of states and the number of transitions of the learned HMMP respectively. On the bottom line, we report the means of these quantities.

First, it can be seen that the fault coverage achieved by using learned HMMP is much larger than the fault coverage of the random sequence. Next, we observe that it is often equal (s298, s1494), sometimes little smaller (s820, s832) and sometimes larger (s444, s526 and general mean) than the fault coverage of the ATPG sequences. This result is surprising and is a good confirmation of our method. It demonstrates that it is possible to infer very efficient construction rules from ATPG sequences. These rules do not ensure accurate generation of the original sequences, but they achieve high fault coverage when the generated sequence set is large enough. The good results obtained with our method could

¹ These benchmarks were created for the *International Symposium on Circuits & Systems (ISCAS)* in 1989. They are available at <http://www.cbl.ncsu.edu/benchmarks/>

Table 1. Fault coverage achieved by ATPG, random sequences and HMMPs

CIRCUIT			ATPG		Len. R.H.		RANDOM		HMMP			
Name	#I.	#F.	%Cov.	Leng	T.Len	ratio	%Best	%Av.	%Best	%Av.	#S.	#E.
s298	3	596	89.9	2408	2.5K	1.03	75.2	68.1	89.9	89.3	5	19
s344	9	670	97.0	427	1K	2.34	96.1	94.2	97.6	97.6	3	7
s382	3	764	85.7	9178	15K	1.63	15.4	15.4	96.7	96.3	5	13
s386	7	772	90.2	754	5K	6.63	74.0	68.8	89.6	89.3	7	20
s444	3	888	75.5	2074	10K	4.82	13.4	13.3	97.1	92.5	4	9
s526	3	1052	52.9	966	10K	10.35	10.8	10.7	85.2	79.3	5	10
s820	18	1640	96.3	4993	30K	6.00	49.4	48.4	93.3	91.2	13	40
s832	18	1664	95.3	5024	30K	5.97	47.3	46.3	93.6	90.6	13	43
s991	65	1948	99.2	1139	6K	5.27	93.8	93.6	96.9	96.6	8	25
s1488	8	2976	95.6	6776	30K	4.42	78.0	75.1	98.6	98.5	8	30
s1494	8	2988	98.1	6723	30K	4.46	78.3	75.1	98.1	98.0	8	29
s3330	40	6660	79.2	5616	30K	5.34	76.5	74.4	78.5	78.1	13	46
Avg:			87.9			4.85	59.0	56.9	92.9	91.4		

also be explained by the weakness of some ATPG sequence sets (and by the NP-hardness of the task) and by the easiness of achieving relatively high fault coverages for some circuits (see the results obtained by the random method).

6 Conclusion

We presented a new probabilistic model for learning Boolean pattern sequences, and its application for testing integrated circuits. This model is close to the classical HMM, but differs in that it defines the emission probability distribution with a boolean pattern. Moreover, we use this model for generation purposes and not for recognition, as in most HMM applications. Experimental results indicate that our model is well adapted for testing integrated circuits. Nevertheless, the fault coverage achieved for some circuits may be improved. A possible solution could be to weight ATPG sequences by the number of faults they detect.

HMMPs were defined to model pattern sequences. This notion of pattern – a set of vectors – is relatively specific to the test problem. Moreover, due to the generation aim of the test problem (with the constraint of generating at least one vector sequence from each pattern sequence computed by the ATPG), our γ operator performs specialization ($\gamma(0, *) = 0$ and $\gamma(1, *) = 1$) and not only generalization as is usually done in the recognition framework.

Nevertheless, as stated in the introduction, HMMPs could be used to model more usual vector sequences (as time series for example). In this case, the non-marked character $*$ does not appear, and γ only performs generalization in the usual sense: $\gamma(0, 1) = \gamma(0, \bar{*}) = \gamma(1, \bar{*}) = *$. Moreover, the φ function is useless (it returns 0 for every pair) and the likelihood is the only selection criterion. Therefore, for Boolean vector sequences, the γ operator and our learning algorithm could be used directly. For sequences of vectors with discrete variables

(ordinal or not), slight modifications of the model, and of the learning procedure (especially of γ) are easily conceivable.

The Boolean patterns define very simple probability distributions over $\{0, 1\}^k$. Numerous more expressive distribution classes could be envisaged, such as, for example, using a Bernoulli distribution associated with each bit. However, the simplicity of Boolean patterns is well suited for microelectronic purpose. Indeed, the $*$ is free and the 1 and 0 have very low cost in terms of silicon area overhead [15], while implementing continuous probabilities is much more expensive. Moreover, our Boolean patterns are very similar to the condition parts of the rules used in symbolic and hybrid classification methods [16], and, just as in these methods, the simplicity of these descriptions is associated to explanatory virtues which should be of interest from a modeling and learning perspective.

References

1. Baum L. E., Petrie T., Soules G., Weiss N.: A maximization technique occurring in statistical analysis of probabilistic functions in Markov Chains. *The Annals of Mathematical Statistics* **41** (1970) 164–171 75
2. Casacuberta F.: Some Relations Among Stochastic Finite State Networks Used in Automatic Speech Recognition *IEEE PAMI* **12** (1990) 691–695 75
3. Baum L. E.: An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. *Inequalities* **3** (1972) 1–8 75
4. Dempster A. P., Laird N. M., Rubin D. B.: Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B* **39** (1977) 1–38 75
5. Abe N., Warmuth M.: On the Computational Complexity of Approximating Distributions by Probabilistic Automata. *Machine Learning* **9** (1992) 205–260 75
6. Angluin D., Smith C.H.: Inductive Inference: Theory and Methods. *ACM Computing Surveys* **15** (1983) 237–269 76
7. Carrasco R. C., Oncina J.: Learning Stochastic Regular Grammars by Means of a State Merging Method. *Proceedings of the Second International ICGI* (1994) 139–152 76
8. Stolcke A., Omohundro S.: Inducing Probabilistic Grammars by Bayesian Model Merging. *Proceedings of the ICGI* (1994) 106–118 76
9. Rabiner L. R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* **77** (1989) 257–285 76
10. Schenkel M., Guyon I., Henderson D.: On-line cursive script recognition using time delay neural networks and hidden Markov models. *Proc. ICASSP '94* (1994) 637–640 76
11. Krogh A., Brown M., Saira Mian I., Sjolander K., Haussler D.: Hidden Markov Models in computational Biology: Applications to protein modelling. *UCSC-CRL-93-32* (1993) 76
12. Ibarra, Sahni: Polynomially Complete Fault Detection Problems. *IEEE/TC: IEEE Transactions on Computers* **24** (1975) 77
13. Forney G. D.: The Viterbi algorithm. *Proc. IEEE* **61** (1973) 268–278 79
14. Bréhélin L., Gascuel O., Caraux G.: Hidden Markov Models with Patterns to Learn Boolean Vector Sequences; Application to the Built-in Self-Test for Integrated Circuits. *TR 99006* (1999) 83

15. Bréhélin L., Gascuel O., Caraux G., Girard P., Landrault C.: Hidden Markov and Independence Models with Patterns for Sequential BIST. 18th IEEE VLSI Test Symposium, In Press (2000) [84](#), [86](#)
16. Gascuel O. and the SYMENU group: Twelve numerical, symbolic and hybrid supervised classification methods. International Journal of Pattern Recognition and Artificial Intelligence (1998) 517–572 [86](#)

Comparing Complete and Partial Classification for Identifying Latently Dissatisfied Customers

Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets

Limburg University Centre, Universitaire Campus, gebouw D
B-3590 Diepenbeek, Belgium
tom.brijs@luc.ac.be

Abstract. This paper evaluates complete versus partial classification for the problem of identifying latently dissatisfied customers. Briefly, latently dissatisfied customers are defined as customers reporting overall satisfaction but who possess typical characteristics of dissatisfied customers. Unfortunately, identifying latently dissatisfied customers, based on patterns of dissatisfaction, is difficult since in customer satisfaction surveys, typically only a small minority of customers reports to be overall dissatisfied and this is exactly the group we want to focus learning on. Therefore, it has been claimed that since traditional (complete) classification techniques have difficulties dealing with highly skewed class distributions, the adoption of partial classification techniques could be more appropriate. We evaluate three different complete and partial classification techniques and compare their performance on a ROC convex hull graph. Results on real world data show that, under the circumstances described above, partial classification is indeed a serious competitor for complete classification. Moreover, external validation on holdout data shows that partial classification is able to identify latently dissatisfied customers correctly.

1 Introduction

Latently dissatisfied customers are customers who report overall satisfaction but who possess typical characteristics of customers reporting overall *dissatisfaction*. In this sense, latently dissatisfied customers constitute an important - but hidden - group that should not be ignored by the management. Indeed, because of their possession of characteristics of overall dissatisfied customers, latently dissatisfied customers have a high probability of becoming overall dissatisfied in the near future and, as a result, they are potential defectors. Therefore, we argue that the identification of latently dissatisfied customers may act as an early warning signal, providing the opportunity to correct a problem before customers decide to defect.

There are mainly two methodological approaches to solve this problem. The first one entails the construction of a classical *complete* classification model (such as decision trees) which has the objective of discriminating between overall satisfaction (negative class) and dissatisfaction (positive class). In this setting, latently dissatisfied instances are considered as false positive (FP) instances, i.e. instances reporting overall satisfaction but who are missclassified by the model as overall dissatisfied. The second approach is based on the construction of a *partial*

classification model (such as an association rules ruleset). The motivation is that previous researchers [3] have argued that, under specific circumstances, the use of classical classification models may be inappropriate and partial classification systems should be used instead. More specifically, and especially relevant in our study, the presence of a very skewed class distribution and, at the same time, the intention to concentrate learning on the low-frequency class (overall dissatisfied customers) advocates the use of a partial classification technique.

The paper is organised as follows. Firstly, we will elaborate on the different methodological approaches to the problem of identifying latently dissatisfied customers. In the second part, an empirical comparison of different techniques on real-world data will be carried out. The objective is to make a comparison in terms of a common performance criterion, such as the ROC convex hull graph [10]. In addition, validation will be carried out on separate testing data. The final section will be reserved for conclusions.

2 Alternative Methodological Approaches

2.1 Approach 1: Complete Classification

The complete classification approach assumes that a classification model can be built that discriminates between overall dissatisfied (positive instances) and overall satisfied (negative instances) customers in the dataset. The term *complete* classification stems from the fact that the model covers *all* instances and *all* classes in the data. Consequently, from the methodological point of view, latently dissatisfied instances can then be defined as false positive (FP) classifications.

In the past, most of the attention in research has been devoted to these kind of classification techniques. In this study, we will concentrate on two well-known complete decision tree classification techniques, i.e. C4.5 [11] and CART [5].

2.2 Approach 2: Partial Classification

The term *partial classification* refers to the discovery of models that show characteristics of the data classes, but may not necessarily cover all classes and all instances of a given class. In fact, the aim of partial classification techniques is to learn rules that are *individually* accurate and, thus not to predict future values, but rather to discover the necessary or most prevalent characteristics of some of the data classes [3]. Especially in domains where the class distributions are very skewed and the user is especially interested in understanding the low-frequency class, partial classification can be preferred over complete classification.

Consequently, in the case of partial classification, latently dissatisfied customers are identified somewhat differently. Firstly, characteristics are generated (in terms of frequently co-occurring attribute-value combinations) that are prevalent within the group of overall dissatisfied customers. Given these frequently co-occurring attribute-value combinations, customers in the other group (overall satisfied) that have similar characteristics, are selected. We call the latter group latently dissatisfied

because customers in this group report overall satisfaction although they possess characteristics that are prevalent to overall dissatisfied customers.

In this paper, we will highlight one specific partial classification technique, i.e. association rules.

2.2.1. Association Rules

Association rules [2] were first introduced as a technique to discover hidden purchase patterns in large sales transaction databases, also known as market basket analysis. In such a context, a typical association rule might look like *beer* $\Rightarrow$ *diapers*, indicating that customers who buy beer also tend to buy diapers with it. Recently, however, other applications of association rules have been put forward [3, 4].

Finding association rules in large databases typically involves two phases. In the first phase, frequent itemsets are discovered, i.e. all combinations of items that are sufficiently supported by the transactions (i.e. exceed some predefined minimum support threshold). In the second phase, frequent itemsets are used to generate association rules that exceed a user-defined confidence threshold. The general idea is that if, say, *ABCD* and *AB* are frequent itemsets, then it can be determined if the rule *AB* $\Rightarrow$ *CD* holds by calculating the ratio $r = \text{support}(ABCD) / \text{support}(AB)$. Detailed information on how to perform each of the two phases can be found in [1, 2].

2.3 Comparison of Both Approaches

Conceptually, the difference between complete and partial classification models can be illustrated as shown in figure 1.

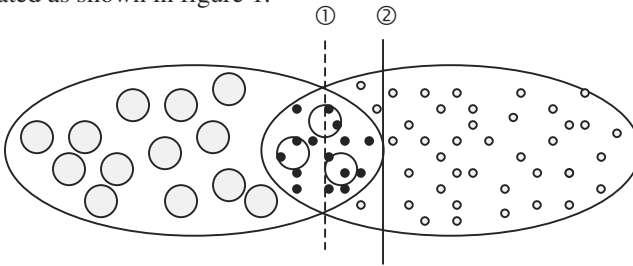


Fig. 1. Manifestly dissatisfied customers are represented as big white dots, whereas small white dots represent manifestly satisfied instances. Latently dissatisfied customers are represented as small black dots

Figure 1 illustrates that it is the objective of *complete* classification to *discriminate* between positive (dissatisfied) and negative (satisfied) instances. In the case of decision trees, this involves the discovery of several (quasi pure) multi-dimensional cubes in the multi-dimensional instance space. However, in the case of *partial* classification, the objective is to find the necessary or most prevalent characteristics of the target class, i.e. to find a description of the target group which is as complete as possible (i.e. which covers as many positive instances as possible).

As pointed out in the introduction, the distinction between complete and partial classification also entails a different way of identifying latently dissatisfied customers. Indeed, in the case of *complete* classification, latently dissatisfied customers are defined as customers who are missclassified as being ‘overall dissatisfied’ which corresponds to instances in the intersection situated on the left-hand side of line 1. In contrast, in the case of *partial* classification, latently dissatisfied customers are those black dots situated on the left-hand side of line 2. This is obvious since the objective of partial classification is to characterise the positive group as completely as possible, causing some of the descriptions (rules) to cover instances of the negative group as well.

Finally, there exists a relationship between line 1 and 2. Namely, in the case of complete classification, increasing the cost of false negative (FN) errors will cause line 1 to shift into the direction of line 2. Indeed, increasing the cost of NF errors will increase the true positive (TP) and false positive (FP) rate, and thus line 1 will shift to the right. In analogy, line 2 will shift into the direction of line 1 by lowering the support (coverage) threshold of the association rules.

3 Empirical Evaluation

3.1 Data

The data being used in this study comes from a large-scale anonymous customer satisfaction survey carried out by a major Belgian bank in 1996. Data were obtained for a random sample of 7264 customers.

3.1.1 Satisfaction Opinions of Different Service Items

Customers were asked about their satisfaction with 16 service items of the bank including questions related to the empathy of the staff (e.g. friendliness), information and communication (e.g. investment advice), and finally the practical organisation of the bank office (e.g. waiting time). Each question (i.e. each attribute in this study) was measured on a 5-point rating scale as illustrated by the following example:

	Never	Seldom	Sometimes	Often	Always	No-opinion
<i>I have to queue for a long time</i>	☐	☐	☐	☐	☐	☐

One specific question probed for the overall level of (dis)satisfaction of the customer. This question was used to allocate customers into two groups: overall satisfied or overall dissatisfied, and it is the target variable in our study.

3.1.2 Complaints Behaviour

Finally, information was collected with regard to the number and type of complaints that a customer had placed during 1996.

3.2 Data Recoding

Figure 2 on the next page illustrates the distribution of the different attribute values, both for the independent variables and the target attribute (i.e. the question probing for the overall level of satisfaction).

Figure 2 shows that, for all attributes, there exists only a very small tendency to be (manifestly) dissatisfied. This has important implications with respect to the construction of appropriate classification models. Indeed, for many attributes the number of customers responding (manifest) dissatisfaction is too low to guarantee statistically reliable models. For instance, in the case of decision trees, already in the very beginning of the growing process of the tree, there will exist nodes with very few instances producing terminal nodes that will contain very few observations, and as a consequence, their classification label will be very doubtful.

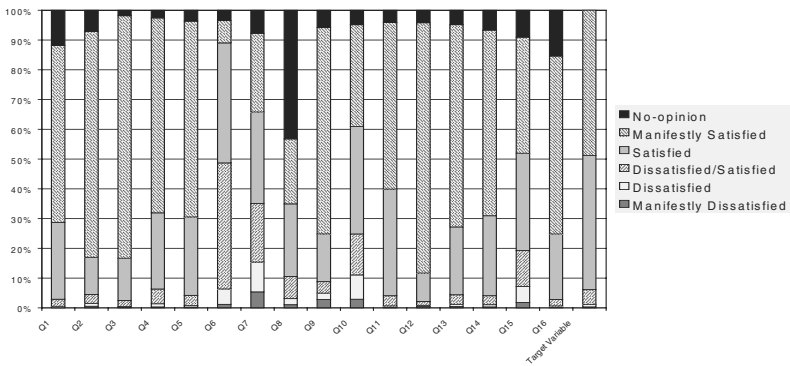


Fig. 2. Distribution of answer patterns on different questions in the survey

To overcome these problems, it is suitable to group certain attribute values to obtain more observations per grouped attribute value, of course with the drawback of losing some detailed information. More specifically, 5 attribute values were recoded into 3 new values, i.e. the answers 'manifestly dissatisfied', 'dissatisfied' and 'dissatisfied/satisfied' were grouped into one new attribute value, with the other attribute values unchanged. Moreover, the target variable was converted into a binary attribute. More specifically, an aggregate value 'overall dissatisfaction' was constructed (still only containing 6.1% of all instances), grouping the attribute values 'manifestly dissatisfied', 'dissatisfied' and 'dissatisfied/satisfied', and an aggregate attribute value 'overall satisfaction' (containing 93.9% of all instances) containing the attribute values 'satisfied' and 'manifestly satisfied'.

3.3 Empirical Evaluation of the Different Techniques

3.3.1 Experimental Design

More specifically, for the complete classification approach, C4.5 was carried out with and without misclassification costs, and with and without grouping of symbolic

values in the tree (C4.5 GSV). The use of misclassification costs is justified because of the skewed class distributions in the data and the grouping of symbolic values in the tree is enabled to obtain a fair comparison with CART which produces *binary* splits. CART was also carried out with and without misclassification costs to adjust the prior probabilities of the target classes.

For the partial classification approach, different association rule models were induced as well. Firstly, we generated all frequent combinations of attribute values for instances of the target class (overall dissatisfied) with a minimum support threshold of 20%. The outcome is the set of all combinations of attribute values that appeared together in the target class with frequency exceeding the minimum support threshold. The support threshold is used to guarantee the discovery of *prevalent* patterns of dissatisfaction. In total, 97 rules for dissatisfaction were obtained.

Secondly, different models of association rules were obtained by modifying the number of rules retained according to some measure of interestingness [4]. This is necessary because the discovered characteristics may also be characteristics of the complete dataset as they represent the necessary but not the sufficient condition for the membership of the positive example set. The following measure was used [4]:

$$Interest_R = S_{Target} - S_{Total} / \max\{S_{Target}, S_{Total}\} \quad (1)$$

where, S_{Target} (resp. S_{Total}) is the support of the rule in the target class (resp. total database). The denominator is introduced to normalise the interestingness between $[-1, +1]$.

3.3.2 The ROC Convex Hull Method

In order to compare the performance of different classification methods on a common basis, we choose the ROC convex hull method [10] because it is robust to imprecise class distributions and misclassification costs. The method decouples classifier performance from specific class and cost distributions, and may be used to specify the subset of methods that are potentially optimal under any cost and class distribution assumptions.

On the ROC convex hull graph (see figure 3), the TP rate is plotted on the Y-axis and the FP rate on the X-axis. One point in the ROC graph (representing one classifier with given parameters) is better than another if it is to the northwest (TP is higher, FP is lower, or both) of the graph.

One can observe different CART, C4.5 and C4.5 GSV models with increasing false negative costs (from cost 1 to 9). This means that for each decision tree technique different models have been induced, each time increasing the penalisation of the FN errors which in turn results in higher TP and FP rates.

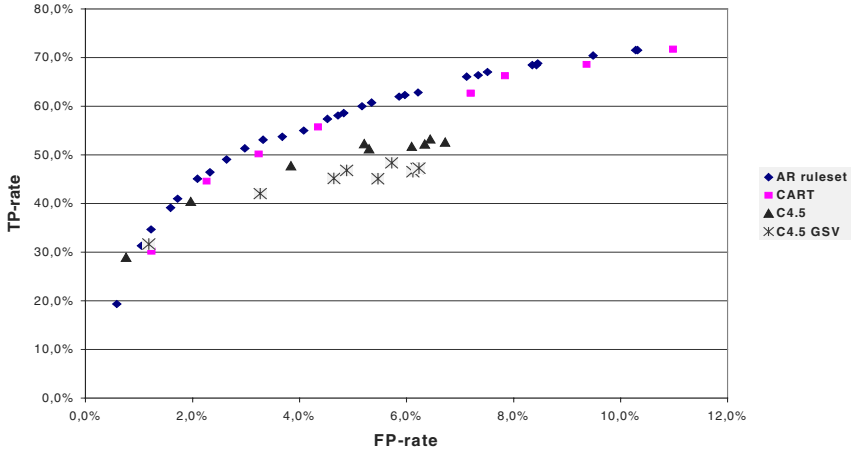


Fig. 3. ROC convex hull graph: performance of different classifiers on separate testing data from 1997

Figure 3 also shows the performance of the different association rule rulesets (AR ruleset). The number of rules in the ruleset determines the performance of the model, i.e. the AR ruleset in the bottom left corner of the graph contains only the single most interesting rule that was generated in section 3.3.1, whereas the top right AR ruleset model contains the 29 most interesting rules as determined by the interestingness measure presented earlier in this paper. The latter ruleset covers more overall dissatisfied instances (higher TP-rate) but because of adding less and less ‘interesting’ rules the FP-rate will increase as well. Decision tree models with higher FN error costs and AR rulesets containing more rules are not plotted on figure 5 since they did not increase the TP-rate significantly but only further increased the FP-rate.

According to [10], only when one classifier dominates another over the entire performance space, it can be declared better. From figure 3, it can be observed that this is the case with the collection of AR ruleset classifiers since for each FP-rate (i.e. for each group size of latently dissatisfied customers), the TP-rate of the AR ruleset is higher than for any of the other types of classifiers (CART, C4.5 and C4.5 GSV) considered in this study. Moreover, for this study, CART is clearly superior to C4.5 with (see C4.5 GSV) and without grouping of symbolic values (see C4.5).

4 External Validation

External validation on holdout data showed that association rulesets are able to capture the idea of latent dissatisfaction since it was discovered that, in the line with marketing theory [6, 7, 9], complaints behaviour and the rate of defection was significantly higher for latently dissatisfied customers than for manifestly satisfied customers. Unfortunately, due to space limitations, we cannot elaborate on this. However, the authors can be contacted for additional information on this issue.

5 Conclusion

In this study, we have compared two different methodological approaches to the identification of latently dissatisfied customers, i.e. complete versus partial classification. More specifically, C4.5, CART and association rule rulesets were evaluated and we compared their performance on a ROC convex hull graph. The reason is that the ROC convex hull graph enables comparison of different types of classification techniques under different misclassification costs and class distributions. We found confirmation for the fact that partial classification would be more appropriate when the data are characterised by very skewed class distributions. Furthermore, external validation results indicated that latently dissatisfied customers put more complaints than manifestly satisfied customers and also have a higher tendency to defect.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast Discovery of Association Rules. In: *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, chapter 12, (1996) 307-328.
2. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington D.C., (1993) 207-216.
3. Ali, K., Manganaris, S., Srikant, R.: Partial Classification using Association Rules. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, The AAAI Press, (1997) 115-118.
4. Anand, S.S., Hughes, J.G., Bell, D.A., Patrick, A.R.: Tackling the Cross-sales Problem using Data Mining. In: *Proceedings of the first Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (1997) 331-343.
5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*, The Wadsworth Statistics/Probability Series, California (1984).
6. Day, R.L.: Modelling choices among alternative responses to dissatisfaction. In: *Advances in Consumer Research* 11, (1984) 496-499.
7. Fornell, C., Wernerfelt, B.: Defensive Marketing strategy by complaint management: a theoretical analysis. In: *Journal of Marketing Research* 24, (1987) 337-346.
8. Greene, W.H.: *Econometric Analysis*. Prentice Hall, chapter 19, (1999).
9. Heskett, J.L., Sasser, W.E., Schlesinger, L.A.: The service profit chain: How leading companies link profit and growth to loyalty, satisfaction and value. In: *The Free Press*, New York, (1997).
10. Provost, F., Fawcett, T.: Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, California (USA), August 14-17, (1997) 43-46.
11. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).

Wrapper Generation via Grammar Induction

Boris Chidlovskii¹, Jon Ragetli^{2*}, and Maarten de Rijke^{2**}

¹ Xerox Research Centre Europe
6, Chemin de Maupertuis, 38240 Meylan, France
`chidlovskii@xrce.xerox.com`

² ILLC, University of Amsterdam
Pl. Muidergracht 24, 1018 TV Amsterdam, The Netherlands
`{ragetli,mdr}@wins.uva.nl`

Abstract. To facilitate effective search on the World Wide Web, meta search engines have been developed which do not search the Web themselves, but use available search engines to find the required information. By means of wrappers, meta search engines retrieve information from the pages returned by search engines. We present an approach to automatically create such wrappers by means of an incremental grammar induction algorithm. The algorithm uses an adaptation of the string edit distance. Our method performs well; it is quick, can be used for several types of result pages and requires a minimal amount of user interaction.

Keywords: inductive learning, information retrieval and learning, web navigation and mining, grammatical inference, wrapper generation, meta search engines.

1 Introduction

As the amount of information available on the World Wide Web continues to grow, conventional search engines expose limitations when assisting users in searching information. To overcome these limitations, mediators and meta search engines (MSEs) have been developed [2,6,7]. Instead of searching the Web themselves, MSEs exploit existing search engines to retrieve information. This relieves the user from having to contact those search engines manually. Furthermore, the user formulates queries using the query language of the MSE — knowing the native query languages of the connected search engines is not necessary. The MSE combines the results of the connected search engines and presents them in a uniform way.

MSEs are connected to search engines by means of so-called *wrappers*: programs that take care of the source-specific aspects of an MSE. For every search engine connected to the MSE, there is a wrapper which translates a user's query into the native query language and format of the search engine. The wrapper also takes care of extracting the relevant information from the results returned

* Supported by the Logic and Language Links project funded by Elsevier Science.

** Supported by the Spinoza project 'Logic in Action.'

Search results for query: wrapper

Number One Wrapper Generator

Description: Welcome to the wrapper generating organisation.
 1000; <http://www.wrapper.org/>

Buy our candy bar wrapper collection

Description: An advantageous offer for every candy addict.
 774; <http://www.candy.com/wrappers/>

Maestro's Candy Bar Wrapper Collection

Description: Yes, I devote my otherwise useless life to collecting wrappers.
 312; <http://www.freehomepages.com/~maestro/>

Fig. 1. Sample result page

by the search engine. We will refer to the latter as ‘wrapper’ and do not discuss the query translation (see [5] for a good overview). An HTML result page from a search engine contains zero or more ‘answer items’, where an answer item is a group of coherent information making up one answer to the query. A wrapper returns each answer item as a tuple consisting of attribute/value pairs. For example, from the result page in Fig. 1 three tuples can be extracted, the first of which is displayed in Fig. 2. A wrapper discards irrelevant information such as layout instructions and advertisements; it extracts information relevant to the user query from the textual content and attributes of certain tags (e.g., the `href` attribute of the `<A>` tag).

Manually programming wrappers is a cumbersome and tedious task [4], and since the presentation of the search results of search engines changes often, it has to be done frequently. To address this, there have been various attempts to automate this task [3,9,10,12,13]. Our approach is based on a simple incremental grammar induction algorithm. As input, our algorithm requires one result page of a search engine, in which the first answer item is labeled: the start and end of the answer need to be indicated, as well as the attributes to be extracted. After this, the incremental learning of the *item grammar* starts, and with an adapted version of the *edit distance* measure further answer items on the page are found and updates to the extraction grammar are carried out. Once all items have been found and the grammar has been adapted accordingly, some post-processing takes place, and the algorithm returns a wrapper for the entire

```
{ url = "http://www.wrapper.org", ~title = "Number One Wrapper
Generator", ~description = "Welcome to the wrapper generating
organization", ~relevance = "1000" }
```

Fig. 2. An item extracted

page. The key features of our approach are limited user interaction (labeling only one answer item) and good performance: for a lot of search engines it generates working wrappers, and it does so very quickly.

The paper is organized as follows. In the next section we show how to use grammar induction for the construction of wrappers. After that we describe our wrapper learning algorithm. We then present experimental results, comparisons and conclusions. Full details can be found in [14].

2 Using Grammar Induction

We view *labeled* HTML files as strings over the alphabet $\Sigma \cup \mathcal{A}$, where every $\sigma \in \Sigma$ denotes an HTML tag, and every $a_i \in \mathcal{A}$ ($i = 1, 2, \dots$) denotes an attribute to be extracted. The symbol a_0 in $\mathcal{A}$ represents the special attribute `void`, that should *not* be extracted; Σ and $\mathcal{A}$ are disjoint. For example, the HTML fragment

```
<title>Wrapper Induction</title>
```

might correspond to the string $ta_1\bar{t}$, where t and $\bar{t}$ are symbols of Σ which denote tags `<title>` and `</title>`, respectively. The text `Wrapper Induction` has to be extracted as the value of attribute $a_1 \in \mathcal{A}$.¹

We aim to construct a wrapper that is able to extract all relevant information from a given labeled page and unseen pages from the same source. We solve the problem by decomposing it into two simpler subtasks. The first one is to find an expression that locates the beginning (**Start**) and the end (**End**) of the list of answer items. The second subtask is to induce a grammar **Item** that can extract all the relevant information from every single item on the page. The grammar describing the entire page will then be of the form **Start (Item)* End**. The **Start** and **End** expressions can easily be found. The grammar induction takes place when the grammar for the items is generated. Here, the item grammar is learned from a number of samples from $(\Sigma \cup \mathcal{A})^+$, corresponding to the answer items on the page. Besides learning the grammar, our algorithm also *finds* the samples on the HTML page that it uses to learn the grammar.

Preprocessing the HTML Page. All known approaches for automatically generating wrappers require as input one or more labeled HTML pages: all or some of the attributes to be extracted have to be marked by the user or some labeling program. As it is hard to create labeling programs for the heterogeneous set of search engines that an MSE must be connected to, and the labeling is a boring and time-consuming job, we have restricted the labeling for our algorithm to a single answer item only. Figure 3 shows the labeled source for the HTML page in Fig. 1. The labeling consists of an indication of the begin (`^BEGIN^`) and end (`^END^`) of the first answer item, the names of the attributes (e.g. `^URL^`), and the end of the attributes (`^^`). After the item has been labeled, it is *abstracted* by our algorithm to turn it into a string over $\Sigma \cup \mathcal{A}$.

¹ This representation is somewhat simplified. The program can also extract tag attributes, such as the `href` attribute for the `A` tag, or split element contents with conventional string separators. Due to space limitations, we omit details.


```

<HTML><HEAD><TITLE>Search results for query: wrapper</TITLE></HEAD>
<BODY bgcolor = "white" text= "black">
<H3>Search results for query: wrapper</H3>
  <dl>
    ^BEGIN^ <dt> ^URL^ <a href="http://www.wrapper.org/"> ^^
    ^TITLE^ Number One Wrapper Generator ^^ </a><br>
    <dd><i>Description:</i> ^DESCR^ Welcome to the wrapper
    generating organisation. ^^ <br>
    <font size="-3"><I> ^REL^ 1000 ^^ </I>;
    http://www.wrapper.org/</font> ^END^
  </dl>
  <dl>
    <dt><a href="http://www.candy.com/wrappers/">
    Buy our candy bar wrapper collection </a><br>
    :
    <font size="-3"><I>312</I>;
    http://www.freehomepages.com/~maestro/</font>
  </dl>
</BODY></HTML>

```

Fig. 3. Labeled HTML source of result page

The Item Grammar. The item grammar has to be learned from merely *positive* examples; this cannot be done efficiently for regular expressions with the full expressive power of Finite State Automata (FSAs) [15]. We aim to learn a very restricted kind of grammar, which we will first describe as a simple form of FSA, called sFSA, where transitions labeled with an attribute $a_i \in \mathcal{A}$ (except a_0) also produce output: the attribute name and the token consumed. After that we show how those sFSAs correspond with a simple form of regular expression. We start by defining an extremely simple class of FSAs.

Definition 1 (Linear FSA). A sequence of nodes $n_1 \dots n_m$, where every node n_i ($1 \leq i < m$) is connected to n_{i+1} by one edge $e_{i,i+1}$ labeled with elements from $\Sigma \cup \mathcal{A}$, is a *linear FSA* if it is the case that whenever $e_{i,i+1}$ is labeled with an element $a \in \mathcal{A}$, then $e_{i-1,i}$ and $e_{i+1,i+2}$ are labeled with elements from Σ .

The fact that the attribute a in Definition 1 is surrounded by HTML tags (from Σ) allows us to extract the attribute. Fig. 4 shows a linear FSA that can only extract the attributes from *one* type of item: an item that has an attribute

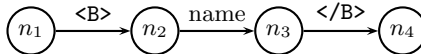


Fig. 4. A linear FSA

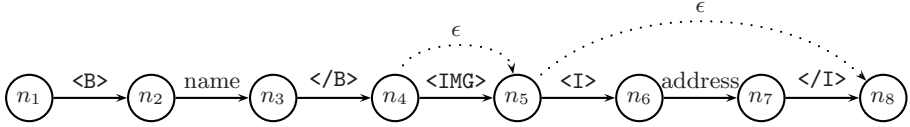


Fig. 5. An sFSA

name between $\langle B \rangle$ and $\langle /B \rangle$ tags (symbols in Σ , like $\langle B \rangle$ and $\langle /B \rangle$ in Fig. 4, represent tokens for *abstracted tags*). Therefore, it is not very useful. The sFSAs that we employ to learn the structure of items, are a bit more complex.

Definition 2 (simple FSA). A linear FSA that also has ϵ -transitions $s_{i,j}$ (transitions labeled with ϵ) from node n_i to node n_j ($i < j$) is called a *simple FSA (sFSA)* if

- whenever there is an ϵ -transition $s_{i,j}$ there is no ϵ -transition $s_{k,l}$ with $i \leq k \leq j$, or $i \leq l \leq j$, and
- whenever there is an ϵ -transition $s_{i,j}$, and $e_{j,j+1}$ is labeled with an element from $\mathcal{A}$, $e_{i-1,i}$ is labeled with an element from Σ .

The first condition demands that ϵ -transitions do not overlap or subsume each other. The second condition states that when an ϵ -transition ends at a node with an outgoing edge with a label from $\mathcal{A}$ (i.e., the abstracted content), it has to start at a node with an incoming edge with a label from Σ (i.e., an abstracted HTML tag). The latter guarantees that an attribute is always surrounded by HTML tags, no matter what path is followed through the automaton.

Figure 5 shows an sFSA that can extract names and addresses from items, where some items do not contain the address between $\langle I \rangle$ and $\langle /I \rangle$, and there may be an image ($\langle \text{IMG} \rangle$) after the name that is enclosed by $\langle B \rangle$ and $\langle /B \rangle$ tags. The ϵ -transitions of the sFSA make it more expressive than a linear FSA, but sFSAs are less expressive than FSAs, since sFSAs do not contain cyclic patterns.

Where do grammars come in? One can represent the language defined by an sFSA by a simple kind of regular expression with fixed and optional parts. Using brackets to indicate optional parts, the sFSA of Fig. 5 can be represented as $\langle B \rangle \text{name} \langle /B \rangle [\langle \text{IMG} \rangle] [\langle I \rangle \text{address} \langle /I \rangle]$. This expression acts as a grammar defining the same sequences of abstracted tags and content as the sFSA. We refer to this representation as *item grammar* or simply *grammar*. The grammar can be this simple, because the HTML pages for which they are created are created dynamically upon user requests and therefore have a regular structure.

3 Inducing the Item Grammar

Our grammar induction algorithm is incremental; item grammar G_n , based on the first n items, is adapted on encountering item $n + 1$, resulting in grammar G_{n+1} . The update of the grammar is based on an algorithm calculating the *string edit distance* [1].

item grammar	$a \ b \ - \ d$	item grammar	$a \ b \ [c] \ d$	item grammar	$a \ b \ - \ d$
string	$a \ b \ c \ d$	string	$a \ b \ c \ -$	string	$a \ - \ c \ d$
new item gr.	$a \ b \ [c] \ d$	new item gr.	$a \ b \ [c] \ [d]$	new item gr.	$a \ [b] \ [c] \ [b] \ d$
(a)		(b)		(c)	

Fig. 6. Three alignments

Definition 3 (Edit distance). The *edit distance* $D(s_1, s_2)$ between two strings of symbols s_1 and s_2 is the minimal number of insertions or deletions of symbols, needed to transform s_1 into s_2 .

For example, $D(abcd, abide) = 3$: to transform $abcd$ into $abide$ at least three insertion or deletion operations have to be performed. Here, and in the examples below, the characters are symbols from $\Sigma \cup \mathcal{A}$. The algorithm that we use to calculate the edit distance also returns a so-called *alignment*, indicating the differences between the strings. For $abcd$ and $abide$ the alignment is the following:

$$\begin{array}{c} a \ b \ c \ - \ d \ - \\ a \ b \ - \ i \ d \ e \end{array}$$

The dashes indicate the insertion and deletion operations; see [1,14] for more details. We have adapted the edit distance algorithm in a way that permits to calculate the distance between an item grammar — a string of symbols with optional parts — and an item. The adaptation amounts to first simplifying the item grammar by removing all brackets, while remembering their position. Now the edit distance between the item and the simplified grammar can be calculated as usual. Using the alignment and the remembered position of the brackets, the new grammar is calculated. We have also adapted the edit distance algorithm to deal with labeled attributes in the grammar, that correspond with unlabeled content in the item; we omit details here.

The algorithm detects and processes different cases in the alignment between G_n and the $n + 1$ -th item. Since the full algorithm description is extensive and space is limited, we can only indicate how it works with the some examples. As the item grammar in Fig. 6 (a) does not contain c , whereas the string to be covered does, the resulting item grammar has an optional c in it, so that it covers both abd and $abcd$. Now suppose the string abc has to be covered by the new item grammar (Fig. 6 (b)). The reason for making d optional is that it does not occur in the new string. The new item grammar covers ab , abc , abd and $abcd$, which is a larger generalization than simply ‘remembering’ the examples. In Fig. 6 (c), the new item grammar $a[b][c][b]d$ is a large generalization; besides abd and acd it covers ad , $abcd$, $acbd$ and $abcbd$, i.e., five other strings besides the original item grammar and the example. The reason we decided to have a large generalization is that based on the examples we can at least conclude that b and c are optional, but they may co-occur in any order.

```

1.  $D_{newlocal} := 998, D_{local} := 999, D_{best} := 1000$ 
2.  $i_b, i_e := 0$ 
3.  $local\_best\_item := \emptyset, best\_item := \emptyset$ 
WHILE  $D_{local} < D_{best}$  and not at end of page
4.  $D_{best} := D_{local}$ 
5.  $best\_item := local\_best\_item$ 
6.  $i_b := \text{next occurrence begin tag}(s)$ 
  WHILE  $D_{newlocal} < D_{local}$ 
7.  $D_{local} := D_{newlocal}$ 
8.  $local\_best\_item := (i_b, i_e)$ 
9.  $i_e := \text{next occurrence end tag}(s)$ 
10.  $D_{newlocal} := D(\text{item grammar}, (i_b, i_e))$ 
11. IF  $D_{best} > \text{Threshold}$  THEN  $best\_item := \emptyset$ 
12. return  $best\_item$  and  $D_{best}$ 

```

- $D_{newlocal}$ stores the distance of the item grammar to the part of the page between the latest found occurrence of the begin and end tag(s)
- D_{local} stores the distance of the item grammar to **local-best-item**
- D_{best} stores the distance of the item grammar to **best-item**
- i_b, i_e are the indexes of the begin and end of a (potential) item
- **local-best-item** stores the potential item starting at i_b that has the lowest distance to the item grammar of the potential items starting at i_b
- **best-item** stores the potential item that has the smallest distance to the item grammar so far

Fig. 7. The Local Optimum Method

4 Finding Answer Items

So far, we have discussed the learning of the grammar based on the answer items on the HTML page. As only the first answer item on the page has been indicated by its labeling, the other items have to be found. For this, we use the distance calculated by the adapted edit distance algorithm. We have implemented three different strategies for finding the answer items on the page, but as space is limited we will only describe the best and most general one: the *Local Optimum Method* (LOM). The other two are simpler and usually quicker, but even with the LOM a wrapper is quickly generated; see Section 6.

Our methods for finding items are based on an important assumption: *all items on the page have the same begin and end tag(s)*. As a consequence we can view the task of finding items on a page as finding substrings on the page below the labeled item that start and end with the same delimiters as the first labeled item. The user can decide for how many tags this assumption holds by setting the parameter *SeparatorLength*. If more begin or end tags are used, it will be easier to find the items on the page; there is less chance of finding for example a sequence of two tags than only one tag. However, setting the parameter too high will result in too simple a grammar without any variation.

The LOM tries to find items on the page that are *local*, i.e., below and close to the item that was found last, and *optimal* in the sense that their distance to the item grammar is low. Figure 7 shows the algorithm. In the first three steps, a number of variables are initialized. As to the outermost *while* loop, once the previous item has been found, or the first labeled item, the LOM looks for the next occurrence of the begin delimiter, and then it looks for the first occurrence of the end delimiter. Material between those delimiters is a potential item; this is checked by calculating its edit distance to the item grammar. Below the last found end delimiter, the LOM looks at the next occurrence of the end delimiter. This is a new potential item to consider, so the distance between the item grammar and this potential item is measured. If this distance is lower than the previous distance, another occurrence of the end delimiter is considered. If not, the previous potential item is stored as the local-best-item, and potential items a bit lower on the page are considered next. The process of considering new end delimiters starts again, resulting in a new local-best-item. Now the two local-best-items are compared. If the second one was better than the first one, LOM will seek the next occurrence of the begin delimiter. If not, the previous local-best-item is returned as the local-optimal item.

In step 11 of the algorithm, a *Threshold* is mentioned. If the distance of the best candidate item exceeds *Threshold*, the algorithm will return $\emptyset$ instead of this item; this prevents the grammar to be adjusted to cover the item, and the process of finding the item stops. *Threshold* is the product of two values: *HighDistance* and *Variation*. *HighDistance* is the maximum distance of any item incorporated so far. Its initial value is set by the user, and it is incremented whenever an item is incorporated whose distance is higher than *HighDistance*; it can be used to compensate for the simplicity of the distance measure. *Variation* is a value that is not adapted during the process of finding the items.

5 The Entire Wrapper Generating Algorithm

We have discussed the two most important components of our wrapper generator: learning the grammar, and finding the items. In Fig. 8 the entire wrapper generating algorithm is described; below we discuss some components.

The first step, **abstract**, abstracts LP , the page labeled by the user, into a sequence of symbols $AP \in (\Sigma \cup \mathcal{A})^+$; see Section 2. The second step initializes the grammar G to the first, labeled item. In the third step, **find-next-item** is the algorithm for finding items, as described in Section 4; in the fourth step **incorporate-item** adjusts the grammar in the way we described in Section 3. In the fifth step, the grammar G is used to make a grammar for the whole page. The user might have labeled the first item smaller than it actually is. By the assumption that all items on the page have the same begin and end tags, the found items (and the resulting grammar) will also be too small. Therefore, the item grammar will be extended if possible. If there is a common suffix of the HTML between the items covered and the HTML before the first item, this suffix is appended to the beginning of the item grammar. If there is a common prefix

```

1.  $AP := \text{abstract}(LP)$ 
2.  $G := \text{initialize}(AP)$ 
REPEAT
  3.  $I := \text{find-next-item}(AP, G)$ 
  4. IF  $I \neq \emptyset$  THEN  $G := \text{incorporate-item}(G, I)$ 
UNTIL  $I = \emptyset$ 
5.  $GP := \text{expression-whole-page}(G, AP)$ 
6.  $W := \text{translate-to-wrapper}(GP)$ 
7. return  $W$ 

```

- LP is the labeled HTML page
- AP is the abstracted page
- G is the item grammar
- I is an item
- GP is a grammar for the entire page
- W is the same grammar, translated into a working wrapper

Fig. 8. The wrapper generating algorithm

of the HTML between the items, and the HTML below the last found item, it is appended to the end of the item grammar. Besides this, expressions for **Start** and **End**, as discussed in Section 2, are also generated in this fifth step. This is easy: the expression for **Start** is the smallest fragment of AP just before the labeled item that does not occur before in AP . **End** is recognized implicitly, by the fact that no items can be recognized anymore.

For skipping the useless HTML in the item list, another grammar is constructed — the *Trash* grammar. It does not contain attributes to be extracted, so the trash grammar will consist of symbols in $\Sigma \cup \{a_0\}$. The indices of the items found have been stored, so this process is a repetition of **incorporate-item**. Once the trash grammar has been constructed, it is appended to the end of the item grammar. Once the item and trash grammars have been generated, our algorithm will detect repetitions, and it will generalize the grammars accordingly.

After all these processing steps, we have an abstract wrapper of the form $\text{Start (Item Trash)}^+$, that is an expression for the beginning of the item list, followed by one or more repetitions of a sequence of the item grammar and the trash grammar. The last step of the algorithm in Fig. 8 is the conversion of the abstract grammar into a working wrapper. In our implementation we translate the abstract grammar into a JavaCC parser [11], as the meta searcher Knowledge Brokers, developed at Xerox Research Centre Europe, is programmed in Java.

6 Experimental Results

We have tested our wrapper generating algorithm on 22 different search engines. This is a random selection of sources to which Knowledge Brokers had already been connected manually. It was quite successful, as it created working wrappers

Table 1. Experimental results

Successfully generated wrappers				
source	URL	size (kB)	NI	time (sec)
ACM	www.acm.org/search	12	10	8.0
Elsevier Science	www.elsevier.nl/homepage/search.htt	11	11	2.6
NCSTRL	www.ncstrl.org	9	8	32.5
IBM Patent Search	www.patents.ibm.com/boolquery.html	19	50	5.3
IEEE	computer.org/search.htm	26	20	3.7
COS U.S. Patents	patents.cos.com	17	25	5.4
Springer Science Online	www.springer-ny.com/search.html	36	100	32.1
British Library Online	www.bl.uk	5	10	2.6
LeMonde Diplomatie	www.monde-diplomatique.fr/md/index.html	6	4	2.5
IMF	www.imf.org/external/search/search.html	10	50	5.3
Calliope	sSs.imag.fr*	22	71	4.1
UseNix Association	www.usenix.org/Excite/AT-usenixquery.html	16	20	4.3
Microsoft	www.microsoft.com/search	26	10	4.5
BusinessWeek	bwarchive.businessweek.com	13	20	3.9
Sun	www.sun.com	20	10	3.7
AltaVista	www.altavista.com	19	10	4.1
Sources for which the algorithm failed to generate a wrapper				
source	URL			
Excite	www.excite.com			
CS Bibliography (Trier)	www.informatik.uni-trier.de/~ley/db/index.html			
Library of Congress	lcweb.loc.gov			
FtpSearch	shin.belnet.be:8000/ftpsearch			
CS Bibliography (Karlsruhe)	liinwww.ira.uka.de/bibliography/index.html			
IICM	www.iicm.edu			

* Only accessible to members of the Calliope library group.

for 16 of the 22 sources. For 2 other sources the generated incorrect wrappers could easily be corrected. The working wrappers were created with only one answer item labeled. This means that good generalizations are being made when inducing the grammar for the items; labeling only *one* item of *one* page is sufficient to create wrappers for many other items and pages. Table 1 summarizes our experimental results; the fourth column, labeled NI, contains the total number of items on the page. The times displayed in Table 1 were measured on a modest computer (PC AMD 200MMX/32 RAM). Still, the time to generate a wrapper is very short; it took at most 32.5 seconds, with the average time being 7.8 seconds. Together with the small amount of labeling that has to be done, this makes our approach to generating wrappers a very rapid one.

Increasing the *SeparatorLength* value (see Section 4) makes our algorithm faster, as fewer fragments of HTML are taken into account. For NCSTRL, the time to generate a wrapper is shown with a *SeparatorLength* of 1 (32.5 seconds), as 1 is the default *SeparatorLength*. However, with a *SeparatorLength* of 2, it takes 22.5 seconds, with 3 it takes 21.4 seconds, and with 4 17.1 seconds.

Robustness of the Wrappers. An important aspect of the generated wrappers is the extent to which the result pages of the search services may change without the wrapper breaking down. The wrappers we generate are not very robust. Little is allowed to change in the list with search results, because the wrapper for that list is generated so as to closely resemble the original HTML code. But even if

the wrappers are not very robust, it is easy to create a new wrapper whenever the search engine’s result pages change. Our algorithm is fast and does not need much interaction, which makes it unproblematic to generate a new wrapper.

Incorrect Wrappers. There are various reasons why our algorithm failed to produce working wrappers for the six sources mentioned in Table 1. In some cases the HTML of the result pages was incorrect (Excite, IICM). In another case attributes were only separated by textual separators and not by HTML tags, making it impossible to create a wrapper for it with our algorithm (Library of Congress). In some cases the algorithm failed to create a working wrapper because the right items were not found due to too much variation in the items (Computer Science Bibliography Trier, FtpSearch). And in another case there was too much variation in the items and in the hierarchical way in which they were presented (Computer Science Bibliography Karlsruhe).

Grammar Evaluation. How do we determine that an induced grammar is correct? Like in all other approaches, the grammar induction is called *successful* if the grammar extracts correctly all items from the example page. For certain sources, one result page was insufficient and more pages were needed to learn all structural variations and induce the working wrappers. However, in all these cases, once the grammar was successfully induced for the initial, labeled page, it was always possible to extend it to new result pages, without additional labeling.

In the general case, the Probably Approximate Correct (PAC) technique is used to estimate the grammar accuracy; however, since our method is really fast at incorporating new structural variations, we found that it is easier to keep incorporating forever; we omit details here.

Comparison to Other Approaches. Most alternative approaches differ from ours in significant ways. Some are far simpler [8], or specify wrappers manually at a far more abstract level [6]. Others differ in that they are based on static templates instead on learning the structure of result pages [12]. Still others are based on assumptions about the structure and lay-out cues [3,16]. Some approaches need much more user interaction as the user has to label several entire pages [13]. The approach of Hsu, Chang and Dung [9,10] is the one that is most similar to ours. Their finite-state transducers, called *single-pass SoftMealy extractors*, resemble the grammars that we generate, although they abstract pages in a more fine-grained way. In their approach, textual content is further divided, e.g., in numeric strings and punctuation symbols. The approach seems to create more robust wrappers than ours, but at the price of more extensive user input. Further comparisons — empirical and analytic — of the approaches are needed to understand the trade-off between user interaction and quality of the wrappers.

7 Conclusion and Further Work

We have presented an approach to automatically generate wrappers. Our method uses grammar induction based on an adapted form of the *edit distance*. Our

wrapper generator is language independent, because it relies on the structure of the HTML code to build the wrappers. Experimental results show that our approach is accurate — 73% (allowing minor modifications: 82%) of the wrappers generated are correct. Our generator is quick, as it takes less than 10 seconds to generate a wrapper for most sources. The most important advantage of our approach is that it requires minimal user input; it suffices to label only one item on the page for which the wrapper has to be generated; the other items are found by the wrapper generator itself.

Although our wrapper generator works well, it can be extended and improved in several ways. For a start, it would be useful if the user could label attributes in a graphical interface that hides the HTML code. Second, we need to extend the wrapper to generate code to handle no result pages. Also, we would like to experiment with relaxing our assumption that all attributes are separated by HTML tags. Further, if a lot of search engines for a specific domain have to be connected to a meta searcher, it may be worthwhile to create *recognizers* [12], modules that find and label the attributes on the page. Finally, we have deliberately investigated the power of our method with minimal user input, but conjecture that labeling more answer items and selecting them carefully improves performance.

References

1. Aho, Alfred V. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 255–300, Elsevier, 1990. 100, 101
2. Andreoli, J.-M., Borghoff, U., Chevalier, P.-Y., Chidlovskii, B., Pareschi, R., and Willamowski, J. The Constraint-Based Knowledge Broker System. *Proc. of the 13th Int'l Conf. on Data Engineering*, 1997. 96
3. Ashish, N., and Knoblock, C. Wrapper Generation for Semi-structured Internet Sources. *SIGMOD Record* 26(4):8–15, 1997. 97, 106
4. Chidlovskii, B., Borghoff, U., Chevalier, P.-Y. Chevalier. Toward Sophisticated Wrapping of Web-based Information Repositories. *Proc. 5th RIAO Conference, Montreal, Canada*, pages 123–135, 1997. 97
5. Florescu, D., Levy, A., and Mendelzon, A. Database techniques for the World-Wide Web: A Survey. *SIGMOD Record* 27(3):59–74, 1998. 97
6. Garcia-Molina, H., Hammer, J., and Ireland, K. Accessing Heterogeneous Information Sources in TSIMMIS. *AAAI Symp. Inform. Gathering*, pages 61–64, 1995. 96, 106
7. Gauch, S., Wang, G., Gomez, M. ProFusion: Intelligent Fusion from Multiple Distributed Search Engines. *J. Universal Computer Science*, 2(9): 637–649, 1996. 96
8. Hammer, J. Garcia-Molina, H., Cho, J., Aranha, R., and Crespo, A. Extracting Semistructured Information from the Web. *Proceedings of the Workshop on Management of Semistructured Data*, 1997. 106
9. Hsu, C.-N., and Chang, C.-C. Finite-State Transducers for Semi-Structured Text Mining. *Proc. IJCAI-99 Workshop on Text Mining*, 1999. 97, 106
10. Hsu, C.-N., and Dung, M.-T., Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8):521–538, 1998. 97, 106

11. JavaCC – The Java parser generator. URL: <http://www.metamata.com/JavaCC/>. 104
12. Kushmerick, N., Weld, D.S., and Doorenbos, R., Wrapper Induction for Information Extraction. *Proc. IJCAI-97*: 729–737, 1997. 97, 106, 107
13. Muslea, I., Minton, S., Knoblock, C. STALKER. *AAAI Workshop on AI & Information Integration*, 1998. 97, 106
14. Ragetli, H.J.N. Semi-automatic Parser Generation for Information Extraction from the WWW. *Master's Thesis, Faculteit WINS, Universiteit van Amsterdam*, 1998. 98, 101
15. Sakakibara, Y. Recent advances of grammatical inference. *Theoretical Computer Science* 185:15–45, 1997. 99
16. Soderland, S. Learning to Extract Text-based Information from the World Wide Web. *Proc. KDD-97*, pages 251–254, 1997. 106

Diversity versus Quality in Classification Ensembles Based on Feature Selection

Pádraig Cunningham and John Carney

Department of Computer Science
Trinity College Dublin, Ireland.
Padraig.Cunningham@tcd.ie

Abstract. Feature subset-selection has emerged as a useful technique for creating diversity in ensembles – particularly in classification ensembles. In this paper we argue that this diversity needs to be monitored in the creation of the ensemble. We propose an entropy measure of the outputs of the ensemble members as a useful measure of the ensemble diversity. Further, we show that using the associated conditional entropy as a loss function (error measure) works well and the entropy in the ensemble predicts well the reduction in error due to the ensemble. These measures are evaluated on a medical prediction problem and are shown to predict the performance of the ensemble well. We also show that the entropy measure of diversity has the added advantage that it seems to model the change in diversity with the size of the ensemble.

1. Introduction

Feature subset selection is an important issue in Machine Learning [1][2][12]. It is a difficult problem due to the potentially huge search space involved and because hill-climbing search techniques do not work so well because of an abundance of local maxima in the search space. Effective feature selection is important for the following reasons:

- **Build better predictors:** better quality predictors/classifiers can be built by removing irrelevant features – this is particularly true for lazy learning systems.
- **Economy of representation:** allow problems/phenomena to be represented as succinctly as possible using the features considered relevant.
- **Knowledge discovery:** discover what features are and are not influential in *weak theory* domains.

Another motivation for feature subset selection has emerged in recent years as illustrated in the research of Ho [6][7] and Guerra-Salcedo and Whitley [4][5]. In their work feature subset selection is used as a mechanism for introducing diversity in ensembles of classifiers. Typically they work with datasets from weak theory domains where features have been oversupplied and there are irrelevant and redundant features in the representation.

In this paper we look at this approach to ensemble creation and propose entropy and cross entropy as measures of diversity and error that should be used in assessing groups of classifiers for forming an ensemble.

The paper starts with a review in section 2 of some existing research on ensembles of classifiers based on different feature subsets. We argue that diversity in the ensemble must be considered explicitly in putting together the ensemble. In section 3 we review the approach to diversity in regression ensembles where variance is the standard measure of diversity. In section 4 we present our algorithm for producing good quality feature subsets and in section 5 we show how the entropy measure of diversity provides a valuable insight into the operation of ensembles of classifiers in a medical application and helps determine the makeup of a very good ensemble.

2. Existing Research

Ho [7] introduces the idea of ensembles of Nearest Neighbour classifiers where the variety in the ensemble is generated by selecting different feature subsets for each ensemble. Since she generates these feature subsets randomly she refers to these different subsets as random subspaces. She points to the ability of ensembles of decision trees based on different feature subsets to improve on the accuracy of individual decision trees [6]. She advocates doing this also for k -Nearest Neighbour (k -NN) classifiers because of the simplicity and accuracy of the k -NN approach. She shows that an ensemble of k -NN classifiers based on random subsets improves on the accuracies of individual classifiers on a hand-written character recognition problem.

Guerra-Salcedo and Whitley [4][5] have improved on Ho's approach by putting some effort into improving the quality of the ensemble members. They use a genetic algorithm (GA) based search process to produce the ensemble members and they show that this almost always improves on ensembles based on the random subspace process. The feature masks (subsets) that define each ensemble member are the product of GA search and should have higher accuracy than masks produced at random. The random masks performed better than the masks produced by genetic search only on datasets with small numbers of features (19 features)[5].

Guerra-Salcedo and Whitley do not suggest any reasons why the random subspace method should outperform the genetic search method on data sets with small numbers of features. We suggest that this is explained by the analysis of diversity and accuracy presented in the next section. In problems with large numbers of features (>30) diversity is not a problem whereas in problems with smaller numbers of features diversity should be monitored. This diversity/quality issue will be discussed in detail in the next section. In concluding this paper we will argue that any work on ensembles should explicitly measure diversity and quality to ensure that the overall quality of the results of the ensemble is maximised.

3. Diversity

Krogh and Vedelsby [8] have shown the following very important relationship between error and ambiguity (diversity) in regression ensembles

$$E = \overline{E} - \overline{A} \quad (1)$$

where E is the overall error of the ensemble over the input distribution, $\bar{E}$ is the average generalisation error or the ensemble components and $\bar{A}$ is the ensemble ambiguity averaged over the input distribution. $\bar{E}$ is a standard quadratic error estimation and $\bar{A}$ is an aggregation of individual ambiguities $\bar{a}(x)$, the ambiguity of a single ensemble member on a single input x :

$$\bar{a}(x) = \frac{1}{N} \sum_{n=1}^N (V^n(x) - \bar{V}(x))^2 \quad (2)$$

where $V^n(x)$ is the prediction of the n^{th} ensemble member for x and $\bar{V}(x)$ is the average prediction of the ensemble. Thus the ambiguity is effectively the variance in the predictions coming from the ensemble members. This ambiguity can be tuned, for instance by overfitting neural networks, in order to maximise generalisation performance [3].

For classifiers, the obvious estimate of accuracy (or error) is the proportion of a test set it classifies correctly.

$$e_i = P_{\hat{c}_i(x)=c(x)} \quad (3)$$

where $\hat{c}_i(x)$ is the category classifier i predicts for x and $c(x)$ is the correct category.

Then a possible measure of agreement (inversely related to ambiguity) is that used by Ho: using a test set of n fixed samples and assuming equal weights, the estimate of classifier agreement $\hat{s}_{i,j}$ can be written as:

$$\hat{s}_{i,j} = \frac{1}{n} \sum_{k=1}^n f(x_k) \quad (4)$$

where

$$f(x_k) = \begin{cases} 1 & \text{if } c_i(x_k) = c_j(x_k) \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

i.e. the measure of agreement is the proportion of test cases on which two classifiers agree [7]. Ho emphasises the importance of disagreement in ensemble members but does not directly evaluate its impact on the overall ensemble performance.

We will show in the evaluation section of this paper that a better measure of agreement (or ambiguity) for ensembles of classifiers is entropy. Tibshirani [10] also suggests that entropy is a good measure of *dispersion* in bootstrap estimation in classification. So for a test set containing M cases in a classification problem where there are K categories a measure of ambiguity is:

$$\tilde{A} = \frac{1}{M} \sum_{x=1}^M \sum_{k=1}^K -P_k^x \log(P_k^x) \quad (6)$$

where P_k^x is the frequency of the k^{th} class for sample x – the more dispersion or randomness in the predictions the more ambiguity. Associated with this entropy-based measure of diversity is a Conditional Entropy-based measure of error (loss function).

$$E_{CEnt} = \sum_{\hat{c}(x) \in K, c(x) \in K} P(\hat{c}(x), c(x)) \log P(\hat{c}(x) | c(x)) \quad (7)$$

where $\hat{c}(x)$ is predicted category for sample x and $c(x)$ is the correct category. We will show in section 5 that if this measure is used as the loss (error) function the entropy measure of ambiguity in the ensemble better predicts the reduction in error due to the ensemble.

4. Producing Ensembles of Feature Masks

For a classification task with p possible input features there are 2^p possible subsets of this feature set and each subset can be represented as a feature mask of 1s and 0s. Masks of this type representing different feature subsets can easily be produced using a random number generator. These masks should score high on diversity because there has been no attempt to learn good quality feature sets. However, because of this, they cannot be expected to have very good scores for $\bar{E}$, the average error. Ho [7] has shown that ensembles of masks of this type can produce very good results – presumably because the lack of quality in the ensemble members is compensated for by the diversity.

At the other end of the quality spectrum, Guerra-Salcedo & Whitley [4][5] have used genetic algorithms (GA) to find high quality feature subsets. Since the GA search is, in Aha & Bankert terms, a *wrapper* process it is very computationally intensive because evaluating each state in the search space involves testing a classifier on a test set [1]. If this estimate of fitness is to be accurate then significant amounts of data must be used to build the classifier and test it. For this reason we use a simpler hill-climbing search technique that produces good quality masks but in reasonable time. The classifier at the centre of the wrapper-based search is a k -Nearest Neighbour (k -NN) classifier. The idea is to focus on managing diversity rather than ensemble member quality to provide overall ensemble quality. The algorithm for this is shown in Table 1.

Typically this algorithm will terminate after four cycles through the mask. At that stage there is no adjacent mask (i.e. a mask different in just one feature) that is better. Thus the masks produced are local maxima in the search space.

5. Evaluation

In this section we will assess this relationship between ambiguity and accuracy using a k -NN classifier on some unpublished In-Vitro Fertilisation (IVF) data. The data consists of 1355 records describing IVF cycles of which 290 cycles have positive

outcomes and 1065 have negative outcomes. In the representation of the data used here each data sample has 53 numeric input features. For the purpose of our evaluation 50 random masks were produced in the manner described in section 4.1. Then 50 better quality masks were produced in the manner described in section 4.2. To guide this search process 580 data samples were used including the 290 with positive outcomes – 330 are used in T_r and 250 in T_s .

Table 1. Producing good quality feature masks using hill-climbing search

We define $Acc(T_r, T_s, L)$ as the accuracy of a classifier on test T_s having been trained with T_r and using mask L . The accuracy is the proportion of T_s that is correctly classified.

1. Initialise mask L randomly as described in section 4.1.
 2. Flag $\leftarrow$ False
 3. For each $l \in L$
 - Produce L' from L by flipping l
 - If $Acc(T_r, T_s, L') > Acc(T_r, T_s, L)$
 - $L \leftarrow L'$
 - Flag $\leftarrow$ True
 6. Repeat from 2 while Flag = True.
-

This search process for producing the masks is very computationally expensive with the cost increasing with the square of the size of the data set used to guide the search. However if we skimp on the amount of data used, the masks will be biased towards the subset of data that does actually get used. Indeed it was clear during the course of the evaluation that the masks did overfit the training data, raising the question of overfitting in feature selection – a neglected research issue.

Then ensembles of size 5, 10, 20, 30, 40 and 50 were produced for the random masks and the better quality masks. These ensembles were tested using leave-one-out testing on the complete data set of 1355 samples. This means that the masks are being tested, in part, with the data used to produce them. This was done because of the small number of positive samples available and is reasonable because the objective is to show the ambiguity/accuracy relationship rather than produce a good estimate of generalisation error. Where possible, multiple different versions of the smaller ensembles were produced (i.e. 10 of size 5, 5 of size 10, 2 of size 20 and 2 of size 30). The results of this set of experiments are shown in Figure 1.

It can be seen that the random masks have an accuracy slightly inferior to the other masks averaging about 58.2% and 58.9% respectively (using a simple count of correct classifications as a measure of accuracy). For the various ensemble sizes there is very little difference in the diversity between the two scenarios. Thus the ensembles based on the better quality masks produce the best results with the ensemble of size 50 producing an accuracy in leave-one-out testing on the full data set of 64.5%. It is important to note that this cannot be claimed as an estimate of generalisation accuracy for the system since the masks in use may overfit this data since some of the data was used in producing the masks. An interesting aspect of the data shown in Figure 1 is that the measure of ensemble diversity used seems to capture the increase in diversity

with ensemble size. As the benefit of increasing the ensemble size tails off around 30-40 members so does the increase in entropy. This would not be the case with the measure of diversity proposed by Ho (see section 3) for instance.

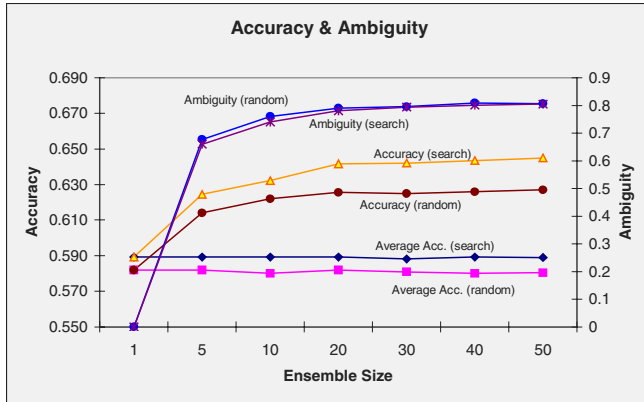


Fig. 1. Measurements of accuracy and ambiguity of ensembles working on the IVF data

This first experiment shows that if accuracy of ensemble members is increased without reducing ambiguity it will increase overall accuracy. In the next experiment we will show how the ambiguity of the ensemble predicts the reduction in error (increase in accuracy) due to the ensemble. These results are shown in Figure 2. In each graph the Y-axis shows the difference between the average error of the ensemble members and the ensemble error. In Figure 2(a) the error is a simple count of correct classifications; in (b) the conditional entropy is used as described in section 3. These graphs suggest that ambiguity as measured by entropy better predicts error reductions when error is measured as conditional entropy, i.e. the relationship in the graph on the right is clearer than that in the graph on the left. This is borne out by the correlation coefficient in both cases; the correlation coefficient for the relationship between change in correct count and entropy is 0.81 while that between change in error as measured by conditional entropy and ensemble entropy is 0.91.

Indeed it might be argued that, even without this useful relationship with ensemble ambiguity, conditional entropy is a particularly appropriate measure of error. After all, it does capture the importance of good accuracy spread across all categories. With the data presented here a good score based on count correct may conceal poor performance on the minority class.

Finally we show how the information provided by the use of entropy as a measure of diversity can inform the construction of a very good quality ensemble. The analysis shows that, in this domain, seeking good quality masks appears not to compromise ensemble diversity. We suggest that this is due to the large number of features in the domain (53). So it should be possible to increase the quality of the ensemble members without loss of diversity. 66 good quality masks were produced using the process described in section 4.2 and their accuracy was tested using leave-one-out testing on the full dataset of 1355 samples. Using this metric of quality the best 20 of these were chosen to form an ensemble. The accuracy of this ensemble measured using leave-

one-out testing on the whole dataset was 66.9%, better than the average of 64.2% for the other two ensembles of size 20 and better than the 64.5% figure for best ensemble of size 50.

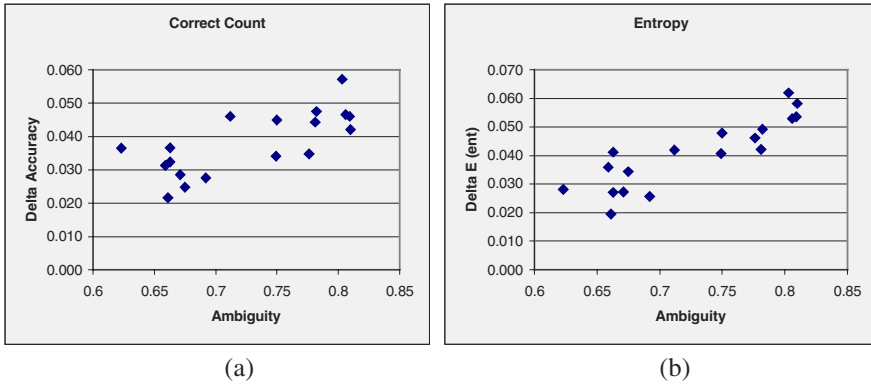


Fig. 2. Plots of the relationship between the reduction in error due to an ensemble and the ambiguity of the ensemble. In (a) error is measured as a count of correct classifications; in (b) it is measured as conditional entropy

6. Conclusion

The main message in this paper is that any work with classification ensembles should explicitly measure diversity in the ensemble and use this measure to guide decisions on the constitution of the ensemble as shown in the last section.

We show that in the same way that variance is a good measure of diversity for regression problems entropy is a useful measure of diversity for classification ensembles. Then associated with entropy as a measure of diversity is conditional entropy as an appropriate error function.

As advocated by Ho and by Guerra-Salcedo and Whitley feature subsets are a useful mechanism for introducing diversity in an ensemble of k -NN classifiers. If the feature space under consideration is large (> 30) then there may be less risk of loss of diversity in searching for good quality ensemble members. In the future we propose to evaluate this analysis on problems with smaller numbers of features where there may be a more clear-cut trade-off between ambiguity and quality of ensemble components.

Finally the quality of this ensemble of classifiers based on components with different feature subsets raises some questions about the issue of feature subset selection with which we opened this paper. The ensemble of classifiers has a better classification performance than any of its individual components. This brings into question the whole feature subset selection idea because it suggests that there is not one global feature set that provides a ‘best’ problem representation. Instead the ensemble exploits a variety of representations that may be combining locally in different parts of the problem space.

The next step is to evaluate these metrics on different classification datasets - However, leave-one-out testing on an ensemble of lazy learners is very

computationally expensive. It will be particularly interesting to see if the entropy measure of diversity does in fact capture aspects of ensemble size as happens with this data set. For the future it will be interesting to tackle the problem of overfitting in the feature selection process.

Acknowledgements

We would like to thank John Haslett for suggesting Entropy as a suitable measure of diversity and also for recommending Conditional Entropy as an associated loss function.

References

- [1] Aha, D. W., & Bankert, R. L., Feature selection for case-based classification of cloud types: An empirical comparison. In D. W. Aha (Ed.) *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press. (NCARAI TR: AIC-94-011), 1994.
- [2] Bonzano A., Cunningham P., Smyth B., Using introspective learning to improve retrieval in CBR: A case study in air traffic control, *Case-Based Reasoning Research and Development*, Proceedings of the 1997 *International Conference on Case-Based Reasoning*, D.B. Leake and E. Plaza Eds., Springer Verlag, Lecture Notes in Artificial Intelligence, pp.291-302, 1997.
- [3] Carney, J., Cunningham, P., The NeuralBAG algorithm: optimizing generalization performance in bagged neural networks, in proceedings of 7th *European Symposium on Artificial Neural Networks*, Bruges (Belgium), pp35-50 1999.
- [4] Guerra-Salcedo, C., Whitley, D., Genetic Approach for Feature Selection for Ensemble Creation. in *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (eds.). Orlando, Florida USA, pp236-243, San Francisco, CA: Morgan Kaufmann, 1999.
- [5] Guerra-Salcedo, C., Whitley, D., Feature Selection Mechanisms for Ensemble Creation: A Genetic Search Perspective, in *Data Mining with Evolutionary Algorithms: Research Directions. Papers from the AAAI Workshop*. Alex A. Freitas (Ed.) Technical Report WS-99-06. AAAI Press, 1999.
- [6] Ho, T.K., The Random Subspace Method for Constructing Decision Forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 8, 832-844, 1998.
- [7] Ho, T.K., Nearest Neighbours in Random Subspaces, *Proc. Of 2nd International Workshop on Statistical Techniques in Pattern Recognition*, A. Amin, D. Dori, P. Puil, H. Freeman, (eds.) pp640-648, Springer Verlag LNCS 1451, 1998.
- [8] Krogh, A., Vedelsby, J., Neural Network Ensembles, Cross Validation and Active Learning, in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretsky, T. K. Leen, eds., pp231-238, MIT Press, Cambridge MA, 1995.
- [10] Tibshirani, R., (1996) Bias, variance and prediction error for classification rules, University of Toronto, Department of Statistics Technical Report, November 1996 (also available at www-stat.stanford.edu/~tibs), 1996.
- [11] Wettschereck, D., Aha, D. W., & Mohri, T. , A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, pp273-314, Vol. 11, Nos. 1-5, 1997.

Minimax TD-Learning with Neural Nets in a Markov Game

Fredrik A. Dahl and Ole Martin Halck

Norwegian Defence Research Establishment (FFI)
P.O. Box 25, NO-2027 Kjeller, Norway
{Fredrik-A.Dahl, Ole-Martin.Halck}@ffi.no

Abstract. A minimax version of temporal difference learning (minimax TD-learning) is given, similar to minimax Q-learning. The algorithm is used to train a neural net to play Campaign, a two-player zero-sum game with imperfect information of the Markov game class. Two different evaluation criteria for evaluating game-playing agents are used, and their relation to game theory is shown. Also practical aspects of linear programming and fictitious play used for solving matrix games are discussed.

1 Introduction

An important challenge to artificial intelligence (AI) in general, and machine learning in particular, is the development of agents that handle uncertainty in a rational way. This is particularly true when the uncertainty is connected with the behavior of other agents.

Game theory is the branch of mathematics that deals with these problems, and indeed games have always been an important arena for testing and developing AI. However, almost all of this effort has gone into deterministic games like chess, go, othello and checkers. Although these are complex problem domains, uncertainty is not their major challenge.

With the successful application of temporal difference learning, as defined by Sutton [1], to the dice game of backgammon by Tesauro [2], random games were included as a standard testing ground. But even backgammon features perfect information, which implies that both players always have the same information about the state of the game.

We believe that imperfect information games, like poker, are more challenging and also more relevant to real world applications. Imperfect information introduces uncertainty about the opponent's current and previous states and actions, uncertainty that he cannot quantify as probabilities because he does not know his opponent's strategy. In games like chess and backgammon deception and bluff has little relevance, because a player's state and actions are revealed immediately, but with imperfect information these are important concepts.

As Koller and Pfeffer [3] and Halck and Dahl [4] have observed, imperfect information games have received very little attention from AI researchers. Some recent exceptions are given by Billings et al [5] and Littman [6].

The present article deals with minimax TD-learning, a value-based reinforcement learning algorithm that is suitable for a subset of two-player zero-sum games called Markov games. The set of Markov games contains some, but not all, imperfect information games, and represents a natural extension of the set of perfect information games. The algorithm is tested on a military air campaign game using a neural net.

The article is structured as follows. Section 2 covers some elementary game theory. Section 3 gives two evaluation criteria that we use for evaluating the performance of game-playing agents. In Section 4 the game Campaign, which will serve as testing-ground for our algorithm, is defined. Section 5 gives the definition of our reinforcement learning algorithm. Section 6 presents implementation and experimental results, and Section 7 concludes the article.

2 Game Theory

We now give a brief introduction to some elementary game-theoretic concepts. The theory we use is well covered by e.g. Luce and Raiffa [7].

A *game* is a decision problem with two or more decision-makers, called *players*. Each player evaluates the possible game outcomes according to some *payoff* (or *utility*) function, and attempts to maximize the expected payoff of the outcome. In this article we restrict our attention to *two-player zero-sum games*, where the two players have opposite payoffs, and therefore have no incentive to co-operate. We denote the players *Blue* and *Red*, and see the game from Blue's point of view, so that the payoff is evaluated by Blue. The zero-sum property implies that Red's payoff is equal to Blue's negated. Note that constant-sum games, where Blue's and Red's payoffs add to a fixed constant c for all outcomes, can trivially be transformed to zero-sum games by subtracting $c/2$ from all payoff values.

A *pure strategy* for a player is a deterministic plan that dictates all decisions the player may face in the course of a game. A *mixed*, or *randomized*, strategy is a probability distribution over a set of pure strategies.

Under mild conditions (e.g. finite sets of pure strategies) a two-player zero-sum game has a *value*. This is a real number v such that Blue has a (possibly mixed) strategy that guarantees the expected payoff to be no less than v , and Red has a strategy that guarantees it to be no more than v . A pair of strategies for each side that has this property is called a *minimax solution* of the game. These strategies are in equilibrium, as no side can profit from deviating unilaterally, and therefore minimax play is considered optimal for both sides.

Games of *perfect information* are an important subclass of two-player zero-sum games containing games like chess and backgammon. With perfect information both players know the state of the game at each decision point, and the turn of the players alternate. In perfect information games there exist minimax solutions that consist of pure strategies, so no randomization is required. In *imperfect information games* like two-player poker, however, minimax play will often require mixed strategies. Any experienced poker player knows that deterministic play is vulnerable. Randomization is best seen as a defensive maneuver protecting against an intelligent opponent who

may predict your behavior. In chess this plays little part, because your actions are revealed to the opponent immediately.

2.1 Matrix Games

A special class, or rather representation, of two-player zero-sum games is *matrix games*. In a matrix game both players have a finite set of pure strategies, and for each combination of Blue and Red strategy there is an instant real valued reward. The players make their moves simultaneously. If Blue's strategies are numbered from 1 to m , and Red's are numbered from 1 to n , the game can be represented by an $m \times n$ matrix $\mathbf{M}$ whose entry m_{ij} equals Blue's payoff if Blue and Red use strategies i and j respectively. Any finite two-player zero-sum game can be transformed to matrix form by enumerating the strategies. If the game is stochastic, the matrix entries will be expected payoffs given Blue and Red strategy. However, if the game has sequences of several decisions, the dimensions of the matrix will grow exponentially, making the matrix representation impractical to produce.

It has long been known that the problem of finding a minimax solution to a matrix game is equivalent to solving a linear programming problem (LP), see Strang [8]. Efficient algorithms exist for LP, such as the simplex procedure. We will return to this in the implementation section.

3 Evaluation Criteria

Our goal is to use machine learning techniques to develop agents that play two-player zero-sum games well. To quantify success of our agents, we need to define evaluation criteria. This is not quite as straightforward as one might believe, because game outcomes are not in general transitive. Even if agent A beats agent B every time, and B beats C consistently, it may well be that C beats A all the time. The obvious example of this is the well-known game scissors-paper-rock. Therefore, one cannot develop a strength measure that ranks a pool of agents consistently so that stronger agents beat weaker ones. Instead we seek to develop evaluation criteria that conform to game theory. These criteria have previously been published in Halck and Dahl [4].

3.1 *Geq*

Our strictest evaluation criterion is called *equity against globally optimizing opponent*, abbreviated *Geq*. The *Geq* of an agent is the minimum of the player's expected outcome, taken over the set of all possible opponents. The *Geq* is less than or equal to the game's value, with equality if and only if the agent is a minimax solution.

Let P_1 and P_2 be agents, and let P be the randomized agent that uses P_1 with probability p , $0 < p < 1$, and P_2 with probability $1 - p$. (Of course, P_1 and P_2 may also

contain randomization, and this is assumed independent of P 's randomization between P_1 and P_2 .) Then

$$Geq(P) \geq p \cdot Geq(P_1) + (1-p) \cdot Geq(P_2). \quad (1)$$

This is easily seen by observing that the most dangerous opponent of P_1 may be different from that of P_2 . Inequality (1) shows that mixing strategies with similar Geq is beneficial according to the Geq measure, particularly if the component players have different weaknesses.

Mixing of strategies is most important in games of imperfect information, where this is required for minimax play, but even in games of perfect information it will often improve the Geq . Consider chess, and imagine a version of IBM's Deep Blue that plays deterministically. As long as there is even just a single way of tricking the program, its Geq would be zero. On the other hand, a completely random agent would do better, getting a positive Geq , as there is no way of beating it with probability 1.

3.2 Peq

Our second performance measure is *equity against perfect opponent*, abbreviated Peq . The Peq of an agent is its expected outcome against a minimax-playing opponent. Note that minimax solutions are not in general unique, so there may actually be a family of related Peq measures to choose from. In the following we assume that one of them is fixed.

For all agents $Peq \geq Geq$, as the minimax agent is included in the set of opponents that the Geq calculations minimize over. The Peq measure also has the game's value as its maximum, and a minimax-playing agent achieves this. But this is not a sufficient condition for minimax play. Consider again our agent P as the mixture of agents P_1 and P_2 . The Peq measure satisfies the following equation:

$$Peq(P) = p \cdot Peq(P_1) + (1-p) \cdot Peq(P_2). \quad (2)$$

This property follows directly from the linearity of the expected value. Equation (2) tells us that according to the Peq measure there is nothing to be gained by randomizing. This makes sense, because randomization is a defensive measure taken only to ensure that the opponent does not adjust to the agent's weaknesses. When playing against a static opponent, even a perfect one, there is no need for randomization. Equation (2) implies that Peq only measures an agent's ability to find strategies that may be a component of some minimax solution.

This touches a somewhat confusing aspect of the minimax solution concept. Like we stated in the game theory section, a pair of agents that both play a minimax solution is in equilibrium, as neither can gain by deviating. However, the equilibrium is not very coercive, because one agent does not have anything to gain from randomizing as long as the other agent does. As we have just seen, all it takes to secure the value against a minimax-playing opponent is any deterministic strategy that may be a randomized component of a minimax solution.

This can be illustrated with an example from poker. In some poker situations it is correct (in the minimax sense) for a player to bluff with a given probability, and for

the opponent to call with a different probability. But the optimal bluffing probability is exactly the one that makes calling and folding equally strong for the opponent. And similarly, if the opponent calls with his optimal probability, it makes no difference if the first player bluffs all the time or not at all.

4 Campaign

In this article we will describe and explore an algorithm that is defined for *Markov games*, which is a proper subclass of two-player zero-sum games. Markov games include some, but not all, games with imperfect information. We have developed our own game, called Campaign, which features imperfect information, as testing ground for agents. Rather than burdening the reader with formal definitions, we present Campaign as an example of a Markov game, and describe general Markov games informally afterwards. Campaign was first defined and analyzed in Dahl and Halck [9].

4.1 Rules

Both players start the game with five *units* and zero *accumulated profit*. There are five consecutive *stages*, and at each stage both players simultaneously allocate their available units between three *roles*: *defense* (*D*), *profit* (*P*) and *attack* (*A*). A unit allocated to *P* increases the player's accumulated profit by one point. Each unit allocated to *D* *neutralizes* two opponent attacking units. Each unit allocated to *A*, and not neutralized, destroys one opponent unit for the remaining stages of the game. Before each stage the players receive information about both side's accumulated profit and number of remaining units. After the last stage the *score* for each player is calculated as the sum of accumulated profit and number of remaining units. The player with the higher score wins, and with equality the game is a draw. Margin of victory is irrelevant. If both players evaluate a draw as "half a win", the game is zero sum. We assign the payoffs 0, 0.5 and 1 to losing, drawing and winning, respectively, which technically makes the game constant-sum. The rules are symmetric for Blue and Red, and the value is clearly 0.5 for both. Campaign has imperfect information due to the simultaneity of the player's actions at each stage.

The military interpretation of the game is an air campaign. Obviously, a model with so few degrees of freedom can not represent a real campaign situation accurately, but it does capture essential elements of campaigning. After Campaign was developed, we discovered that it was in fact very similar to "The Tactical Air Game" developed by Berkovitz [10]. This may indicate that Campaign is a somewhat canonical air combat model.

Define the game's *state* as a four-tuple (b, r, p, n) , with b being the number of remaining Blue units, r the number of Red units, p Blue's lead in accumulated profit points and n the number of rounds left. The initial state of the game is then (5,5,0,5). Note that it is sufficient to represent the difference in the players' accumulated profit, as accumulated profit only affects evaluation of the final outcome, and not the dynamics of the game. Our state representation using Blue's lead in accumulated

profit introduces some asymmetry in an otherwise symmetric game, but this is of no relevance. For that matter, both players may regard themselves as Blue, in which case a state seen as (a, b, c, d) for one player, would be perceived as $(b, a, -c, d)$ to the other. An allocation is represented as a three-tuple (D, P, A) of natural numbers summing to the side's number of remaining units. A sample game is given in Table 1.

Table 1. A sample game of Campaign

Stage	State	Blue action	Red action
1	(5,5,0,5)	(2,2,1)	(2,3,0)
2	(5,5,-1,4)	(1,4,0)	(2,3,0)
3	(5,5,0,3)	(2,2,1)	(0,0,5)
4	(4,4,2,2)	(1,2,1)	(1,0,3)
5	(3,4,4,1)	(0,3,0)	(0,4,0)
Final state:	(3,4,3,0)		

Blue wins the game, as his final lead in accumulated profit (3) is larger than his deficit in remaining units ($4 - 3 = 1$).

4.2 Solution

Because perfect information is available to the players before each stage, earlier states visited in the game can be disregarded. It is this property that makes the game solvable in practical terms with a computer. As each state contains perfect information, it can be viewed as the starting point of a separate game, and therefore has a value, by game theory. This fact we can use to decompose the game further by seeing a state as a separate game that ends after both players have made their choice. The outcome of this game is the value of the next state reached. At each state both players have at most 21 legal allocations, so with our decomposition a game state's value is defined as the value of a matrix game with at most 21 pure strategies for both sides, with matrix entries being values of succeeding game states. One can say that our solution strategy combines dynamic programming and solution of matrix games. First all games associated with states having one stage left are solved using linear programming. (Again we refer to the implementation section for a discussion of solution algorithms for matrix games.) These have matrix entries defined by the rules of the game. Then states with two stages remaining are resolved, using linear programming on the game matrices resulting from the previous calculations, and so on.

To shed some light on what the game solution looks like, we will give some examples of states with solutions. For all states $(b, r, p, 1)$, that is, the last stage of the game, allocating all units to profit is a minimax solution. Other states have far more complex solutions. Figure 1 shows the solutions for three different states, in which solutions are unique. In each one both players have all five units remaining, and their complete state descriptions are (5,5,0,5), (5,5,0,3) and (5,5,2,2). Superficially these states appear similar, but as the figure shows, their solutions are very different. Note in particular that the allocation of one unit to defense, three to profit and one to attack is not given positive probability for the first two states, and it is in general rarely a

good allocation. But in the special case of Blue leading with two points, with all units intact and two stages to go, it is the only allocation that forces a win.

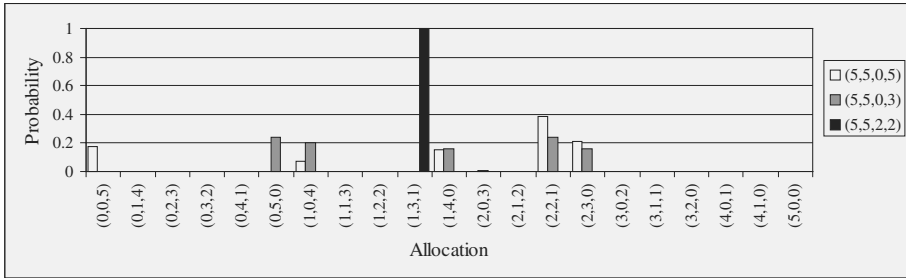


Fig. 1. Examples of solutions for some states

These examples show that apparently similar game states may have very different solutions. Therefore the game should pose a serious challenge to machine learning techniques.

4.3 Markov Games

We mentioned above that Campaign belongs to the game class called Markov games (see e.g. [6]). Markov games have the same general structure as Campaign with the players making a sequence of simultaneous decisions. The “Markov” term is taken from stochastic process theory, and loosely means that the future of a game at a given state is independent of its past, indicating that the state contains all relevant information concerning the history of the game.

There are three general features that Markov games may have that Campaign does not have. Firstly, the game may return to states that have previously been visited, creating cycles. Secondly, there may be payoffs associated with all state-action combinations, not just the terminal game states. The combination of these effects, cycles and payoffs of non-terminal states, opens the possibility of unlimited payoff accumulations, and this is usually prevented by some discounting factor that decreases exponentially with time. Thirdly, there may be randomness in the rules of the game, so that a triple (blue-action, state, red-action) is associated with a probability distribution over the set of states, rather than just a single state.

Markov games extend the class of perfect information games into the area of imperfect information. Note that perfect information games are included in Markov games by collapsing the decision set of the player that is not on turn to a single action. Therefore the simultaneous decisions trivially include sequential alternating decisions. Markov games can also be seen as a generalization of Markov decision problems (MDP), as an MDP is a “game” where the opponent’s options are collapsed completely.

5 Minimax TD-Learning

Littman [6] defines minimax Q-learning for Markov games, which is similar to Q-learning in MDP, and uses this successfully for a simple soccer game. If the agent one is training knows the rules of the game (called *complete information* in the game theory language), a simpler learning rule can be used, that only estimates values of states, and this is what we do in this article. We are not aware of this algorithm being published previously, and give it the natural name of *minimax TD-learning*. We do not claim that this is an important new concept, more of a modification of minimax Q-learning. Minimax TD-learning is in fact even more similar to standard TD-learning for MDP than minimax Q-learning is to Q-learning. We assume the reader is familiar with TD-learning. Barto et al [11] gives an overview of TD-learning and other machine learning algorithms related to dynamic programming.

We describe the Minimax TD-learning method with Campaign in mind, but it should be obvious how it works for more general Markov games, featuring the general properties not present in Campaign. Minimax TD-learning trains a state evaluator to estimate the game-theoretic minimax value of states. This state evaluator, be it a lookup table, a neural net or whatever, is used for playing games, and standard TD(λ)-learning is used to improve estimates of state values based on the sequence of states visited.

The way that the state evaluator controls the game, however, is different from the MDP case. At each state visited a game matrix is assembled. For each combination of Blue and Red strategy the resulting state is calculated (which is why the algorithm requires knowledge of the rules). The evaluator's value estimate of that state is used as the corresponding game matrix entry. If the resulting state is a terminal one, the actual payoff is used instead. Then the matrix game is solved, and random actions are drawn for Blue and Red according to the resulting probability distributions. This procedure is repeated until the game terminates. A long sequence of games will normally be needed to get high quality estimations from the TD-learning procedure.

It is a well-known fact that TD-learning in MDPs may get stuck with a sub-optimal solution, unless some measures are taken that forces the process to explore the state space. This may of course happen with Markov games as well, being a superset of MDPs.

6 Implementation and Experimental Results

In this section we describe implementation issues concerning our state evaluator, different techniques used for solving matrix games, calculation of performance and experience with the learning algorithm itself.

6.1 Neural Net State-Evaluator

We have implemented our state evaluator as a neural net. The net is a “vanilla-flavored” design with one layer of hidden units, sigmoid activation functions, and

back-propagation of errors. The net has four input nodes associated to the state variables (b, r, p, n) , each scaled by a factor 0.2 to get an approximate magnitude range of $[0,1]$. The net has one output node, which gives the estimated state value. The number of hidden nodes was set to eight.

6.2 Solving Matrix Games by Linear Programming

It is a well-established fact that matrix games can be solved by LP techniques, see e.g. Strang [8]. However, the practical problems encountered when implementing and using the simplex algorithm surprised us, and we would like to share this with the public. The problems would surely be less if we had used a commercial LP package, but that would require close integration of it into our program, which was not desirable. Instead we copied the simplex procedure published in [12].

Recall that our game matrix $\mathbf{M} \in \mathbf{R}^{m \times n}$ has as entry m_{ij} Blue's (expected) payoff when Blue uses his strategy with index i and Red uses his strategy j . We see the game from Blue's side, so we surely need variables $x_1, \dots, x_m$ that represent his probability distribution. These will give the randomizing probabilities associated with his pure strategies. They must be non-negative (which fits the standard representation of LP problems), and sum to 1, to represent a probability distribution: $\sum_{i=1}^m x_i = 1$.

We also need a variable x_{m+1} for the game's value, which is not necessarily non-negative. A standard trick for producing only non-negative variables is to split the unbounded variable into its positive and negative parts: $x_{m+1} = v - u$. We do not have any convincing explanation of it, but from our experience this was not compatible with the simplex algorithm used, as it claimed the solution was unbounded. A different problem arose in some cases when the optimal value was exactly 0, as it was unable to find any feasible solution. This is probably due to rounding errors. To eliminate these problems we transformed the matrix games by adding a constant, slightly higher than the minimum matrix entry negated, to all matrix entries, thereby keeping the solution structure and ensuring strictly positive value. Afterwards the same constant must of course be deducted from the calculated value.

Minimax solutions are characterized by the fact that Blue's expected payoff is no less than the value, whichever pure strategy Red uses. For each $j \in \{1, \dots, n\}$ this gives the inequality $\sum_{i=1}^m x_i \cdot m_{ij} - x_{m+1} \geq 0$. The objective function is simply the value x_{m+1} .

With this problem formulation the simplex procedure appeared to be stable, but only with double precision floating point numbers.

6.3 Solving Matrix Games by Fictitious Play

During our agonizing problems with the simplex algorithm we quickly implemented an iterative algorithm for matrix games called *fictitious play*. This algorithm is also far from new, see Luce and Raiffa [7]. Fictitious play can be viewed as a two-sided competitive machine learning algorithm. It works like this: Blue and Red sequentially find the most effective pure strategy, under the assumption that the opponent will play

according to the probability distribution manifested in the histogram of his actions in all previous iterations. The algorithm is very simple to implement, and it is completely stable. Its convergence is inverse linear, which does not compete with the simplex algorithm that reaches the exact solution in a finite number of steps. However, we do not need exact solutions in the minimax TD-training, because the game matrices are also not exact. From our experience fictitious play is faster than simplex for the required precision in training. But when it comes to calculating the Campaign solution, which is needed for evaluating the *Peq* of agents, the accuracy provided by the simplex algorithm is preferable. After this work had been done, we registered that Szepesvari and Littman [13] also suggests the use of fictitious play in minimax Q-learning to make the implementation “LP-free”.

6.4 Calculating *Geq* and *Peq* Performance

To measure the progress of our Campaign-playing agent we need to evaluate its *Geq* and *Peq* performance.

The *Geq* calculations are very similar to the algorithm we use for calculating the solution. Because the behavior of the agent that is evaluated (Blue) is given, the problem of identifying its most effective opponent degenerates to an MDP problem, which can be solved by dynamic programming. First Red’s most effective actions are calculated for states with one stage left $(b, r, p, 1)$. Then the resulting state values are used for identifying optimal actions at states with two stages left, and so on.

The *Peq* could be calculated in much the same way, except that no optimization is needed as both Blue and Red’s strategies are fixed. However, it is more efficient to propagate probability distributions forwards in the state space, and calculating the expected outcome with respect to the probability distribution at the terminal states. This saves time because calculations are done only for states that are visited when this Blue agent plays against the given minimax Red player.

6.5 Experimental Results

When used without exploration the minimax TD-algorithm did not behave well. The first thing the net learned was the importance of profit points. This led to game matrices that result in a minimax strategy of taking profit only, for all states. The resulting games degenerated to mere profit-taking by both sides, and all games were draws. Therefore it never discovered the fact that units are more important than profit points, particularly in early game states. In retrospect it is not surprising that the algorithm behaved this way, because the action of using all units for profit is optimal in the last stage of the game, which is likely to be the first thing it learns about.

One way of forcing the algorithm to explore the state space more is by introducing random actions different from those recommended by the algorithm. Instead we have been randomizing the starting state of the game. Half of the games were played from the normal starting state $(5,5,0,5)$, and the rest were drawn randomly. To speed up training, the random starting states were constructed to be “interesting”. This was

done by ensuring that a player cannot start with a lead in both number of units and profit points.

TD-learning is subject to random noise. We were able to reduce this problem by utilizing a symmetry present in the game. If a given state (b, r, p, n) receives feedback v , it implicitly means that the state $(r, b, -p, n)$ seen by the opponent deserves feedback $1 - v$. Adding this to the training procedure helps the net towards consistent evaluations, and automatically ensures that symmetric states (like the starting state) get neutral feedback (that is 0.5) on average. This reduced the random fluctuations in the net's evaluations considerably.

With these modifications the algorithm behaved well. Figure 2 shows the learning curves of the agent's *Geq* and *Peq*, with $\lambda = 0$ and learning rate decreasing from 1 to 0.1. The number of iterations in the fictitious play calculations was increased linearly from 100 to 500. The unit on the x-axis is 1000 games, and the curves are an average of five training batches.

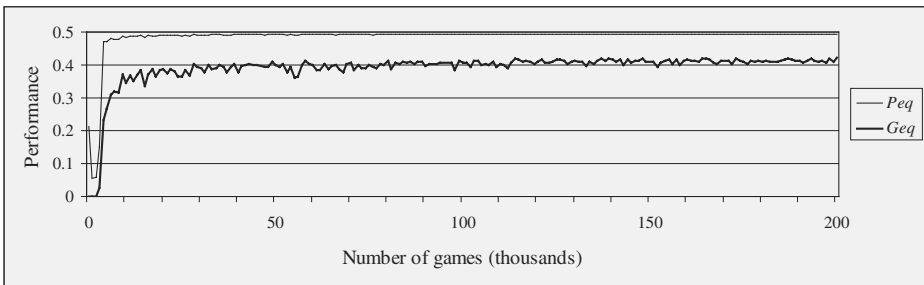


Fig. 2. *Geq* and *Peq* learning curves

We see that the *Peq* of the agent quickly approaches the optimal 0.5. The *Geq* values do not quite reach this high, but the performance is acceptable in light of the relatively small neural net used. The *Geq* is close to zero in the first few thousand games, and the *Peq* also has dip in the same period. This is because the agent first learns the value of profit points, and it takes some time before the exploration helps it to discover the value of units.

7 Conclusion

Our main conclusion is that minimax TD-learning works quite well for our Markov game named Campaign. Unlike the experience of Littman [6], the algorithm fails completely without forced exploration, but our exploration technique of randomizing the starting point of the game appears successful.

The results show that it is far easier to achieve high performance according to the *Peq* measure (expected outcome against a minimax-playing opponent) than according to *Geq* (expected outcome against the agent's most effective opponent).

Our experience indicates that the simple fictitious play algorithm can compete with LP algorithms for producing solutions for matrix games in cases where high precision is not needed. As a bonus fictitious play is also far simpler to implement.

References

1. Sutton, R.S.: Learning to predict by the methods of temporal differences, *Machine Learning* **3** (1988) 9–44.
2. Tesauro, G.J.: Practical issues in temporal difference learning, *Machine Learning* **8** (1992) 257–277.
3. Koller, D., Pfeffer, A.: Representations and solutions for game-theoretic problems. *Artificial Intelligence* **94**(1) (1997) 167–215.
4. Halck, O.M., Dahl, F.A.: On classification of games and evaluation of players – with some sweeping generalizations about the literature. In: Fürnkranz, J., Kubat, M. (eds.): *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing*, Jozef Stefan Institute, Ljubljana (1999).
5. Billings, D., Papp, D., Schaeffer, J., Szafron, D.: Poker as a testbed for machine intelligence research. In: Mercer, R., Neufeld, E. (eds.): *Advances in Artificial Intelligence*, Springer-Verlag, Berlin–Heidelberg–New York (1998) 228–238.
6. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, New Brunswick (1994) 157–163.
7. Luce, R.D., Raiffa, H.: *Games and Decisions*. Wiley, New York (1957).
8. Strang, G.: *Linear Algebra and Its Applications. Second Edition*. Harcourt Brace Jovanovich, Orlando (1980).
9. Dahl, F.A., Halck, O.M.: Three games designed for the study of human and automated decision making. Definition and properties of the games Campaign, Operation Lucid and Operation Opaque. FFI/RAPPORT-98/02799, Norwegian Defence Research Establishment (FFI), Kjeller, Norway (1998).
10. Berkovitz, L.D.: The Tactical Air Game: A multimove game with mixed strategy solution. In: Grote, J.D. (ed.): *The Theory and Application of Differential Games*, Reidel Publishing Company, Dordrecht, The Netherlands (1975) 169–177.
11. Barto, A.G., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. *Artificial Intelligence* **72** (1995) 81–138.
12. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK (1988).
13. Szepesvari, C., Littman, M.L.: A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation* **11** (1999) 2017–2060.

Boosting Applied to Word Sense Disambiguation

Gerard Escudero, Lluís Màrquez, and German Rigau*

TALP Research Center, LSI Department. Universitat Politècnica de Catalunya (UPC)
Jordi Girona Salgado 1-3, E-08034, Barcelona. Catalonia
{escudero,lluism,g.rigau}@lsi.upc.es

Abstract. In this paper Schapire and Singer’s AdaBoost.MH boosting algorithm is applied to the Word Sense Disambiguation (WSD) problem. Initial experiments on a set of 15 selected polysemous words show that the boosting approach surpasses Naive Bayes and Exemplar-based approaches, which represent state-of-the-art accuracy on supervised WSD. In order to make boosting practical for a real learning domain of thousands of words, several ways of accelerating the algorithm by reducing the feature space are studied. The best variant, which we call **LazyBoosting**, is tested on the largest sense-tagged corpus available containing 192,800 examples of the 191 most frequent and ambiguous English words. Again, boosting compares favourably to the other benchmark algorithms.

1 Introduction

Word Sense Disambiguation (WSD) is the problem of assigning the appropriate meaning (sense) to a given word in a text or discourse. This meaning is distinguishable from other senses potentially attributable to that word. Resolving the ambiguity of words is a central problem for language understanding applications and their associated tasks [11], including, for instance, machine translation, information retrieval and hypertext navigation, parsing, spelling correction, reference resolution, automatic text summarization, etc.

WSD is one of the most important open problems in the Natural Language Processing (NLP) field. Despite the wide range of approaches investigated and the large effort devoted to tackling this problem, it is a fact that to date no large-scale, broad coverage and highly accurate word sense disambiguation system has been built.

The most successful current line of research is the corpus-based approach in which statistical or Machine Learning (ML) algorithms have been applied to learn statistical models or classifiers from corpora in order to perform WSD. Generally, supervised approaches (those that learn from a previously semantically annotated corpus) have obtained better results than unsupervised methods on small sets of selected highly ambiguous words, or artificial pseudo-words. Many

* This research has been partially funded by the Spanish Research Department (CI-CYT’s BASURDE project TIC98-0423-C06) and by the Catalan Research Department (CIRIT’s consolidated research group 1999SGR-150, CREL’s Catalan WordNet project and CIRIT’s grant 1999FI 00773).

standard ML algorithms for supervised learning have been applied, such as: Naive Bayes [19,22], [19,10], Exemplar-based learning Decision Lists [28], Neural Networks [27], etc. Further, Mooney [17] has also compared all previously cited methods on a very restricted domain and including Decision Trees and Rule Induction algorithms. Unfortunately, there have been very few direct comparisons of alternative methods on identical test data. However, it is commonly accepted that Naive Bayes, Neural Networks and Exemplar-based learning represent state-of-the-art accuracy on supervised WSD.

Supervised methods suffer from the lack of widely available semantically tagged corpora, from which to construct really broad coverage systems. This is known as the “knowledge acquisition bottleneck”. Ng [20] estimates that the manual annotation effort necessary to build a broad coverage semantically annotated corpus would be about 16 man-years. This extremely high overhead for supervision and, additionally, the also serious overhead for learning/testing many of the commonly used algorithms when scaling to real size WSD problems, explain why supervised methods have been seriously questioned.

Due to this fact, recent works have focused on reducing the acquisition cost as well as the need for supervision in corpus-based methods for WSD. Consequently, the following three lines of research can be found: 1) The design of efficient example sampling methods [6,10]; 2) The use of lexical resources, such as WordNet [16], and WWW search engines to automatically obtain from Internet arbitrarily large samples of word senses [12,15]; 3) The use of unsupervised EM-like algorithms for estimating the statistical model parameters [22]. It is also our belief that this body of work, and in particular the second line, provides enough evidence towards the “opening” of the acquisition bottleneck in the near future. For that reason, it is worth further investigating the application of new supervised ML methods to better resolve the WSD problem.

Boosting Algorithms. The main idea of boosting algorithms is to combine many simple and moderately accurate hypotheses (called weak classifiers) into a single, highly accurate classifier for the task at hand. The weak classifiers are trained sequentially and, conceptually, each of them is trained on the examples which were most difficult to classify by the preceding weak classifiers.

The AdaBoost.MH algorithm applied in this paper [25] is a generalization of Freund and Schapire’s AdaBoost algorithm [9], which has been (theoretically and experimentally) studied extensively and which has been shown to perform well on standard machine-learning tasks using also standard machine-learning algorithms as weak learners [23,8,5,2].

Regarding Natural Language (NL) problems, AdaBoost.MH has been successfully applied to Part-of-Speech (PoS) tagging [1], Prepositional-Phrase-attachment disambiguation [1], and, Text Categorization [26] with especially good results.

The Text Categorization domain shares several properties with the usual settings of WSD, such as: very high dimensionality (typical features consist in testing the presence/absence of concrete words), presence of many irrelevant and highly dependent features, and the fact that both, the learned concepts and the

examples, reside very sparsely in the feature space. Therefore, the application of AdaBoost.MH to WSD seems to be a promising choice. It has to be noted that, apart from the excellent results obtained on NL problems, AdaBoost.MH has the advantages of being theoretically well founded and easy to implement.

The paper is organized as follows: Section 2 is devoted to explain in detail the AdaBoost.MH algorithm. Section 3 describes the domain of application and the initial experiments performed on a reduced set of words. In Section 4 several alternatives are explored for accelerating the learning process by reducing the feature space. The best alternative is fully tested in Section 5. Finally, Section 6 concludes and outlines some directions for future work.

2 The Boosting Algorithm AdaBoost.MH

This section describes the Schapire and Singer’s AdaBoost.MH algorithm for multiclass multi-label classification, using exactly the same notation given by the authors in [25,26].

As already said, the purpose of boosting is to find a highly accurate classification rule by combining many **weak hypotheses** (or weak rules), each of which may be only moderately accurate. It is assumed that there exists a separate procedure called the **WeakLearner** for acquiring the weak hypotheses. The boosting algorithm finds a set of weak hypotheses by calling the weak learner repeatedly in a series of T rounds. These weak hypotheses are then combined into a single rule called the **combined hypothesis**.

Let $S = \{(x_1, Y_1), \dots, (x_m, Y_m)\}$ be the set of m training examples, where each instance x_i belongs to an instance space $\mathcal{X}$ and each Y_i is a subset of a finite set of labels or classes $\mathcal{Y}$. The size of $\mathcal{Y}$ is denoted by $k = |\mathcal{Y}|$.

The pseudo-code of AdaBoost.MH is presented in figure 1. AdaBoost.MH maintains an $m \times k$ matrix of weights as a distribution D over examples and labels. The goal of the **WeakLearner** algorithm is to find a weak hypothesis with moderately low error with respect to these weights. Initially, the distribution D_1 is uniform, but the boosting algorithm updates the weights on each round to force the weak learner to concentrate on the pairs (examples,label) which are hardest to predict.

More precisely, let D_t be the distribution at round t , and $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ the weak rule acquired according to D_t . The sign of $h_t(x, l)$ is interpreted as a prediction of whether label l should be assigned to example x or not. The magnitude of the prediction $|h_t(x, l)|$ is interpreted as a measure of confidence in the prediction. In order to understand correctly the updating formula this last piece of notation should be defined. Thus, given $Y \subseteq \mathcal{Y}$ and $l \in \mathcal{Y}$, let $Y[l]$ be +1 if $l \in Y$ and -1 otherwise.

Now, it becomes clear that the updating function increases (or decreases) the weights $D_t(i, l)$ for which h_t makes a good (or bad) prediction, and that this variation is proportional to $|h_t(x, l)|$.

Note that WSD is not a multi-label classification problem since a unique sense is expected for each word in context. In our implementation, the algorithm runs

```

procedure AdaBoost.MH (in:  $S = \{(x_i, Y_i)\}_{i=1}^m$ )
###  $S$  is the set of training examples
### Initialize distribution  $D_1$  (for all  $i$ ,  $1 \leq i \leq m$ , and all  $l$ ,  $1 \leq l \leq k$ )
     $D_1(i, l) = 1/(mk)$ 
for  $t:=1$  to  $T$  do
    ### Get the weak hypothesis  $h_t : \mathcal{X} \times \mathcal{Y} \rightarrow$ 
     $h_t = \text{WeakLearner}(X, D_t)$ ;
    ### Update distribution  $D_t$  (for all  $i$ ,  $1 \leq i \leq m$ , and all  $l$ ,  $1 \leq l \leq k$ )
     $D_{t+1}(i, l) = \frac{D_t(i, l) \exp(-Y_i[l] h_t(x_i, l))}{Z_t}$ 
    ###  $Z_t$  is a normalization factor (chosen so that  $D_{t+1}$  will be a distribution)
end-for
return the combined hypothesis:  $f(x, l) = \sum_{t=1}^T h_t(x, l)$ 
end AdaBoost.MH

```

Fig. 1. The AdaBoost.MH algorithm

exactly in the same way as explained above, except that sets Y_i are reduced to a unique label, and that the combined hypothesis is forced to output a unique label, which is the one that maximizes $f(x, l)$.

Up to now, it only remains to be defined the form of the `WeakLearner`. Schapire and Singer [25] prove that the Hamming loss of the AdaBoost.MH algorithm on the training set¹ is at most $\prod_{t=1}^T Z_t$, where Z_t is the normalization factor computed on round t . This upper bound is used in guiding the design of the `WeakLearner` algorithm, which attempts to find a weak hypothesis h_t that minimizes: $Z_t = \sum_{i=1}^m \sum_{l \in \mathcal{Y}} D_t(i, l) \exp(-Y_i[l] h_t(x, l))$.

2.1 Weak Hypotheses for WSD

As in [1], very simple weak hypotheses are used to test the value of a boolean predicate and make a prediction based on that value. The predicates used, which are described in section 3.1, are of the form “ $f = v$ ”, where f is a feature and v is a value (e.g.: “previous_word = hospital”). Formally, based on a given predicate p , our interest lies on weak hypotheses h which make predictions of the form:

$$h(x, l) = \begin{cases} c_{0l} & \text{if } p \text{ holds in } x \\ c_{1l} & \text{otherwise} \end{cases}$$

where the c_{jl} ’s are real numbers.

For a given predicate p , and bearing the minimization of Z_t in mind, values c_{jl} should be calculated as follows. Let X_1 be the subset of examples for which the

¹ i.e. the fraction of training examples i and labels l for which the sign of $f(x_i, l)$ differs from $Y_i[l]$.

predicate p holds and let X_0 be the subset of examples for which the predicate p does not hold. Let $\llbracket \pi \rrbracket$, for any predicate π , be 1 if π holds and 0 otherwise. Given the current distribution D_t , the following real numbers are calculated for each possible label l , for $j \in \{0, 1\}$, and for $b \in \{+1, -1\}$:

$$W_b^{jl} = \sum_{i=1}^m D_t(i, l) \llbracket x_i \in X_j \wedge Y_i[l] = b \rrbracket$$

That is, W_{+1}^{jl} (W_{-1}^{jl}) is the weight (with respect to distribution D_t) of the training examples in partition X_j which are (or not) labelled by l .

As it is shown in [25], Z_t is minimized for a particular predicate by choosing:

$$c_{jl} = \frac{1}{2} \ln \left(\frac{W_{+1}^{jl}}{W_{-1}^{jl}} \right)$$

These settings imply that:

$$Z_t = 2 \sum_{j \in \{0, 1\}} \sum_{l \in \mathcal{Y}} \sqrt{W_{+1}^{jl} W_{-1}^{jl}}$$

Thus, the predicate p chosen is that for which the value of Z_t is smallest.

Very small or zero values for the parameters W_b^{jl} cause c_{jl} predictions to be large or infinite in magnitude. In practice, such large predictions may cause numerical problems to the algorithm, and seem to increase the tendency to overfit. As suggested in [26], smoothed values for c_{jl} have been used.

3 Applying Boosting to WSD

3.1 Corpus

In our experiments the boosting approach has been evaluated using the DSO corpus containing 192,800 semantically annotated occurrences² of 121 nouns and 70 verbs. These correspond to the most frequent and ambiguous English words. The DSO corpus was collected by Ng and colleagues [18] and it is available from the Linguistic Data Consortium (LDC)³.

For our first experiments, a group of 15 words (10 nouns and 5 verbs) which frequently appear in the related WSD literature has been selected. These words are described in the left hand-side of table 1. Since our goal is to acquire a classifier for each word, each row represents a classification problem. The number of classes (senses) ranges from 4 to 30, the number of training examples from 373 to 1,500 and the number of attributes from 1,420 to 5,181. The MFS column on the right hand-side of table 1 shows the percentage of the most frequent sense for each word, i.e. the accuracy that a naive “Most-Frequent-Sense” classifier would obtain.

The binary-valued attributes used for describing the examples correspond to the binarization of seven features referring to a very narrow linguistic context. Let “ $w_{-2} \ w_{-1} \ w \ w_{+1} \ w_{+2}$ ” be the context of 5 consecutive words around the

² These examples are tagged with a set of labels which correspond, with some minor changes, to the senses of WordNet 1.5 [21].

³ LDC e-mail address: ldc@unagi.cis.upenn.edu

word w to be disambiguated. The seven features mentioned above are exactly those used in [19]: w_{-2} , w_{-1} , w_{+1} , w_{+2} , (w_{-2}, w_{-1}) , (w_{-1}, w_{+1}) , and (w_{+1}, w_{+2}) , where the last three correspond to collocations of two consecutive words.

3.2 Benchmark Algorithms and Experimental Methodology

AdaBoost.MH has been compared to the following algorithms:

Naive Bayes (NB). The naive Bayesian classifier has been used in its most classical setting [4]. To avoid the effect of zero counts when estimating the conditional probabilities of the model, a very simple smoothing technique has been used, which was proposed in [19].

Exemplar-based learning (EB_k). In our implementation, all examples are stored in memory and the classification of a new example is based on a k -NN algorithm using Hamming distance to measure closeness (in doing so, all examples are examined). If k is greater than 1, the resulting sense is the weighted majority sense of the k nearest neighbours (each example votes its sense with a strength proportional to its closeness to the test example). Ties are resolved in favour of the most frequent sense among all those tied.

The comparison of algorithms has been performed in series of controlled experiments using exactly the same training and test sets for each method. The experimental methodology consisted in a 10-fold cross-validation. All accuracy/error rate figures appearing in the paper are averaged over the results of the 10 folds. The statistical tests of significance have been performed using a 10-fold cross validation paired Student's t -test with a confidence value of: $t_{9,0.975} = 2.262$.

3.3 Results

Figure 2 shows the error rate curve of AdaBoost.MH, averaged over the 15 reference words, and for an increasing number of weak rules per word. This plot shows that the error obtained by AdaBoost.MH is lower than those obtained by NB and EB₁₅ ($k=15$ is the best choice for that parameter from a number of tests between $k=1$ and $k=30$) for a number of rules above 100. It also shows that the error rate decreases slightly and monotonically, as it approaches the maximum number of rules reported⁴.

According to the plot in figure 2, no overfitting is observed while increasing the number of rules per word. Although it seems that the best strategy could be “learn as many rules as possible”, in [7] it is shown that the number of rounds must be determined individually for each word since they have different behaviours. The adjustment of the number of rounds can be done by cross-validation on the training set, as suggested in [1]. However, in our case, this cross-validation inside the cross-validation of the general experiment would generate a prohibitive overhead. Instead, a very simple stopping criterion (sc) has been

⁴ The maximum number of rounds considered is 750, merely for efficiency reasons.

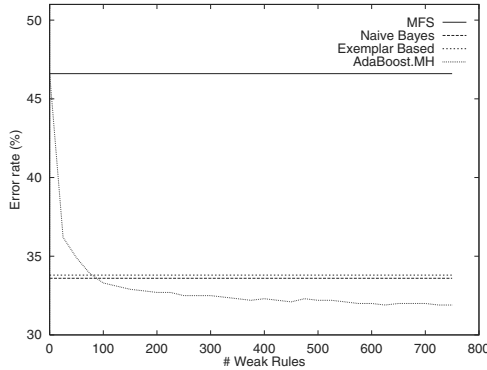


Fig. 2. Error rate of AdaBoost.MH related to the number of weak rules

used, which consists in stopping the acquisition of weak rules whenever the error rate on the training set falls below 5%, with an upper bound of 750 rules. This variant, which is referred to as AB_{sc} , obtained comparable results to AB_{750} but generating only 370.2 weak rules per word on average, which represents a very moderate storage requirement for the combined classifiers.

The numerical information corresponding to this experiment is included in table 1. This table shows the accuracy results, detailed for each word, of NB, EB_1 , EB_{15} , AB_{750} , and AB_{sc} . The best result for each word is printed in boldface.

As it can be seen, in 14 out of 15 cases, the best results correspond to the boosting algorithms. When comparing global results, accuracies of either AB_{750} or AB_{sc} are significantly greater than those of any of the other methods. Finally, note that accuracies corresponding to NB and EB_{15} are comparable (as suggested in [19]), and that the use of k 's greater than 1 is crucial for making Exemplar-based learning competitive on WSD.

4 Making Boosting Practical for WSD

Up to now, it has been seen that AdaBoost.MH is a simple and competitive algorithm for the WSD task. It achieves an accuracy performance superior to that of the Naive Bayes and Exemplar-based algorithms tested in this paper. However, AdaBoost.MH has the drawback of its computational cost, which makes the algorithm not scale properly to real WSD domains of thousands of words.

The space and time-per-round requirements of AdaBoost.MH are $\mathcal{O}(mk)$ (recall that m is the number of training examples and k the number of senses), not including the call to the weak learner. This cost is unavoidable since AdaBoost.MH is inherently sequential. That is, in order to learn the $(t+1)$ -th weak rule it needs the calculation of the t -th weak rule, which properly updates the matrix D_t . Further, inside the **WeakLearner**, there is another iterative process that examines, one by one, all attributes so as to decide which is the one that

Table 1. Set of 15 reference words and results of the main algorithms

Word	POS	Number of			Accuracy (%)					
		Senses	Examp.	Attrib.	MFS	NB	EB ₁	EB ₁₅	AB ₇₅₀	AB _{sc}
age	n	4	493	1662	62.1	73.8	71.4	71.0	74.7	74.0
art	n	5	405	1557	46.7	54.8	44.2	58.3	57.5	62.2
car	n	5	1381	4700	95.1	95.4	91.3	95.8	96.8	96.5
child	n	4	1068	3695	80.9	86.8	82.3	89.5	92.8	92.2
church	n	4	373	1420	61.1	62.7	61.9	63.0	66.2	64.9
cost	n	3	1500	4591	87.3	86.7	81.1	87.7	87.1	87.8
fall	v	19	1500	5063	70.1	76.5	73.3	79.0	81.1	80.6
head	n	14	870	2502	36.9	76.9	70.0	76.9	79.0	79.0
interest	n	7	1500	4521	45.1	64.5	58.3	63.3	65.4	65.1
know	v	8	1500	3965	34.9	47.3	42.2	46.7	48.7	48.7
line	n	26	1342	4387	21.9	51.9	46.1	49.7	54.8	54.5
set	v	19	1311	4396	36.9	55.8	43.9	54.8	55.8	55.8
speak	v	5	517	1873	69.1	74.3	64.6	73.7	72.2	73.3
take	v	30	1500	5181	35.6	44.8	39.3	46.1	46.7	46.1
work	n	7	1469	4923	31.7	51.9	42.5	47.2	50.7	50.7
Avg. nouns		8.6	1040.1	3978.5	57.4	71.7	65.8	71.1	73.5	73.4
verbs		17.9	1265.6	4431.9	46.6	57.6	51.1	58.1	59.3	59.1
all		12.1	1115.3	4150.0	53.3	66.4	60.2	66.2	68.1	68.0

minimizes Z_t . Since there are thousands of attributes, this is also a time consuming part, which can be straightforwardly spedup either by reducing the number of attributes or by relaxing the need to examine all attributes at each iteration.

4.1 Accelerating the WeakLearner

Four methods have been tested in order to reduce the cost of searching for weak rules. The first three, consisting in aggressively reducing the feature space, are frequently applied in Text Categorization. The fourth consists in reducing the number of attributes that are examined at each round of the boosting algorithm.

Frequency filtering (Freq): This method consists in simply discarding those features corresponding to events that occur less than N times in the training corpus. The idea beyond that criterion is that frequent events are more informative than rare ones.

Local frequency filtering (LFreq): This method works similarly to Freq but considers the frequency of events locally, at the sense level. More particularly, it selects the N most frequent features of each sense.

RLM ranking: This third method consists in making a ranking of all attributes according to the RLM distance measure [13] and selecting the N most relevant features. This measure has been commonly used for attribute selection in decision tree induction algorithms⁵.

⁵ RLM distance belongs to the distance-based and information-based families of attribute selection functions. It has been selected because it showed better perfor-

LazyBoosting: The last method does not filter out any attribute but reduces the number of those that are examined at each iteration of the boosting algorithm. More specifically, a small proportion p of attributes are randomly selected and the best weak rule is selected among them. The idea behind this method is that if the proportion p is not too small, probably a sufficiently good rule can be found at each iteration. Besides, the chance for a good rule to appear in the whole learning process is very high. Another important characteristic is that no attribute needs to be discarded and so we avoid the risk of eliminating relevant attributes⁶.

The four methods above have been compared for the set of 15 reference words. Figure 3 contains the average error-rate curves obtained by the four variants at increasing levels of attribute reduction. The top horizontal line corresponds to the MFS error rate, while the bottom horizontal line stands for the error rate of AdaBoost.MH working with all attributes. The results contained in figure 3 are calculated running the boosting algorithm 250 rounds for each word.

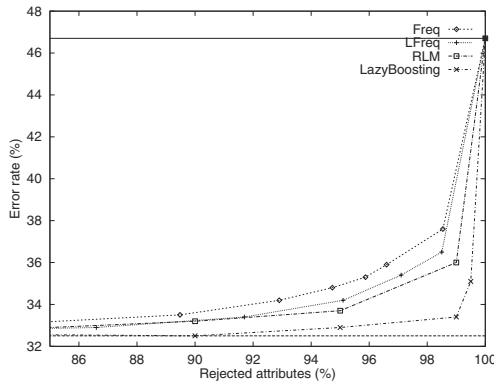


Fig. 3. Error rate obtained by the four methods, at 250 weak rules per word, with respect to the percentage of rejected attributes

The main conclusions that can be drawn are the following:

- All methods seem to work quite well since no important degradation is observed in performance for values lower than 95% in rejected attributes. This may indicate that there are many irrelevant or highly dependent attributes in our domain.

mance than seven other alternatives in an experiment of decision tree induction for PoS tagging [14].

⁶ This method will be called **LazyBoosting** in reference to the work by Samuel and colleagues [24]. They applied the same technique for accelerating the learning algorithm in a Dialogue Act tagging system.

- LFreq is slightly better than Freq, indicating a preference to make frequency counts for each sense rather than globally.
- The more informed RLM ranking performs better than frequency-based reduction methods Freq and LFreq.
- LazyBoosting is better than all other methods, confirming our expectations: it is worth keeping all information provided by the features. In this case, acceptable performance is obtained even if only 1% of the attributes is explored when looking for a weak rule. The value of 10%, for which LazyBoosting still achieves the same performance and runs about 7 times faster than AdaBoost.MH working with all attributes, will be selected for the experiments in section 5.

5 Evaluating LazyBoosting

The LazyBoosting algorithm has been tested on the full semantically annotated corpus with $p = 10\%$ and the same stopping criterion described in section 3.3, which will be referred to as AB_{l10sc} . The average number of senses is 7.2 for nouns, 12.6 for verbs, and 9.2 overall. The average number of training examples is 933.9 for nouns, 938.7 for verbs, and 935.6 overall.

The AB_{l10sc} algorithm learned an average of 381.1 rules per word, and took about 4 days of CPU time to complete⁷. It has to be noted that this time includes the cross-validation overhead. Eliminating it, it is estimated that 4 CPU days would be the necessary time for acquiring a word sense disambiguation boosting-based system covering about 2,000 words.

The AB_{l10sc} has been compared again to the benchmark algorithms using the 10-fold cross-validation methodology described in section 3.2. The average accuracy results are reported in the left hand-side of table 2. The best figures correspond to the LazyBoosting algorithm AB_{l10sc} , and again, the differences are statistically significant using the 10-fold cross-validation paired t -test.

Table 2. Results of LazyBoosting and the benchmark methods on the 191-word corpus

	Accuracy (%)				Wins-Ties-Losses	
	MFS	NB	EB ₁₅	AB_{l10sc}	AB_{l10sc} vs. NB	AB_{l10sc} vs. EB ₁₅
Nouns (121)	56.4	68.7	68.0	70.8	99(51)–1–21(3)	100(68)–5–16(1)
Verbs (70)	46.7	64.8	64.9	67.5	63(35)–1–6(2)	64(39)–2–4(0)
Average (191)	52.3	67.1	66.7	69.5	162(86)–2–27(5)	164(107)–7–20(1)

The right hand-side of the table shows the comparison of AB_{l10sc} versus NB and EB₁₅ algorithms, respectively. Each cell contains the number of wins,

⁷ The current implementation is written in PERL-5.003 and it was run on a SUN UltraSparc2 machine with 194Mb of RAM.

ties, and losses of competing algorithms. The counts of statistically significant differences are included in brackets. It is important to point out that EB_{15} only beats significantly $\text{AB}_{l_{10sc}}$ in one case while NB does so in five cases. Conversely, a significant superiority of $\text{AB}_{l_{10sc}}$ over EB_{15} and NB is observed in 107 and 86 cases, respectively.

6 Conclusions and Future Work

In the present work, Schapire and Singer’s AdaBoost.MH algorithm has been evaluated on the word sense disambiguation task, which is one of the hardest open problems in Natural Language Processing. As it has been shown, the boosting approach outperforms Naive Bayes and Exemplar-based learning, which represent state-of-the-art accuracy on supervised WSD. In addition, a faster variant has been suggested and tested, which is called **LazyBoosting**. This variant allows the scaling of the algorithm to broad-coverage real WSD domains, and is as accurate as AdaBoost.MH. Further details can be found in an extended version of this paper [7].

Future work is planned to be done in the following directions:

- Extensively evaluate AdaBoost.MH on the WSD task. This would include taking into account additional attributes, and testing the algorithms in other manually annotated corpora, and especially on sense-tagged corpora automatically obtained from Internet.
- Confirm the validity of the **LazyBoosting** approach on other language learning tasks in which AdaBoost.MH works well, e.g.: Text Categorization.
- It is known that mislabelled examples resulting from annotation errors tend to be hard examples to classify correctly, and, therefore, tend to have large weights in the final distribution. This observation allows both to identify the noisy examples and use boosting as a way to improve data quality [26,1]. It is suspected that the corpus used in the current work is very noisy, so it could be worth using boosting to try and improve it.

References

1. Abney, S., Schapire, R.E. and Singer, Y.: *Boosting Applied to Tagging and PP-attachment*. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999. 130, 132, 134, 139
2. Bauer, E. and Kohavi, R.: *An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants*. Machine Learning Journal. Special issue on IMLM for Improving and Scaling Machine Learning Algorithms, 1999. 130
3. Breiman, L.: *Arcing Classifiers*. The Annals of Statistics, 26(3), 1998.
4. Duda, R. O. and Hart, P. E.: *Pattern Classification and Scene Analysis*. Wiley, New York, 1973. 134

5. Dietterich, T.G.: *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization*. Machine Learning (to appear). 130
6. Engelson, S.P. and Dagan, I.: *Minimizing Manual Annotation Cost in Supervised Training from Corpora*. In S. Wermter, E. Riloff and G. Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, LNAI, 1040. Springer, 1996. 130
7. Escudero, G., Màrquez, L. and Rigau, G.: *Boosting Applied to Word Sense Disambiguation*. Technical Report LSI-00-3-R, LSI Department, UPC, 2000. 134, 139
8. Freund, Y. and Schapire, R.E.: *Experiments with a New Boosting Algorithm*. In *Procs. of the 13th International Conference on Machine Learning, ICML, 1996*. 130
9. Freund, Y. and Schapire, R.E.: *A Decision-theoretic Generalization of On-line Learning and an Application to Boosting*. *Journal of Computer and System Sciences*, 55(1), 1997. 130
10. Fujii, A., Inui, K., Tokunaga, T. and Tanaka, H.: *Selective Sampling for Example-based Word Sense Disambiguation*. *Computational Linguistics*, 24(4), ACL, 1998. 130
11. Ide, N. and Véronis, J.: *Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art*. *Computational Linguistics*, 24(1), ACL, 1998. 129
12. Leacock, C., Chodorow, M. and Miller, G.A.: *Using Corpus Statistics and Word-Net Relations for Sense Identification*. *Computational Linguistics*, 24(1), ACL, 1998. 130
13. López de Mántaras, R.: *A Distance-based Attribute Selection Measure for Decision Tree Induction*. *Machine Learning*, 6(1), 1991. 136
14. Màrquez, L.: *Part-of-speech Tagging: A Machine Learning Approach based on Decision Trees*. Phd. Thesis, LSI Department, UPC, 1999. 137
15. Mihalcea, R. and Moldovan, I.: *An Automatic Method for Generating Sense Tagged Corpora*. In *Proceedings of the 16th National Conference on Artificial Intelligence, AAAI, 1999*. 130
16. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K.: *Five Papers on WordNet*. *Special Issue of International Journal of Lexicography*, 3(4), 1990. 130
17. Mooney, R.J.: *Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning*. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP, 1996*. 130
18. Ng, H.T. and Lee, H.B.: *Integrating Multiple Knowledge Sources to Disambiguate Word Senses: An Exemplar-based Approach*. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL, 1996*. 133
19. Ng, H.T.: *Exemplar-based Word Sense Disambiguation: Some Recent Improvements*. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, EMNLP, 1997*. 130, 134, 135
20. Ng, H.T.: *Getting Serious about Word Sense Disambiguation*. In *Proceedings of the SIGLEX Workshop "Tagging Text with Lexical Semantics: Why, What and How?"*, 1997. 130
21. Ng, H.T., Chung, Y. L. and Shou, K. F.: *A Case Study on Inter-Annotation Agreement for WSD*. In *Proceedings of the SIGLEX Workshop "Standardizing Lexical Resources"*, Maryland, USA, 1999. 133

22. Pedersen, T. and Bruce, R.: *Knowledge Lean Word-Sense Disambiguation*. In Proceedings of the 15th National Conference on Artificial Intelligence, 1998. 130
23. Quinlan, J.R.: *Bagging, Boosting and C4.5*. In Proceedings of the 13th National Conference on Artificial Intelligence, AAAI, 1996. 130
24. Samuel, K.: *Lazy Transformation-Based Learning*. In Proceedings of the 11th International Florida AI Research Symposium Conference, 1998. 137
25. Schapire, R.E. and Singer, Y.: *Improved Boosting Algorithms Using Confidence-rated Predictions*. Machine Learning (to appear). 130, 131, 132, 133
26. Schapire, R.E. and Singer, Y.: *Booster: A Boosting-based System for Text Categorization*. Machine Learning (to appear). 130, 131, 133, 139
27. Towell, G. and Voorhees, E.M.: *Disambiguating Highly Ambiguous Words*. Computational Linguistics, 24(1), ACL, 1998. 130
28. Yarowsky, D.: *Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French*. In Proceedings of the 32nd annual Meeting of the Association for Computational Linguistics, ACL, 1994. 130

A Multiple Model Cost-Sensitive Approach for Intrusion Detection

Wei Fan¹, Wenke Lee², Salvatore J. Stolfo¹, and Matthew Miller¹

¹ Department of Computer Science, Columbia University
1214 Amsterdam Avenue Room 450, New York, NY 10027-7003, USA
`{wfan,sal,mmiller}@cs.columbia.edu`

² Department of Computer Science, North Carolina State University
Raleigh, NC 27695-7550, USA
`wenke@csc.ncsu.edu`

Abstract. Intrusion detection systems (IDSs) need to maximize security while minimizing costs. In this paper, we study the problem of building cost-sensitive intrusion detection models to be used for real-time detection. We briefly discuss the major cost factors in IDS, including consequential and operational costs. We propose a multiple model cost-sensitive machine learning technique to produce models that are optimized for user-defined cost metrics. Empirical experiments in off-line analysis show a reduction of approximately 97% in operational cost over a single model approach, and a reduction of approximately 30% in consequential cost over a pure accuracy-based approach.

1 Introduction

Intrusion Detection (ID) is an important component of infrastructure protection mechanisms. Many intrusion detection systems (IDSs) are emerging in the market place, following research and development efforts in the past two decades. They are, however, far from the ideal security solutions for customers. Investment in IDSs should bring the highest possible benefit and maximize user-defined security goals while minimizing costs. This requires ID models to be sensitive to cost factors. Currently these cost factors are ignored as unwanted complexities in the development process of IDSs.

We developed a data mining framework for building intrusion detection models. It uses data mining algorithms to compute activity patterns and extract predictive features, and applies machine learning algorithms to generate detection rules [7]. In this paper, we report the initial results of our current research in extending our data mining framework to build cost-sensitive models for intrusion detection. We briefly examine the relevant cost factors, models and metrics related to IDSs. We propose a multiple model cost-sensitive machine learning technique that can automatically construct detection models optimized for given cost metrics. Our models are learned from training data which was acquired from an environment similar to one in which a real-time detection tool may be deployed. Our data consists of network connection records processed from

raw `tcpdump` [5] files using MADAM ID (a system for Mining Audit Data for Automated Models for Intrusion Detection) [7].

The rest of the paper is organized as follows: Section 2 examines major cost factors related to IDSs and outlines problems inherent in modeling and measuring the relationships among these factors. Section 3 describes our multiple model approach to reducing operational cost and a MetaCost [3] procedure for reducing damage cost and response cost. In Section 4, we evaluate this proposed approach using the 1998 DARPA Intrusion Detection Evaluation dataset. Section 5 reviews related work in cost-sensitive learning and discusses extensions of our approach to other domains and machine learning algorithms. Section 6 offers conclusive remarks and discusses areas of future work.

2 Cost Factors, Models, and Metrics in IDSs

2.1 Cost Factors

There are three major cost factors involved in the deployment of an IDS. Damage cost, $DCost$, characterizes the maximum amount of damage inflicted by an attack when intrusion detection is unavailable or completely ineffective. Response cost, $RCost$, is the cost to take action when a potential intrusion is detected. Consequential cost, $CCost$, is the total cost caused by a connection and includes $DCost$ and $RCost$ as described in detail in Section 2.2. The operational cost, $OpCost$, is the cost inherent in running an IDS.

2.2 Cost Models

The cost model of an IDS formulates the total expected cost of the IDS. In this paper, we consider a simple approach in which a prediction made by a given model will always result in some action being taken. We examine the cumulative cost associated with each of these outcomes: false negative (FN), false positive (FP), true positive (TP), true negative (TN), and misclassified hits. These costs are known as *consequential costs* ($CCost$), and are outlined in Table 1.

FN Cost is the cost of not detecting an intrusion. It is therefore defined as the damage cost associated with the particular type of intrusion i_t , $DCost(i_t)$.

TP Cost is the cost incurred when an intrusion is detected and some action is taken. We assume that the IDS acts quickly enough to prevent the damage of the detected intrusion, and therefore only pay $RCost(i_t)$.

FP Cost is the cost incurred when an IDS falsely classifies a normal connection as intrusive. In this case, a response will ensue and we therefore pay $RCost(i)$, where i is the detected intrusion.

TN Cost is always 0, as we are not penalized for correct normal classification.

Misclassified Hit Cost is the cost incurred when one intrusion is incorrectly classified as a different intrusion – when i is detected instead of i_t . We take a pessimistic approach that our action will not prevent the damage of the intrusion at all. Since this simplified model assumes that we always respond to a predicted intrusion, we also include the response cost of the detected intrusion, $RCost(i)$.

Table 1. Consequential Cost (CCost) Matrix

Outcome	CCost(c)
Miss (FN)	DCost(i_t)
False Alarm (FP)	RCost(i)
Hit (TP)	RCost(i_t)
Normal (TN)	0
Misclassified Hit	RCost(i) + DCost(i_t)
c : connection, i_t : true class, i : predicted class	

2.3 Cost Metrics

Cost-sensitive models can only be constructed and evaluated using given cost metrics. Qualitative analysis is applied to measure the relative magnitudes of the cost factors, as it is difficult to reduce all factors to a common unit of measurement (such as dollars). We have thus chosen to measure and minimize CCost and OpCost in two orthogonal dimensions.

An intrusion taxonomy must be used to determine the damage and response cost metrics which are used in the formulation of CCost. A more detailed study of these cost metrics can be found in our on-going work [8]. Our taxonomy is the same as that used in the DARPA evaluation, and consists of four types of intrusions: probing (PRB), denial of service (DOS), remotely gaining illegal local access (R2L), and a user gaining illegal root access (U2R). All attacks in the same category are assumed to have the same DCost and RCost. The relative scale or metrics chosen are shown in Table 2a.

Table 2. Cost Metrics of Intrusion Classes and Feature Categories

Category	DCost	RCost	Category	OpCost
U2R	100	40	Level 1	1 or 5
R2L	50	40	Level 2	10
DOS	20	20	Level 3	100
PRB	2	20		
normal	0	0		

(a)
(b)

The operational cost of running an IDS is derived from an analysis of the computational cost of computing the features required for evaluating classification rules. Based on this computational cost and the added complexity of extracting and constructing predictive features from network audit data, features are categorized into three relative levels. Level 1 features are computed using at most the first three packets of a connection. Level 2 features are computed in the middle of or near the end of a connection using information of the current connection only. Level 3 features are computed using information from all connections within a

given time window of the current connection. Relative magnitudes are assigned to these features to represent the different computational costs as measured in a prototype system we have developed using NFR [10]. These costs are shown in Table 2b. The cost metrics chosen incorporate the computational cost as well as the availability delay of these features. It is important to note that level 1 and level 2 features must be computed individually. However, because all level 3 features require iteration through the entire set of connections in a given time window, all level 3 features can be computed at the same time, in a single iteration. This saves operational cost when multiple level 3 features are computed for analysis of a given connection.

3 Cost-Sensitive Modeling

In the previous section, we discussed the consequential and operational costs involved in deploying an IDS. We now explain our cost-sensitive machine learning methods for reducing these costs.

3.1 Reducing Operational Cost

In order to reduce the operational cost of an IDS, the detection rules need to use low cost features as often as possible while maintaining a desired accuracy level. Our approach is to build multiple rulesets, each of which uses features from different cost levels. Low cost rules are always evaluated first by the IDS, and high cost rules are used only when low cost rules can not predict with sufficient accuracy. We propose a multiple ruleset approach based on RIPPER, a popular rule induction algorithm [2].

Before discussing the details of our approach, it is necessary to outline the advantages and disadvantages of two major forms of rulesets that RIPPER computes, *ordered* and *un-ordered*. An ordered ruleset has the form **if** $rule_1$ **then** $intrusion_1$ **elseif** $rule_2$ **then** $intrusion_2$, ..., **else** *normal*. To generate an ordered ruleset, RIPPER sorts class labels according to their frequency in the training data. The first rule classifies the most infrequent class, and the end of the ruleset signifies prediction of the most frequent (or default) class, *normal*, for all previously unpredicted instances. An ordered ruleset is usually succinct and efficient, and there is no rule generated for the most frequent class. Evaluation of an entire ordered ruleset does not require each rule to be tested, but proceeds from the top of the ruleset to the bottom until any rule evaluates to *true*. The features used by each rule can be computed one by one as evaluation proceeds. An un-ordered ruleset, on the other hand, has at least one rule for each class and there are usually many rules for frequently occurring classes. There is also a default class which is used for prediction when none of these rules are satisfied. Unlike ordered rulesets, all rules are evaluated during prediction and all features used in the ruleset must be computed before evaluation. Ties are broken by using the most accurate rule. Un-ordered rulesets are less efficient in execution, but there are usually several rules of varying precision for the most

frequent class, *normal*. Some of these *normal* rules are usually more accurate than the default rule for the equivalent ordered ruleset.

With the advantages and disadvantages of ordered and un-ordered rulesets in mind, we propose the following multiple ruleset approach:

- We first generate multiple training sets T_{1-4} using different feature subsets. T_1 uses only cost 1 features. T_2 uses features of costs 1 and 5, and so forth, up to T_4 , which uses all available features.
- Rulesets R_{1-4} are learned using their respective training sets. R_4 is learned as an ordered ruleset for its efficiency, as it may contain the most costly features. R_{1-3} are learned as un-ordered rulesets, as they will contain accurate rules for classifying *normal* connections.
- A *precision* measurement p_r ¹ is computed for *every* rule, r , except for the rules in R_4 .
- A threshold value τ_i is obtained for every single class, and determines the tolerable precision required in order for a classification to be made by any ruleset except for R_4 .

In real-time execution, the feature computation and rule evaluation proceed as follows:

- All cost 1 features used in R_1 are computed for the connection being examined. R_1 is then evaluated and a prediction i is made.
- If $p_r > \tau_i$, the prediction i will be fired. In this case, no more features will be computed and the system will examine the next connection. Otherwise, additional features required by R_2 are computed and R_2 will be evaluated in the same manner as R_1 .
- Evaluation will continue with R_3 , followed by R_4 , until a prediction is made.
- When R_4 (an ordered ruleset) is reached, it computes features as needed while evaluation proceeds from the top of the ruleset to the bottom. The evaluation of R_4 does not require any firing condition and will always generate a prediction.

The OpCost for a connection is the total computational cost of all unique features used before a prediction is made. If any level 3 features (of cost 100) are used at all, the cost is counted only once since all level 3 features are calculated in one function call.

This evaluation scheme is further motivation for our choice of learning R_{1-3} as un-ordered rulesets. If R_{1-3} were learned as ordered rulesets, a *normal* connection could not be predicted until R_4 since the default *normal* rules of these rulesets would be less accurate than the default rule of R_4 . OpCost is thus reduced, resulting in greater system throughput, by only using low cost features to predict normal connections.

¹ Precision describes how accurate a prediction is. Precision is defined as $p = \frac{|P \cap W|}{|P|}$, where P is the set of predictions with label i , and W is the set of all instances with label i in the data set.

The precision and threshold values can be obtained during model training from either the training set or a separate hold-out validation set. Threshold values are set to the precisions of R_4 on that dataset. Precision of a rule can be obtained easily from the positive, p , and negative, n , counts of a rule, $\frac{p}{p+n}$. The threshold value will, on average, ensure that the predictions emitted by the first three rulesets are not less accurate than using R_4 as the only hypothesis.

3.2 Reducing Consequential Cost

The MetaCost algorithm, introduced by Domingos [3], has been applied to reduce CCost. MetaCost re-labels the training set according to the cost-matrix and decision boundaries of RIPPER. Instances of intrusions with $DCost(i) < RCost(i)$ or a low probability of being learned correctly will be re-labeled as *normal*.

Table 3. Intrusions, Categories and Sampling

U2R		R2L		DOS		PRB	
buffer_overflow	1	ftp_write	4	back	1	ipsweep	1
loadmodule	2	guess_passwd	1	land	1	nmap	1
multihop	6	imap	2	neptune	$\frac{1}{20}$	portsweep	1
perl	6	phf	3	pod	1	satan	1
rootkit	2	spy	8	smurf	$\frac{1}{20}$		
		warezclient	1	teardrop	1		
		warezmaster	1				

4 Experiments

4.1 Design

Our experiments use data that were distributed by the 1998 DARPA evaluation, which was conducted by MIT Lincoln Lab. The data were gathered from a military network with a wide variety of intrusions injected into the network over a period of 7 weeks. The data were then processed into connection records using MADAM ID. The processed records are available from the UCI KDD repository as the 1999 KDD Cup Dataset [11]. A 10% sample was taken which maintained the same distribution of intrusions and normal connections as the original data.² We used 80% of this sample as training data. For infrequent intrusions in the training data, those connections were repeatedly injected to prevent the learning algorithm from neglecting them as statistically insignificant and not generating rules for them. For overwhelmingly frequent intrusions, only 1 out of 20 records

² The full dataset is around 743M. It is very difficult to process and learn over the complete dataset in a reasonable amount of time with limited resources given the fact that RIPPER is memory-based and MetaCost must learn multiple bagging models to estimate probabilities.

were included in the training data. This is an ad hoc approach, but produced a reasonable ruleset. The remaining 20% of our sample data were left unaltered and used as test data for evaluation of learned models. Table 3 shows the different intrusions present in the data, the category within our taxonomy that each belongs to, and their sampling rates in the training data.

We used the training set to calculate the precision for each rule and the threshold value for each class label. We experimented with the use of a hold-out validation set to calculate precisions and thresholds. The results (not shown) are similar to those reported below.

4.2 Measurements

We measure expected operational and consequential costs in our experiments. The expected OpCost over all occurrences of each connection class and the average OpCost per connection over the entire test set are defined as $\frac{\sum_{c \in S_i} OpCost(c)}{|S_i|}$ and $\frac{\sum_{c \in S} OpCost(c)}{|S|}$, respectively, where S is the entire test set, i is a connection class, and S_i represents all occurrences of i in S . In all of our reported results, $OpCost(c)$ is computed as the sum of the feature computation costs of all unique features used by all rules evaluated until a prediction is made for connection c . CCost is computed as the cumulative sum of the cost matrix entries, defined in Table 1, for all predictions made over the test set.

4.3 Results

In all discussion of our results, including all tables, “RIPPER” is the single model learned over the original dataset, “Multi-RIPPER” is the respective multiple model, “MetaCost” is the single model learned using RIPPER with a MetaCost re-labeled dataset, and “Multi-MetaCost” is the respective multiple model.

As shown in Table 5, the average OpCost per connection of the single MetaCost model is 191, while the Multi-MetaCost model has an average OpCost of 5.78. This is equivalent to the cost of computing only a few level 1 features per connection and offers a reduction of 97% from the single ruleset approach. The single MetaCost model is 33 times more expensive. This means that in practice we can classify most connections by examining the first three packets of the connection at most 6 times. Additional comparison shows that the average OpCost of the Multi-RIPPER model is approximately half as much as that of the single RIPPER model. This significant reduction by Multi-MetaCost is due to the fact that R_{1-3} accurately filter *normal* connections (including low-cost intrusions re-labeled as *normal*), and a majority of connections in real network environments are *normal*. Our multiple model approach thus computes more costly features only when they are needed to detect intrusions with $DCost > RCost$. Table 4 lists the detailed average OpCost for each connection class. It is important to note that the difference in OpCost between RIPPER and MetaCost models is explainable by the fact that MetaCost models do not contain (possibly costly) rules to classify intrusions with $DCost < RCost$.

Table 4. Average OpCost per Connection Class

IDS	RIPPER	Multi-RIPPER	MetaCost	Multi-MetaCost
back	223	143	191	1
buffer_overflow	172	125.8	175	91.6
ftp_write	172	113	146	71.25
guess_passwd	198.36	143	191	87
imap	172	107.17	181	108.08
ipsweep	222.98	100.17	191	1
land	132	2	191	1
loadmodule	155.33	104.78	168.78	87
multihop	183.43	118.43	182.43	100.14
neptune	223	100	191	1
nmap	217	119.63	191	1
normal	222.99	111.14	190.99	4.99
perl	142	143	151	87
phf	21	143	191	1
pod	223	23	191	1
portsweep	223	117.721	191	1
rootkit	162	100.7	155	63.5
satan	223	102.84	191	1
smurf	223	143	191	1
spy	131	100	191	46.5
teardrop	223	23	191	1
warezclient	223	140.72	191	86.98
warezmaster	89.4	48.6	191	87

Table 5. Average OpCost per Connection

	RIPPER	Multi-RIPPER	MetaCost	Multi-MetaCost
OpCost	222.73	110.64	190.93	5.78

Table 6. CCost and Error Rate

	RIPPER	Multi-RIPPER	MetaCost	Multi-MetaCost
CCost	42026	41850	29866	28026
Error	0.0847%	0.1318%	8.24%	7.23%

Table 7. Precision and Recall for Each Connection Class

		RIPPER	Multi- RIPPER	MetaCost	Multi- MetaCost
back	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
buffer_overflow	TP	1.0	1.0	0.8	0.6
	p	1.0	1.0	0.67	0.75
ftp_write	TP	1.0	0.88	0.25	0.25
	p	1.0	1.0	1.0	1.0
guess_passwd	TP	0.91	0.91	0.0	0.0
	p	1.0	1.0	na	na
imap	TP	1.0	0.83	1.0	0.92
	p	1.0	1.0	1.0	1.0
ipsweep	TP	0.99	0.99	0.0	0.0
	p	1.0	1.0	na	na
land	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
load_module	TP	1.0	1.0	0.44	0.67
	p	0.9	1.0	1.0	1.0
multihop	TP	1.0	1.0	1.0	0.86
	p	0.88	0.88	0.88	1.0
neptune	TP	1.0	1.0	na	na
	p	1.0	1.0	na	na
nmap	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
normal	TP	0.99	0.99	0.99	0.99
	p	0.99	0.99	0.92	0.93
perl	TP	1.0	1.0	1.0	1.0
	p	1.0	1.0	1.0	1.0
phf	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
pod	TP	1.0	1.0	0.0	0.0
	p	0.98	0.98	na	na
portsweep	TP	0.99	0.99	0.0	0.0
	p	1.0	1.0	na	na
rootkit	TP	1.0	0.6	0.5	0.2
	p	0.77	1.0	0.83	1.0
satan	TP	1.0	0.98	0.0	0.0
	p	0.99	0.99	na	na
smurf	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
spy	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
teardrop	TP	1.0	1.0	0.0	0.0
	p	1.0	1.0	na	na
warezclient	TP	0.99	0.99	0.0	0.9
	p	1.0	1.0	na	1.0
warezmaster	TP	0.6	0.6	0.0	0.0
	p	1.0	1.0	na	na

Table 8. Comparison with fcs-RIPPER

	Multi- MetaCost	MetaCost	fcs-RIPPER									
			$\omega = .1$	.2	.3	.4	.5	.6	.7	.8	.9	1.0
OpCost	5.78	191	151	171	191	181	181	161	161	171	171	171

Our CCost measurements are shown in Table 6. As expected, both MetaCost and Multi-MetaCost models yield a significant reduction in CCost over RIPPER and Multi-RIPPER models. These reductions are both approximately 30%. The consequential costs of the Multi-MetaCost and Multi-RIPPER models are also slightly lower than those of the single MetaCost and RIPPER models.

The detailed precision and TP³ rates of all four models are shown in Table 7 for different connection classes. The values for the single classifier and multiple classifier methods are very close to each other. This shows that the coverages of the multiple classifier methods are identical to those of the respective single classifier methods. It is interesting to point out that MetaCost fails to detect *warezclient*, but Multi-MetaCost is highly accurate. The reason is that R_4 completely ignores all occurrences of *warezclient* and classifies them as *normal*.

The error rates of all four models are also shown in Table 6. The error rates of MetaCost and Multi-MetaCost are much higher than those of RIPPER and Multi-RIPPER. This is because many intrusions with $DCost < RCost$ are re-labeled as *normal* by the MetaCost procedure. Multi-RIPPER misclassified such intrusions more often than RIPPER, which results in its slightly lower CCost and slightly higher error rate. Multi-MetaCost classifies more intrusions correctly (*warezclient*, for example) and has a lower CCost and error rate than MetaCost.

4.4 Comparison with fcs-RIPPER

In previous work, we introduced a feature cost-sensitive method, “fcs-RIPPER”, to reduce OpCost [8,9]. This method favors less costly features when constructing a ruleset. Cost sensitivity is controlled by the variable $\omega \in [0, 1]$ and sensitivity increases with the value of ω . We generated a single ordered ruleset using different values of ω with fcs-RIPPER. In Table 8, we compare the average OpCost over the entire test set for the proposed multiple classifier method with that of fcs-RIPPER. We see that fcs-RIPPER reduces the operational cost by approximately 10%, whereas Multi-MetaCost reduces this value by approximately 97%. The expected cost of Multi-MetaCost is approximately 30 times lower than that of fcs-RIPPER, RIPPER, and MetaCost. This difference is significant.

5 Related Work

Much research has been done in cost-sensitive learning, as indicated by Turney’s online bibliography [13]. Within the subset of this research which focuses on multiple models, Chan and Stolfo proposed a meta-learning approach to reduce consequential cost in credit card fraud detection [1]. MetaCost is another approach which uses bagging to estimate probabilities. Fan et al. proposed a

³ Unlike precision, TP rate describes the fraction of occurrences of a connection class that were correctly labeled. Using the same notation as in the definition of precision, $TP = \frac{P \cap W}{W}$.

variant of AdaBoost for misclassification cost-sensitive learning [4]. Within research on feature-cost-sensitive learning, Lavrac et al. applied a hybrid genetic algorithm effective for feature elimination [6].

Credit card fraud detection, cellular phone fraud detection and medical diagnosis are related to intrusion detection because they deal with detecting abnormal behavior, are motivated by cost-saving, and thus use cost-sensitive modeling techniques. Our multiple model approach is not limited to IDSs and is applicable in these domains as well.

In our study, we chose to use an inductive rule learner, RIPPER. However, the multiple model approach is not restricted to this learning method and can be applied to any algorithm that outputs a precision along with its prediction.

6 Conclusion and Future Work

Our results using a multiple model approach on off-line network traffic analysis show significant improvements in both operational cost (a reduction of 97% over a single monolithic model) and consequential costs (a reduction of 30% over accuracy-based model). The operational cost of our proposed multiple model approach is significantly lower than that of our previously proposed fcs-RIPPER approach. However, it is desirable to implement this multiple model approach in a real-time IDS to get a practical measure of its performance. Since the average operational cost is close to computing at most 6 level 1 features, we expect efficient real-time performance. The moral of the story is that computing a number of specialized models that are accurate and cost-effective for particular subclasses is demonstrably better than building one monolithic ID model.

6.1 Future Work

It was noted in Section 2.2 that we only consider the case where a prediction made by a given model will always result in an action being taken. We have performed initial investigation into the utility of using an additional decision module to determine whether action is necessary based upon whether $DCost > RCost$ for the predicted intrusion. Such a method would allow for customizable cost matrices to be used, but may result in higher OpCost, as the learned model would make cost-insensitive predictions.

In off-line experiments, rulesets are evaluated using formatted connection records such that rulesets are evaluated after all connections have terminated. In real-time execution of ID models, a major consideration is to evaluate rulesets as soon as possible for timely detection and response. In other words, we need to minimize the detection delay. To achieve this, we can first translate each of the rulesets produced by our multiple model approach, each using different levels of features, into multiple modules of a real-time IDS. Since features of different levels are available and computed at different stages of a connection, we can evaluate our multiple models in the following manner: as the first packets arrive, level 1 features are computed and R_1 rules are evaluated; if a rule evaluates to

true and that rule has sufficient precision, then no other checking for the connection is done. Otherwise, as the connection proceeds, either on a per-packet basis or multi-packet basis, level 2 features are computed and R_2 rules are evaluated. This process will continue through the evaluation of R_4 until a prediction is made. Our current single model approach computes features and evaluates rulesets at the end of a connection. It is thus apparent that this multiple model approach will significantly reduce the detection delay associated with the single model approach. However, it remains to be seen whether additional operational cost will be incurred because we must trigger the computation of various features at different points throughout a connection. We plan to experiment in the real-time evaluation of our multiple model approach using both NFR and Bro [12], two network monitoring tools used for real-time intrusion detection.

References

1. P. Chan and S. Stolfo. Towards scalable learning with non-uniform class and cost distribution: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, August 1999. 151
2. W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann. 145
3. P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proc. of the Fifth ACM SIGKDD Int'l. Conf. on Knowledge Discovery & Data Mining*, pages 155–164, San Diego, CA, 1999. ACM. 143, 147
4. W. Fan, S. Stolfo, J. X. Zhang, and P. K. Chan. Adacost: misclassification cost-sensitive learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 97–105, Bled, Slovenia, 1999. Morgan Kaufmann. 152
5. V. Jacobson, C. Leres, and S. McCanne. *tcpdump*. available via anonymous ftp to ftp.ee.lbl.gov, June 1989. 143
6. N. Lavrac, D. Gamberger, and P. Turney. Cost-sensitive feature reduction applied to a hybrid genetic algorithm. In *Proceedings of the Seventh International Workshop on Algorithmic Learning Theory*, pages 127–134, Sydney, Australia, 1996. 152
7. W. Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Columbia University, June 1999. 142, 143
8. W. Lee, M. Miller, and S. Stolfo et al. Toward cost-sensitive modeling for intrusion detection. Technical Report CUCS-002-00, Computer Science, Columbia University, 2000. 144, 151
9. M. Miller. Learning cost-sensitive classification rules for network intrusion detection using ripper. Technical Report CUCS-035-99, Computer Science Department, Columbia University, December 1999. 151
10. Network Flight Recorder Inc. Network flight recorder. <http://www.nfr.com/>, 1997. 145
11. University of California at Irvine. UCI KDD archive. <http://kdd.ics.uci.edu/>. 147
12. V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proc. of the Seventh USENIX Security Symposium*, San Antonion, TX, 1998. 153

13. P. Turney. Cost-sensitive learning bibliographies.
<http://ai.iit.nrc.ca/bibliographies/cost-sensitive.html>. 151

Value Miner: A Data Mining Environment for the Calculation of the Customer Lifetime Value with Application to the Automotive Industry

Katja Gelbrich¹ and Reza Nakhaeizadeh²

¹ DaimlerChrysler Research Center, Postfach 2360, D-89013 Ulm, Germany
Rheza.Nakhaeizadeh@daimlerchrysler.com

² DaimlerChrysler Research Center, Postfach 2360, D-89013 Ulm, Germany
gelbrich@rcs.urz.tu-dresden.de

Abstract. The acquisition of new accounts is a major task of marketers. It is often carried out rather unsystematically, though. However, by now, one has come to terms that customer acquisition is a matter of quality. An instrument to evaluate prospective accounts is the Customer Lifetime Value (CLV). This paper introduces a Data Mining environment for its calculation and demonstrates its applicability to marketing in the automotive industry. The Car Miner refers to the evaluation of prospects rather than of current customers. This and other restrictions will be discussed along with guidelines for future research.

1 Introduction

Not all customers contribute to the profit of their suppliers. According to the "pareto-rule", a minority of 20% high-valuable customers subsidizes 80% less valuable ones [12]. Therefore, the acquisition budget should be spent on the right prospects. The Customer Lifetime Value (CLV) supports this decision [4], [9]. *Chapter 2* introduces a definition of CLV and describes constraints concerning the customer acquisition in the automotive industry which leads to an adjusted CLV model. *Chapter 3* reflects the development of a Data Mining environment to calculate the CLV. *Chapter 4* introduces restrictions of the model and discusses ideas to improve the Value Miner.

2 Conceptualization of the Customer Lifetime Value

2.1 Classical Definition of CLV

Several models of CLV are introduced in the literature. They agree that CLV is the *present value of expected revenues less costs caused directly by a customer during his relationship with the seller* [2], [4], [8]. Costs include spending for acquisition (e.g. advertising, promotion) and account maintenance (e.g. post purchase marketing). Revenues include the monetary benefit from the customer, i.e. the money he spends

on the supplier's products / services [8]. Some authors also mention soft benefits, e.g. the customer's reference value [6]. The definition reveals several *key statements*:

1. Since the value of a customer is revenues less costs, it represents a *net* value.
2. The term "lifetime" refers to the *duration of the relationship* between buyer and seller. This requires an estimation of the prospective end of the relationship.
3. Revenues can be economic (e.g. turnover) and *non-economic* (e.g. reference value). Non-economic benefits have to be quantified.
4. The term "present" suggests to *discount* future payments. The implied devaluation of future streams of payment is due to the fact that they are uncertain and that the company could alternatively invest its money into the capital market.
5. The definition only covers revenues and costs *in the future*.

2.2 Adjustment of the Definition with Respect to the Acquisition of Car Buyers

Given the acquisition of new accounts in the automotive industry, only two of the statements above are relevant: Future payments should be discounted to the present, because they are uncertain (statement 4). Past payments from the prospect should be neglected, because – as opposed to current customers – they are not relevant to the company concerned (statement 5). However, the statements 1, 2, and 3 do not hold:

- *Statement 1 (net value)*. As opposed to current customers, the individual costs of a prospect can hardly be estimated, because there is no individual historical data. Subsequently, some researchers [4] simply divide the historical overall spending for the acquisition and retention of customers by the number of accounts yielding to per capita costs. But if one does so, costs can easily be neglected at all, because they reduce the revenue of all prospects by the same amount.
- *Statement 2 (time frame)*. Most authors, e.g. [10], equate the time frame of CLV with the relationship's duration. But the end of a relationship is uncertain. We argue to extend the time frame to the day when the buyer stops consuming, i.e. when he dies. This is reasonable, because one should aim at keeping customers for good.
- *Statement 4 (non-economic benefit)*. Many authors point out the relevance of soft benefits, but – with only a few exceptions [3] – refrain from measuring them, because they can hardly be quantified. If soft facts are included at all, they are more easily gathered from actual customers rather than from prospects. Table 1 summarizes the adjustments of the state-of-the-art definition.

Table 1. Definition of CLV adjusted to the customer acquisition in the automobile industry

Problem	Classical Definition	Adjustment	Reason for Adjustment
Gross or net value?	Net value	Gross value	Costs are assumed to be equal among customers.
Present value?	Present value	No adjustment	–
Time frame?	Estimated duration of the relationship	Remaining lifetime	The goal is to keep the customer for ever.
Soft benefits?	Inclusion of soft benefits	No inclusion of soft benefits	Soft benefits can hardly be estimated for prospects.
Past payments?	No past payments	No adjustment	–

Based on these adjustments, the CLV is conceptualized by discounting the *price acceptance* in every year the customer purchases a car to the presence. Given the restrictions which come along with the evaluation of prospects, this conceptualization is preferred to competing definitions (e.g. CLV as net present value). Cars are not purchased frequently, though. Thus, the term y , representing the year of purchase, does not increase by one, but in smaller steps, called purchase frequency (see equation 1).

$$CLV = \sum_{y=0}^n \frac{PA_y}{(1+r)^y} = PA_0 + \frac{PA_{y+PF}}{(1+r)^{[PF]}} + \frac{PA_{y+2PF}}{(1+r)^{[2 \cdot PF]}} + \dots + \frac{PA_{y+n \cdot PF}}{(1+r)^{[n \cdot PF]}} \quad (1)$$

PA ... Price acceptance r ... Rate of discount PF ... Purchase frequency
y ... Year of purchase n ... Last year of purchase (when customer dies)

According to equation (1), the following information are required: purchase frequency, price acceptance, rate of discount, age of customer, average life expectancy. The main source for the *Data Warehouse* was the "Consumer Analysis 1999" (CA) with data from 31,337 German residents. Since only purchasers of *new cars* were considered, the data bases melted down to 6,039 people. Some data (life expectancy, discount rate) were added from external sources (Federal Statistical Office, FAZ).

3 Data Mining Environment for the Calculation of the CLV

Predicting future revenues and discounting them to the presence seems to be the main Data Mining task for calculating the CLV (chapter 3.2). The calculation, however, is not as trivial as equation (1) suggests: Upcoming arguments necessitated a refinement of the model. First, the purchase frequency is not constant (chapter 3.1). Second, the discount rate is a component of market interest and price increase (chapter 3.3).

3.1 Data Mining Task 1: Prediction of the Purchase Frequency

The purchase frequency was not acquired directly. Thus, it had to be predicted by other variables. We assumed, that the purchase frequency, i.e. the *time span an individual keeps a car*, decreases with income, intensity of care usage, usage for business reasons, and a positive attitude towards brands. Moreover, it was supposed that the frequency is not constant – as equation (1) states –, but that older people purchase less often. In order to predict the purchase frequency, we had to consider *two more items*:

- People were asked how old their current car was (CAR_0).
- They were asked if they intended to buy a new car in the course of the this year (INTENTION). To the people who did, CAR_0 represents their *purchase frequency*.

We draw a *subsample* of those people who declared their upcoming purchase intention ($n = 2,260$) and conducted several analyses with CAR_0 (i.e. purchase frequency) as the dependent and the items stated above as *independent variables*:

- Age of the customer in $t = 0$ (AGE_0),
- Net income of the household (INCOME),

- Intensity of car usage, quantified by the kilometers driven per year (KILO),
- Usage for business or private reasons (PRIVATE),
- Attitude towards the consumption of brands (BRAND).

Cross-validation was used to choose the model which best predicts the purchase frequency. The idea is to split the data base into two subsamples, to calibrate the model on one part and validate it on the other. The model yielding to the highest R square on the validation subsample should be chosen [7]. *Three alternative models* were tested on the calibration sample: a multiple linear regression, a multiple non-linear regression, and a chaid analysis. The *linear regression* yielded to equation 2. As hypothesized, older people and private users purchase less often (PF increases); high income, high intensity of usage, and positive attitude towards brands reduce the time span.

In the multiple *non-linear regression*, the directions of influence were the same, but the dependencies for AGE₀ (cubic), INCOME (exponential), KILO (cubic), and PRIVATE (exponential) were non-linear (see equation 3). The *chaid analysis* searches for the independent variables yielding to the highest split of the dependent variable. Chaid stands for Chi-squared Automatic Interaction Detector pointing out that it is based on chi-square tests automatically detecting interactions between variables [1]. Figure 1 partly displays the chaid output¹.

$$\text{PF} = 5.674 + (0.170 \bullet \text{AGE}_0) - (0.124 \bullet \text{INCOME}) - (0.182 \bullet \text{KILO}) + (0.847 \bullet \text{PRIVATE}) - (0.906 \bullet \text{BRAND}) \quad (2)$$

$$\text{PF} = 5.834 + (-0.075 \bullet \text{AGE}_0) + (0.0017 \bullet \text{AGE}_0^2) + (-0.00000096 \bullet \text{AGE}_0^3) + (e^{-0.109 \bullet \text{INCOME}}) + (-1.556 \bullet \text{KILO}) + (0.429 \bullet \text{KILO}^2) + (-0.040 \bullet \text{KILO}^3) + (e^{0.647 \bullet \text{PRIVATE}}) - (0.961 \bullet \text{BRAND}) \quad (3)$$

PF	... Purchase frequency	AGE ₀	... Current age of customer
INCOME	... Net household income	BRAND	... "The reputation of a brand is a crucial criteria for the purchase of a new car."
PRIVATE	... Private usage		
KILO	... km driven per year		

Applied to the validation subsample, the non-linear regression model performed best. It explained 12.3% of PF's variance, the linear regression model explaining slightly less (11.4%). The chaid analysis hardly yielded to an R square of 5%. Subsequently, the *non-linear model* was selected to calculate the purchase frequency. Using equation 3, the purchase frequency of each customer *at any age* could be predicted in the main sample. However, the first purchase frequency depends on *two cases*:

1. The car is now "younger" than a person's purchase frequency at his age usually is → CAR₀ < PF (AGE₀). If there is a 40-year-old, whose predicted purchase fre-

¹ One disadvantage of the chaid analysis is, that it is limited to independent variables with a maximum of 31 categories. Therefore, the numerical variable "age" had to be categorized. Pre-analyses with different intervals (constant, non-constant) and different numbers of categories caused no better split of the purchase frequency than the one displayed in figure 1.

quency according to equation (3) is, say, 5.4 years and his car is only three years old ($3 \text{ years} < 5.4 \text{ years}$), he will purchase his next car in 2.4 years, in $t = 1$.

2. The car is "older" than or as old as the purchase frequency is $\rightarrow \text{CAR}_0 \geq \text{PF}(\text{AGE}_0)$. For the car of our 40-year-old, which is, say, eight years old, this means: $8 \text{ years} > 6.3 \text{ years}$. The customer's car is "overdue", he purchases now, in $t = 0$.

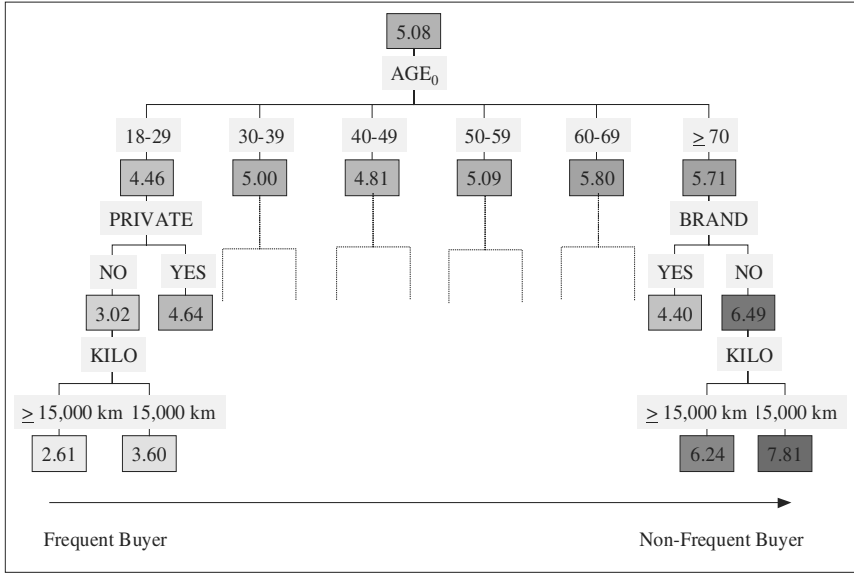


Fig. 1. PF, which is 5.08 years on average, is first split by AGE_0 yielding to 6 subgroups. Young people (18–29 years) purchase cars every 4.46 years, old people (≥ 70) less often. However, the dependency is not linear. Then, the algorithm searches for the variables causing the highest split of PF in the 6 subgroups. In level two, two different attributes split the purchase frequency: In $\text{AGE}_0 = 18\text{--}29$ years, PRIVATE is used while in category $\text{AGE}_0 \geq 70$ years, BRAND causes the best split. The corresponding leaves of the tree show, that people who use their cars for private reasons purchase less frequently than others. Similarly, consumers who are concerned with brand are more frequent buyers than others. On level three, KILO is used in both categories shown: The more people drive the faster they replace their old car

Using equation (3) and taking the two cases above into account, the purchase frequencies at any future purchase could now be computed by the following algorithm:

If $\text{CAR}_0 < \text{PF}(\text{AGE}_0)$ {case 1, first purchase in $t = 1$ }
 then $(\text{Age}_0 - \text{Car}_0) + \text{PF}(\text{AGE}_0) = \text{AGE}_1$

If $\text{CAR}_0 \geq \text{PF}(\text{AGE}_0)$ {case 2, first purchase in $t = 0$ }
 then $[\text{AGE}_0 + \text{PF}(\text{AGE}_0)] = \text{AGE}_1$

Compute

$$\text{PF}(\text{AGE}_1) = 5.834 + (-0.075 \cdot \text{AGE}_1) + (0.0017 \cdot \text{AGE}_1^2) + (-0.00000096 \cdot \text{AGE}_1^3)$$

$$+ (e^{-0.109 \cdot \text{INCOME}}) + (-1.556 \cdot \text{KILO}) + (0.429 \cdot \text{KILO}^2) + (-0.040 \cdot \text{KILO}^3)$$

$$+ (e^{0.647 \cdot \text{PRIVATE}}) - (0.961 \cdot \text{BRAND})$$

Compute $\text{AGE}_2 = \text{AGE}_1 + \text{PF}(\text{AGE}_1)$

```
If      sex = male
      then continue until AGEm ≥ 72,99
          {average life expectancy of males}
If      sex = male
      then continue until AGEm ≥ 79,59
          {average life expectancy of females}
```

(Note: AGE_t = Age of customer in t. t is not equivalent to y in equation (1): t increases by one, y represents the years when a car is purchased. So, for our 40-year-old whose car is overdue, t = 1 will be in 5.4 years from now (y = 5). t and y coincide at present, when both are zero.)

3.2 Data Mining Task 2: Prediction of Price Acceptance

To calculate the CLV, the price acceptance *at any year of purchase* had to be estimated. Price acceptance increases with age decreasing again when people retire. We related the individual price acceptance in t = 0 to the price acceptance a person of the same age usually got (see table 2). If our 40-year-old customer intends to pay 35 TDM for a car, he spends 27% more than people of his age (see equation 4). The price acceptance at any of the customer's future purchases (variable PA_y in equation 1) has to be multiplied by this price ratio, because one can assume that if a person spends more on a car than others today, he will do so in the future as well.

Table 2. Price acceptance (PA) in terms of age (excerpt)

Age	PA (Median)	Age	PA (Median)	Age	PA (Median)	Age	PA (Median)
22	17,500 DM	35	27,500 DM	50	27,500 DM	65	22,500 DM
25	22,500 DM	40	27,500 DM	55	27,500 DM	70	22,500 DM
30	22,500 DM	45	27,500 DM	60	27,500 DM	75	22,500 DM

Note: PA was acquired in categories. To avoid spans, we substituted the class by its mean.

Price Ratio = PR = $\frac{35,000\text{DM}}{27,500\text{DM}}$ = 1.2727 $\hat{=}$ 127.27% . (4)

3.3 Data Mining Task 3: Prediction of the Rate of Discount

In order to discount future streams of payment, we must consider both the *market interest* and the *inflation* (see equation 5). The *market interest* depends on the alternative investment. We assumed a certain investment keeping complexity minimal. The time span considered, i.e. the time until the customer dies, is rather long. The investment with the longest repayment period is a 30-years federal loan. Its interest receivables, the so-called spot interest rates, were taken from a leading German newspaper [5]. However, the time to maturity is 30 years, while our stream of payments goes much more far into the future. Taking the most extreme example, a 18 year old female who dies at almost 80, will purchase cars over the next 62 years. To calculate the market interest for the years 30– 62, we conducted a regression analysis with the time as the independent and the spot interest rates as the dependent variable. Several regression models (linear, logarithmic, cubic, exponential etc.) were tested if they could

reflect the slope of the observed spot interest rates, i.e. the market interest. The logarithmic function (see equation 6) yielded to the highest explained variance (99,8%).

$$\text{Discount Rate } (r) = \text{Market Interest } (mr) - \text{Price Increase } (pi) \quad . \quad (5)$$

$$\text{Spot Interest Rate} = 3.1052 + [0.9485 \cdot \ln(t)] \quad . \quad (6)$$

The market interest (mi) for the years 30–62 were estimated using equation 6. In order to estimate the *price increase* for cars (pi) for the next 62 years, we extrapolated historical data from the Federal Statistical Office. According to this, the price increase fluctuated quite evidently within the last 30 years (8.2% to -0,5%). Several regressions models explained only up to 51% of the variance. Thus, we used *exponential smoothing*. One problem is to determine the smoothing factor alpha. The higher it is, the heavier the weight of recent data. Moreover, a model with a high alpha is sensitive to structural changes [11]. These arguments suggest a relatively high alpha, say 0.5, which yields to a future price increase of about 0.8%. This seemed to be too less, the most recent years (with almost zero inflation) being quite untypical. So we chose alpha = 0.1 yielding to a future price increase of 2.5%. Finally, the *discount rate* was computed by subtracting the 2.5% price increase from the market interest.

3.4 Prediction of CLV with the Car Miner

To summarize the discussion from the last chapters, equation (1) has to be refined: As equation 7 shows, the CLV is the present value of the price acceptance at any future year of purchase. Y does not increase by one, but by the purchase frequency, which is a function of AGE_t, INCOME, PRIVATE, KILO, and BRAND. AGE_t is the only variable in this function which, in turn, depends on the purchase frequency of the year before (see the algorithm in chapter 1). Applied to the given data base, we predicted the CLV for all customers. Table 3 shows the average CLV of car drivers in the upper market segment. According to this, it is desirable to acquire BMW drivers. A driver of a BMW 7, for example, will spend about 240,000 DM on cars in his remaining life.

$$\text{CLV} = \text{PA}_0 + \sum_{y=0}^n \frac{\text{PA}(\text{AGE}_y) \cdot \text{PR}}{(1 + mr - pi)^y} \quad . \quad (7)$$

$$\text{while } y = \sum_{t=0}^m \text{PF}_t = \sum_{t=0}^m \text{PF}(\text{AGE}_t, \text{INCOME}, \text{PRIVATE}, \text{KILO}, \text{BRAND})$$

PA ₀	... Price acceptance in t = 0	PF	... Purchase frequency
PR	... Price ratio	y	... Year of purchase
n	... Year of last purchase	t	... Time period
m	... Last period	r	... Rate of discount
mr	... Market rate of interest	pi	... Price increase for cars

Note: PA₀ only has to be included if the car is "overdue". Furthermore, PRIVATE is not constant. We set it to YES when people reached their retirement age.

Table 3. CLV as gross present value in terms of preferred brand

Brand and Type	CLV (in TDM)	Brand and Type	CLV (in TDM)
BMW 7	240	Audi 100 / A 6	127
BMW 5	180	Mercedes S	111
BMW 3	148	Mercedes 190 / C	103
Mercedes 200 / E	145	Audi 80 / A4	97

4 Restrictions and Guidelines for Future Research

- The regression model explains only 12% of the purchase frequency. Its power could be improved by including external variables (e.g. macro-economic trends).
- The FED influences the market interest as well as the price increase. Thus, the Value Miner should be calibrated with respect to different FED policy scenarios.
- The life expectancy has been increasing for the last years and is expected to rise in the future. So people will purchase more cars than assumed. On the opposite, people don't drive until they die. So the two effects cancel each other out to some degree. However, a more precise model should take both effects into consideration.
- The composite model introduced in chapter 3.4 is subject to further evaluation. The impact of varying constituents and / or parameters (e.g. linear regression model for the estimation of PF, uncertainty when estimating the interest receivables from the alternative investment) should be shown in alternative models.
- When it comes to the retention of current customers, the CLV should be calculated as a net value. This requires a sophisticated accounting. Moreover, soft benefits such as reference potential should be included in that case.

References

1. Brosius, F. 1997. SPSS Chaid (in German). Internat. Thomson Publ., Bonn (1997)
2. Bitran, G. R., Mondschein, S.: Mailing Decisions in the Catalogue Sales Industry. *Management Science* 9 (1996) 1364–1281
3. Cornelsen, J.: Customer Value. Working Paper, University of Erlangen-Nürnberg (1996)
4. Dwyer, F. R.: Customer Lifetime Profitability to Support Marketing Decision Making. *Journal of Direct Marketing* 4 (1989) 8–15
5. FAZ Frankfurter Allgemeine Zeitung (1999) 41
6. Johnson, M. D., Herrmann, A., Huber, F., Gustafsson (eds.): Customer Retention in the Automotive Industry. Gabler Verlag, Wiesbaden (1997)
7. Homburg, C.: Cross-Validation and Information Criteria in Causal Modeling. *Journal of Marketing Research*, 2 (1991) 137–145
8. Jackson, D. R.: Strategic Application of Customer Lifetime Value in the Direct Marketing. *Journal of Targeting, Measurement and Analysis for Marketing* 1 (1994) 9–17
9. Keane, T. J., Wang, P.: Applications for the Lifetime Value Model in Modern Newspaper Publishing. *Journal of Direct Marketing* 2 (1995) 59–66
10. Mulhern, F. J.: Customer Profitability. *Journal of Interactive Marketing* 1 (1999) 25–40
11. Nieschlag, R., Dichtl, E., Hörschgen, H.: Marketing, Duncker & Humblot, Berlin (1998)
12. Reichheld, F. F., Aspinall, K.: Building High-Loyalty Business Systems. *Journal of Retail Banking* 4 (1993/1994) 21–29

Investigation and Reduction of Discretization Variance in Decision Tree Induction

Pierre Geurts and Louis Wehenkel

University of Liège, Department of Electrical and Computer Engineering
Institut Montefiore, Sart-Tilman B28, B4000 Liège, Belgium

Abstract. This paper focuses on the variance introduced by the discretization techniques used to handle continuous attributes in decision tree induction. Different discretization procedures are first studied empirically, then means to reduce the discretization variance are proposed. The experiment shows that discretization variance is large and that it is possible to reduce it significantly without notable computational costs. The resulting variance reduction mainly improves interpretability and stability of decision trees, and marginally their accuracy.

1 Variance in Decision Tree Induction

Decision trees ([1], [2]) can be viewed as models of conditional class probability distributions. Top down tree induction recursively splits the input space into non overlapping subsets, estimating class probabilities by frequency counts based on learning samples belonging to each subset. Tree variance is the variability of its structure and parameters resulting from the randomness of the learning set; it translates into prediction variance yielding classification errors.

In regression models, prediction variance can be easily separated from bias, using the well-known bias/variance decomposition of the expected square error. Unfortunately, there is no such decomposition for the expected error rates of classification rules (e.g. see [3,4]). Hence, we will look at decision trees as multidimensional regression models for the conditional class probability distributions and evaluate their variance by the regression variance resulting from the estimation of these probabilities. Denoting by $\hat{P}_N(C_i|x)$ the conditional class probability estimates given by a tree built from a random learning set of size N at a point x of the input space, we can write this variance (for one class C_i):

$$Var(\hat{P}_N(C_i|\cdot)) = E_X\{E_{LS}\{(\hat{P}_N(C_i|x) - E_{LS}\{\hat{P}_N(C_i|x)\})^2\}\}, \quad (1)$$

where the innermost expectations are taken over the set of all learning sets of size N and the outermost expectation is taken over the whole input space. Friedman [4] has studied the impact of this variance on classification error rates, concluding to the greater importance of this term as compared to bias.

Sources of Tree Variance. A first (important) variance source is related to the need for discretizing continuous attributes by choosing thresholds. In

local discretization, such thresholds are determined on the subset of learning samples which reach a particular test node. Since many test nodes correspond to small sample sizes (say, less than 200), we may expect high threshold variance unless particular care is taken. We will show that classical discretization methods actually lead to very high threshold variance, even for large sample sizes.

Another variance source is the variability of tree structure, i.e. the chosen attribute at a particular node, which also depends strongly on the learning set. For example, for the OMIB database (see appendix), 50 out of 50 trees built from randomly selected learning sets of size 500 agreed on the choice of the root attribute, but only 27 at the left successor and only 22 at the right successor.

A last variance source relates to the estimation of class probabilities, but this effect turns out to be negligible (for pruned trees). Indeed, fixing tree structure and propagating different random learning sets to re-estimate class probabilities and determine the variance, yields with the OMIB database a variance of 0.004, which has to be compared to a total variance of 0.05 (see Table 2).

To sum up, tree variance is important and mainly related to the local node splitting technique which determines the tree structure. The consequences are : (i) questionable interpretability (we can not really trust the choice of attributes and thresholds); (ii) poor estimates of conditional class probabilities; (iii) sub-optimality in terms of classification accuracy, but we have still to prove this.

Reduction of Tree Variance. In the literature, two approaches have been proposed : pruning and averaging. Pruning is computationally inexpensive, reduces complexity significantly and variance to some extent, but also increases bias. Thus, it improves only slightly interpretability and accuracy. Averaging reduces variance and indirectly bias, and hence leads in some problems to spectacular improvements in accuracy. Unfortunately, it destroys the main attractive features of decision trees, i.e. computational efficiency and interpretability.

It is therefore relevant to investigate whether it is possible to reduce decision tree variance without jeopardizing efficiency and interpretability. In what follows, we will focus on the local discretization technique used to determine thresholds for continuous attributes and investigate its variance and ways to reduce it. We show that this variance may be very large, even for reasonable sample sizes, and may be reduced significantly without notable computational costs.

In the next section we will study empirically the threshold variance of three different discretization techniques, then propose a modification of the classical method in order to reduce threshold variance significantly. In the following section we will assess the resulting impact in terms of global tree performance, comparing our results with those obtained with tree bagging [5].

2 Evaluating and Reducing Threshold Variance

Classical Local Discretization Algorithm. In the case of numerical attributes, the first stage of node splitting consists in selecting a discretization threshold for each attribute. Denoting by a an attribute and by $a(o)$ its value for a given sample o , this amounts to selecting a threshold value a_{th} in order to split

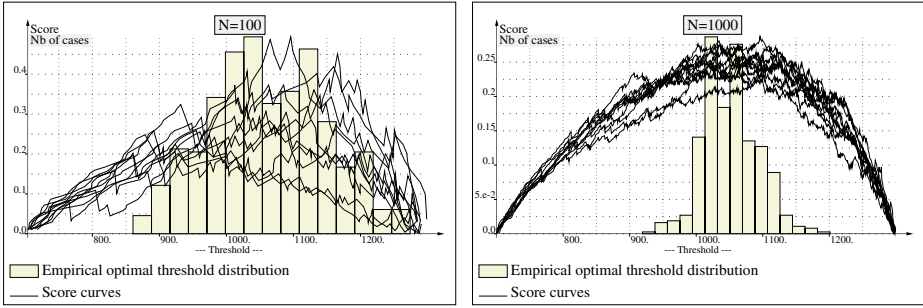


Fig. 1. 10 score curves and empirical optimal threshold distribution for learning sets of size 100 (left) and 1000 (right). OMIB database, attribute Pu

the node according to the test $T(o) \equiv [a(o) < a_{th}]$. To determine a_{th} , normally a search procedure is used so as to maximize a score measure evaluated using the subset $ls = \{o_1, o_2, \dots, o_n\}$ of learning samples which reach the node to split. Supposing that the ls is already sorted by increasing values of a , most discretization techniques exhaustively enumerate all thresholds $\frac{a(o_i) + a(o_{i+1})}{2} (i = 1 \dots n - 1)$. Denoting the observed classes by $C(o_i), (i = 1, \dots, n)$, the score measures how well the test $T(o)$ correlates with the class $C(o)$ on the sample ls . In the literature, many different score measures have been proposed. In our experiments we use the following normalization of Shannon information (see [6,7] for a discussion)

$$C_C^T = \frac{2I_C^T}{H_C + H_T}, \quad (2)$$

where H_C denotes class entropy, H_T test entropy (also called split information by Quinlan), and I_C^T their mutual information.

Figure 1 represents the relationship between C_C^T and the discretization threshold, for the OMIB database (see appendix). Each curve shows the variation of score in terms of discretization threshold for a given sample. The histograms beneath the curves correspond to the sampling distribution of the global maxima of these curves (i.e. the threshold selected by the classical method). One observes that even for large sample sizes (right hand curves), the variance of the “optimal” threshold determined by the classical method remains rather high.

Figure 2 shows results for sample sizes $N \in [50; 2500]$ obtained on the GAUSSIAN database according to the following procedure : (i) for each value of N , 100 samples $ls_1, \dots, ls_{100}$ of size N are drawn; (ii) for each ls_i the threshold a_{th}^i maximizing $\hat{C}_C^T(ls_i)$ is computed, as well as left and right hand estimates of conditional class probabilities. The graphs of Figure 2 plot the averages ($\pm$ standard deviation) of these 100 numbers as a function of N ; it highlights clearly how slowly threshold variance decreases with sample size.

Alternative Discretization Criteria. To assess whether the information theoretic measure is responsible for the threshold variance, we have compared it with two alternative criteria : (i) **Kolmogorov-Smirnov measure** (see [8]); (ii) **Median**, a naïve method discretizing at the (local) sample median.

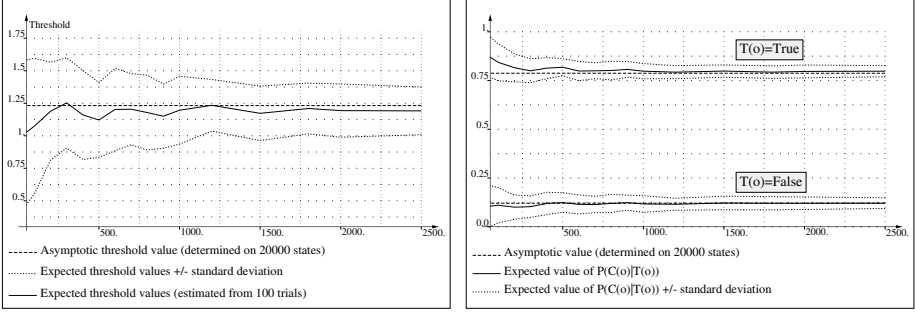


Fig. 2. Expected threshold values and standard deviation (left); Class probability estimates and standard deviation (right). Attribute a_1 of GAUSSIAN database

Table 1. OMIB database, asymptotic value of $a_{th}=1057$, $\sigma_{attribute} = 170$

method	$N = 50$			$N = 500$			$N = 2000$		
	$\sigma_{a_{th}}$	$b(a_{th})$	$Var(\hat{P})$	$\sigma_{a_{th}}$	$b(a_{th})$	$Var(\hat{P})$	$\sigma_{a_{th}}$	$b(a_{th})$	$Var(\hat{P})$
classic	91.0	-15.6	0.01335	55.4	-1.5	0.00383	36.8	-8.6	0.00138
Kolmogorov	59.3	-13.8	0.00900	26.6	-13.5	0.00126	18.7	-18.6	0.00042
median	38.2	-55.9	0.00772	13.1	-59.2	0.00095	6.1	-58.8	0.00016
averaging	34.6	-49.3	0.00945	20.3	-20.0	0.00115	14.3	-13.0	0.00035
bootstrap	56.0	22.4	0.00834	37.0	2.8	0.00194	25.9	-8.5	0.00071
smoothing	96.6	-1.7	0.01485	51.6	-1.0	0.00317	33.2	-8.8	0.00108

The upper part of Table 1 shows results obtained for one of the test databases (using the same experimental setup as above). It provides, for different sample sizes, threshold standard deviations ($\sigma_{a_{th}}$) and bias ($b(a_{th})$, the average difference with the asymptotic threshold determined by the classical method and using the whole database), and standard deviations of class probability estimates (average of the two successor subsets, denoted $Var(\hat{P})$). Note that the results for the other two databases described in the appendix are very similar to those shown in Table 1. They confirm the high variance of thresholds and probability estimates determined by the classical technique, independently of the considered database. On the other hand the “median” and to a lesser extent the “Kolmogorov-Smirnov measure” reduce variance very strongly, but lead to a significant bias with respect to the classical information theoretic measure. Note that median is not a very sensible choice for decision tree discretization, since it neglects the distribution of classes along the attribute values.

Improvements of the Classical Method. The very chaotic nature of the curves of Figure 1 obviously is responsible of the high threshold variance. We have thus investigated different techniques to “smoothen” these curves before determining the optimal threshold, of which we report the three following :

Smoothing : a moving-average filter of a fixed window size is applied to the score curve before selecting its maximum (window size was fixed to $ws = 21$).

Averaging : (i) the score curve and the optimal threshold are first computed, yielding test T^* as well as the score estimate $\hat{C}_C^{T^*}$ and its standard deviation estimate $\hat{\sigma}_{C_C^{T^*}}$ (see [9]); (ii) a second pass through the score curve determines the smallest and largest threshold values $\underline{a}_{th}$ and $\bar{a}_{th}$ yielding a score larger than $\hat{C}_C^{T^*} - \lambda \hat{\sigma}_{C_C^{T^*}}$, where λ is a tunable parameter set to 2.5 in our experiments; (iii) finally the discretization threshold is computed as $\underline{a}_{th}^* = (\underline{a}_{th} + \bar{a}_{th})/2$.

Bootstrap : the procedure is as follows : (i) draw by bootstrap (i.e. with replacement) 10 learning sets from the original local learning subset; (ii) use the classical procedure on each subsample to determine 10 threshold values; (iii) determine discretization threshold as the average of these latter.

These variants of the classical method were evaluated using the same experimental setup as before. Results are shown in the lower part of Table 1; they show that “averaging” and “bootstrap” allow to reduce the threshold variance significantly, while only the former increases (slightly) bias. The same holds in terms of reductions of probability estimate variance. Hence averaging is the most interesting, since it does not increase significantly computing times.

3 Global Effect on Decision Trees

To evaluate the various discretization techniques in terms of global performance of decision trees, we carried out further experiments. The databases are first split into three disjoint parts : a set used to pick random samples for tree growing (LS), a set used for cross-validation during tree pruning (PS), a set used for testing the pruned trees (TS) (the divisions for each database are shown in Table 3, in the appendix). Then, for a given sample size N , 50 random subsets are drawn without replacement from the pool LS , yielding $LS_1, LS_2, \dots, LS_{50}$, and for each method the following procedure is carried out

- A tree is grown from each LS_i and for each discretization method.
- These trees are pruned (see [10] for a description of the method), yielding the trees $\mathcal{T}_i, (i = 1, \dots, 50)$.
- Average test set error rate $\bar{P}_e$ and complexity $\bar{C}$ of the 50 trees are recorded.
- To evaluate variance, the quantity (1) is estimated using the test sample, providing $\hat{Var}(\hat{P}_{\mathcal{T}_i}(C|\cdot))$

Table 2 shows results obtained on the three databases for a learning sample size of $N = 1000$; note that similar results were obtained for smaller and larger learning sets but are not reproduced here due to space limitations (for more details please refer to [11]). The last line of the table provides, as a ground for comparison, the results obtained by tree bagging, implemented using 10 bootstrap samples and aggregation of class-probability estimates of pruned trees, reporting the sum of the complexities of the 10 trees. One observes that all the methods succeed in decreasing the variance of the probability estimates on the three databases, the most effective being the median, followed by averaging and Kolmogorov-Smirnov. But, comparing the reduction in variance with the one obtained in the previous section, we note that the decrease is less impressive

Table 2. Results on three databases (global tree performances for $N = 1000$)

method	Gaussian ($P_e^B = 11.8\%$)			Omib ($P_e^B = 0\%$)			Waveform ($P_e^B = 14\%$)		
	P_e	C	$\hat{v}ar$	P_e	C	$\hat{v}ar$	P_e	C	$\hat{v}ar$
classic	12.56	10.32	0.0147	11.20	67.6	0.0572	27.30	45.96	0.0434
Kolmogorov	12.85	9.92	0.0109	10.41	73.6	0.0493	27.57	54.12	0.0432
median	12.17	14.28	0.0083	10.39	103.92	0.0383	27.30	66.04	0.0382
averaging	12.21	17.32	0.0105	10.69	98.68	0.0493	27.56	55.64	0.0386
bootstrap	12.49	12.28	0.0133	11.59	74.6	0.0500	27.39	49.48	0.0402
smoothing	12.56	9.88	0.0137	10.89	77.4	0.0532	27.23	47.68	0.0396
tree bagging	12.07	92.3	0.0047	8.29	468.6	0.0133	20.83	367.3	0.0100

here. The main reason for this is that tree pruning, as it adapts the tree complexity to the method, has the side effect of increased complexity of the trees obtained with the variance reduction techniques. This balances to some extent the local variance reduction effect. From the tables it is quite clear that median and averaging reduce variance locally most effectively, but also lead to the highest increase in tree complexity. The error rates are mostly unaffected by the procedure; they decrease slightly on the GAUSSIAN and OMIB databases while they remain unchanged on the WAVEFORM database.

Unsurprisingly, tree bagging gives very impressive results in terms of variance reduction and error rates improvement on all the databases, and especially on the WAVEFORM. Of course, we have to keep in mind that this improvement comes with a loss of interpretability and a much higher computational cost.

4 Discussion and Related Work

In this paper, we have investigated the reduction of variance of top down induction of decision trees due to the discretization of continuous attributes, considering its impact on both *local* and *global* tree characteristics (interpretability, complexity, variance, error rates). In this, our work is complementary to most existing work on discretization which has been devoted exclusively to the improvement of *global* characteristics of trees (complexity and predictive accuracy), neglecting the question of threshold variance and interpretability.

On the other hand, several authors have proposed tree averaging as a means to decrease the important variance of the decision tree induction methods, focusing again on global accuracy improvements. This has led to variations on the mechanism used to generate alternative trees and on the schemes used to aggregate their predictions. The first well known work in this context concerns the Bayesian option trees proposed by Buntine [12], where several trees are maintained in a compact data structure, and a Bayesian scheme is used to determine a posteriori probabilities in order to weight the predictions of these trees. More recently, so-called tree bagging and boosting methods were proposed respectively by Breiman [5] and Freund and Schapire [13]. In addition to the spectacular accuracy improvement provided by these latter techniques, they are attractive

because of their generic and non-parametric nature. From our investigations it is clear that these approaches are much more effective in improving global accuracy than local variance reduction techniques such as those proposed in this paper. However, the price to pay is a definite shift towards black-box models and a significant increase in computational costs. Our intuitive feeling (see also the discussion in Friedman [4]) is that tree averaging leads to local models, closer in behavior to nearest-neighbor techniques than classical trees. In terms of predictive accuracy, we may thus expect it to outperform classical trees in problems where the kNN method outperforms them (as a confirmation of this, we notice that kNN actually outperforms tree bagging significantly on the WAVEFORM dataset).

Another recent class of proposals more related to our local approach and similar in spirit to the early work of Carter and Catlett [14], consists in using continuous transition regions instead of crisp thresholds. This leads to overlapping subsets at the successor nodes and weighted propagation mechanisms. For example, in a fuzzy decision tree, fuzzy logic is used in order to build hierarchies of fuzzy subsets. Wehenkel ([9]) showed that in the context of numerical attributes this type of fuzzy partitioning allows indeed to reduce variance significantly. In [4], Friedman proposes a technique to split the learning subset into overlapping subsets and uses again voting schemes to aggregate competing predictions. Along the same ideas, we believe that a Bayesian approach to discretization ([9]) or probabilistic trees (such as those proposed in [15]) would allow to reduce variance. The main advantage of this type of approach with respect to global model averaging is to preserve (possibly to improve) the interpretability of the resulting models. The main disadvantage is a possibly significant increase in computational complexity at the tree growing stage.

With respect to all the intensive research, we believe that the contribution of this paper is to propose low computational cost techniques which improve interpretability by stabilizing the discretization thresholds and by reducing the variance of the resulting predictions. In the problems where decision trees are competitive, these techniques also improve predictive accuracy. We also believe that our study sheds some light on features of decision tree induction and may serve as a starting point to improve our understanding of its weaknesses and strengths and eventually yield further improvements.

Although we have focused here on local (node by node) discretization philosophies, it is clear from our results that global discretization must show similar variance problems and that some of the ideas and methodology discussed in this paper could be successfully applied to global discretization as well. More broadly, all machine learning methods which need to discretize continuous attributes in some way, could take advantage of our improvements.

In spite of the positive conclusions, our results show also the limitations of what can be done by further improving decision tree induction without relaxing its intrinsic representation bias. A further significant step would need a relaxation of this representation bias. However, if we want to continue to use the resulting techniques for data exploration and data mining of large datasets, this must be

achieved in a cautious way without jeopardizing interpretability and scalability. We believe that fuzzy decision trees and Bayesian discretization techniques are promising directions for future work in this respect.

References

1. L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth International (California), 1984. 162, 169
2. J.R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann (San Mateo), 1986. 162
3. R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proc. of the Thirteenth International Conference on Machine Learning*, 1996. 162
4. J. H. Friedman. Local learning based on recursive covering. Technical report, Department of Statistics, Stanford University, August 1996. 162, 168
5. L. Breiman. Bagging predictors. Technical report, University of California, Department of Statistics, September 1994. 163, 167
6. R.L. De Mantaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92, 1991. 164
7. L. Wehenkel. On uncertainty measures used for decision tree induction. In *Proc. of Info. Proc. and Manag. Of Uncertainty*, pages 413–418, 1996. 164
8. J. H. Friedman. A recursive partitioning decision rule for nonparametric classifier. *IEEE Transactions on Computers*, C-26:404–408, 1977. 164
9. L. Wehenkel. Discretization of continuous attributes for supervised learning : Variance evaluation and variance reduction. In *Proc. of The Int. Fuzzy Systems Assoc. World Congress (IFSA'97)*, pages 381–388, 1997. 166, 168
10. L. Wehenkel. *Automatic learning techniques in power systems*. Kluwer Academic, Boston, 1998. 166, 169
11. P. Geurts. Discretization variance in decision tree induction. Technical report, University of Liège, Dept. of Electrical and Computer Engineering, Jan. 2000. (<http://www.montefiore.ulg.ac.be/~geurts/>) 166
12. W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992. 167
13. Y. Freund and R.E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. In *Proc. of the 2nd European Conference on Computational Learning Theory*, pages 23–27. Springer Verlag, 1995. 167
14. C. Carter and J. Catlett. Assessing credit card applications using machine learning. *IEEE Expert*, Fall:71–79, 1987. 168
15. M. I. Jordan. A statistical approach to decision tree modeling. In *Proc. of the 7th Annual ACM Conference on Computational Learning Theory*. ACM Press, 1994. 168

A Databases

Table 3 describes the datasets (last column is the Bayes error rate) used in the empirical studies. They provide large enough samples and present different features : GAUSSIAN corresponds to two bidimensional Gaussian distributions; OMIB is related to electric power system stability assessment [10]; WAVEFORM denotes Breiman’s database [1].

Table 3. Datasets (request from `geurts@montefiore.ulg.ac.be`)

Dataset	#Variables	#Classes	#Samples	#LS	#PS	#TS	P_e^{Bayes}
GAUSSIAN	2	2	20000	16000	2000	2000	11.8
OMIB	6	2	20000	16000	2000	2000	0.0
WAVEFORM	21	3	5000	3000	1000	1000	14.0

Asymmetric Co-evolution for Imperfect-Information Zero-Sum Games

Ole Martin Halck and Fredrik A. Dahl

Norwegian Defence Research Establishment (FFI)
P.O. Box 25, NO-2027 Kjeller, Norway
{Ole-Martin.Halck,Fredrik-A.Dahl}@ffi.no

Abstract. We present an asymmetric co-evolutionary learning algorithm for imperfect-information zero-sum games. This algorithm is designed so that the fitness of the individual agents is calculated in a way that is compatible with the goal of game-theoretic optimality. This compatibility has been somewhat lacking in previous co-evolutionary approaches, as these have often depended on unwarranted assumptions about the absolute and relative strength of players. Our algorithm design is tested on a game for which the optimal strategy is known, and is seen to work well.

1 Introduction

Within the field of machine learning, learning to play games presents special challenges. Whereas other learning tasks usually involve a fixed problem environment, game environments are more variable, as a game-playing agent must expect to face different opponents. In imperfect-information games, a class of games that has received relatively little attention in machine learning, the challenges are even greater, due to the need of acting unpredictably. In addition to the challenges encountered during the learning itself, there are also difficulties connected to evaluating the success of the training procedure, as this evaluation will need to take into account the agent's performance against varying opposition.

One main approach that has been applied to the problem of learning to play games is *co-evolution*. In co-evolutionary learning, agents are evaluated and evolved in accordance to their performance in actual game-play against other evolving agents. The degree of success achieved by the co-evolution of agents has been variable; in this paper, we attempt to shed some light on the reasons for this.

The main contributions of this paper comprise a theoretical and a practical component. We argue that much previous research of machine learning in games reveals a need of theoretical awareness regarding the evaluation of game-playing agents in the two phases of the learning itself and the assessment of the success of learning. We attempt to address this need by presenting a theoretical evaluation criterion that is consistent with game theory. On a practical level, we use this theoretical viewpoint in reviewing different co-evolutionary learning methods, and present a new, asymmetric co-evolutionary design that solves some of the problems attached to more traditional approaches.

The remainder of the paper is laid out as follows: Section 2 treats the relationship between machine learning and game theory; here we present our evaluation criterion and discuss the goals of learning in games. In Section 3 we describe different designs for co-evolutionary learning in games – including our new algorithm – and examine their properties in a game-theoretical light. Section 4 describes experiments that illustrate the treatment given in Section 3. A discussion of our goals, method and results is given in Section 5, while Section 6 concludes the paper.

2 Machine Learning and Imperfect-Information Games

In game theory, a distinction is made between games with perfect and imperfect information. In perfect-information games, the players always have the same information about the game state; in imperfect-information games, the players have different state information. Poker is an example of an imperfect-information game – the players know their own cards, but not those of their opponents. A seemingly different source of information imperfection occurs in games with simultaneous actions, such as scissors–paper–rock. However, these games may be transformed into equivalent alternating-turn games with “normal” imperfect information (see e.g. [2]), and vice versa.

In the literature on machine learning in games, most of the focus has been on games with perfect information. Imperfect-information games seem to have been somewhat neglected in comparison, as noted and discussed in [4] and [2].

In this paper, we restrict our attention to two-player zero-sum games with imperfect information. The consequences of the zero-sum restriction, along with other important game-theoretical background, is explained in the following. We then turn to the significance this theory has for evaluating game-playing agents and for machine learning of games.

2.1 Theory of Imperfect-Information Zero-Sum Games

In the tradition of von Neumann and Morgenstern [10], a *game* is defined as a decision problem with two or more decision makers – *players* – where the outcome for each player may depend on the decisions made by all players. Each player evaluates possible outcomes in terms of his own utility function, and works to maximise his own expected utility only. Here, we restrict ourselves to games with two players; these players will be called Blue and Red.

A *pure strategy* for a player is a deterministic plan dictating his actions in every possible observed state of the game. A *mixed* or *randomized strategy* is a weighted average of pure strategies, where the weights are interpreted as the probability of choosing the associated pure strategy. A mixed strategy may also be specified in a behavioural way, by giving the probability distributions over available actions in each possible game state. Only finite games are considered in this paper, that is, we will assume that each player has a finite number of pure strategies, and that the payoffs are bounded.

We further limit our attention to *zero-sum* games, that is games where one player wins what the other loses, thus eliminating any incentive for co-operation between the players. Any finite two-player zero-sum game has a *value* v , a real number with the property that Blue has a strategy (possibly mixed) which guarantees that the expected payoff will be at least v , while Red has a strategy guaranteeing that Blue's payoff is at most v [9]. Clearly, these strategies are then *minimax strategies* or *solutions*, strategies that give the respective players their highest payoffs against their most dangerous respective opponents. Furthermore, when (and only when) both players employ minimax strategies, a *minimax equilibrium* or *solution* of the game occurs; the definition of such an equilibrium is that neither player gains by deviating from his strategy, assuming that the opponent does not deviate from his. The minimax equilibrium need not be unique, but in zero-sum games all such equilibria are associated with the same value.

In perfect-information games, there exist deterministic minimax equilibria, that is equilibria where each player can play optimally in the game-theoretic sense by employing a pure strategy. In games with imperfect information, however, mixed strategies are in general necessary. In scissors–paper–rock, for instance, the unique minimax strategy for each player is to choose randomly, with uniform probability, between the three pure strategies.

A more thorough treatment of these and other aspects of game theory can be found in e.g. [7].

2.2 Evaluating Performance

We now present a game-theoretic evaluation criterion for players of two-player zero-sum imperfect-information games. The set of all mixed strategies as defined above is denoted by M , a player is specified by the strategy it employs. Although this theoretical criterion is not practically applicable in games that have not been solved, it is crucial for a stringent treatment of game learning. For a further discussion of evaluation criteria in games, see [2].

The criterion we use is that of *equity against worst-case opponent*, denoted by *Geq*. For a given $P \in M$ it is defined as

$$Geq(P) = \inf_{Q \in M} \{E(P, Q)\}, \quad (1)$$

where $E(P, Q)$ denotes the expected outcome of P when playing against Q . According to this definition, the *Geq* measure gives the expected outcome for P when playing against its most dangerous opposing strategy. In a game with value v , it is clear that $Geq(P) \leq v$ for all $P \in M$; P is a minimax solution if and only if $Geq(P) = v$. Thus, the *Geq* criterion has an immediate game-theoretic interpretation. It should also be noted that for a given P , there exists a *pure* opposing strategy Q that reaches the infimum, that is, there is a deterministic agent which makes P look the worst.

2.3 The Goals of Learning

The general goal of machine learning algorithms is to perform well in a problem domain by using information gained from experience within that domain. The agent typically trains itself on a limited set of domain data in order to become adept at handling situations that are not covered by the training set. If this is to succeed, it is clearly important that the feedback received during training corresponds to what we mean by good performance within the domain. In addition, the practitioner of machine learning needs to assess the degree to which the learning has been successful. Thus, performance evaluation is important both in the learning itself and in the assessment of the success of the learning procedure. Without evaluation, learning can neither be measured nor occur.

Machine learning in games presents special problems compared to other domains. The feedback received by an agent during game play depends critically on the agent it is playing against; that is, the environment is not fixed. Furthermore, for games that have not already been solved, it is difficult to define an objective evaluation criterion, and performance has to be measured in actual game play, which, again, depends on the opponents used.

Often, the goal of game-learning work is inadequately expressed. It is taken for granted that we want the resulting player to play “well”, hopefully even “optimally”, without any clear definition of what this entails – the idea of objective game-play quality is taken for granted. In some cases, agents are trained against and evaluated by the same opponents; this in essence turns the problem into a normal learning problem rather than a game-learning one. Sometimes, however, the goal is clearly stated, as in [6], where it is said: “In the game theory literature, the resolution of this dilemma is to eliminate the choice and evaluate each policy with respect to the opponent that makes it look the worst.” According to this view, “optimal play” takes on the natural meaning of “game-theoretic solution”, and the *Geq* criterion is the correct one for player evaluation. This is the view that will be used in the following.

3 Co-evolutionary Approaches

Within the framework of evolutionary computation, *co-evolution* has been used as a way of overcoming the problems presented by game-playing domains, see e.g. [1] and [12]. Here, an agent’s fitness is measured by its performance in actual game play against other evolving agents, rather than how well it performs in a fixed environment. The idea is that the evolving players will drive each other toward the optimum by an evolutionary “arms race”. In the following we discuss some basic forms of co-evolutionary learning and certain problems associated with these, and present an algorithm designed to overcome these problems. In all cases discussed, we consider two-population co-evolution, where each population contains players of one side in the game.

3.1 Basic Forms of Co-evolution

Accumulated Fitness. A seemingly natural way of evaluating the individuals in co-evolving populations is to play a tournament where each Blue individual plays each Red individual, accumulating the scores from the single games. An individual's fitness is then the total number of points scored against all members of the other population.

This approach may work in special cases, but fails in general. Several plausible reasons for this type of failure have been suggested, e.g. in [12], along with remedies for these problems. However, we see the main problem as lying in the “arms race” assumption mentioned above. This assumption is based on the idea that relative performance between players correlates well to the quality of the players as measured by the ultimate goal of the training (in our case a high *Geq* score), so that players beating each other in turn will get closer to this goal. Thus, a high degree of transitivity in the “who-beats-whom” relation is assumed.

Unfortunately, games generally display a lack of such transitivity, and this is especially true for imperfect-information games – scissors–paper–rock provides a trivial example. (This has also been recognised in e.g. [11].) Thus we see that the essential fault in this form of co-evolution is the discrepancy between the criterion used for giving feedback to the players and the criterion we evaluate them according to after training is done.

Worst-Case Fitness. With the above in mind, we naturally seek a better way of assigning fitness to the players during co-evolution, a way which corresponds better to our goal of a high *Geq* score. Since the *Geq* criterion tells us the expected performance when pitted against the most effective counter-strategy, it is tempting to let each individual's fitness be given by an estimate of its performance against the member of the other population which is most dangerous to the individual being evaluated.

Due to the mixed strategies of the agents, this calls for a more time-consuming tournament than in the case of accumulated fitness. With accumulated fitness, one game against each opponent gives an unbiased estimate of the fitness, as the expected value of the sum of the outcomes equals the sum of the expected values. Worst-case fitness, on the other hand, requires several games against each opponent, as the expected value of the minimum of the outcomes (which can be estimated by playing one game against each) is different from the minimum of the expected values, which is the fitness measure we want.

What is even worse, though, is that even if we play the number of games necessary for achieving good worst-case fitness estimates, this method cannot be expected to converge towards an optimal *Geq* score. The reason lies in the somewhat paradoxical nature of the minimax solution concept. At an equilibrium, where both sides play mixed strategies that are minimax solutions, neither side has anything to gain by deviating unilaterally. On the other hand, there is also nothing to *lose* by unilateral deviation, as long as only pure strategies present in the optimal mixture are used. Thus, even if the co-evolutionary procedure were to attain the optimum, this would not be a stable state.

3.2 An Algorithm for Asymmetric Co-evolution

We are now able to identify some conditions that should be met by a co-evolutionary game-learning algorithm if we are to expect convergence towards the game-theoretic optimum. First, the fitness evaluations should conform to the goal of the training – that is, they should be estimates of the *Geq* values of the individuals. Secondly, the minimax strategy – which is what we want – should be a stable state of the algorithm. We here propose an algorithm that is designed to meet these conditions.

The Populations. The most important feature of our algorithm is its *asymmetry*. Recall from Section 2.2 that among the most effective strategies against a given individual, there is always a pure one. Since we want the fitness of our resulting individuals to reflect the *Geq* criterion, we give one of the populations the task of being *Geq* estimators, and let it consist of *deterministic* agents rather than randomising ones. This also solves the problem of the minimax solution being unstable, as the solution is the only strategy that is not punished by any pure strategy.

Consequently, we let the Blue population be the one we train towards the optimal game-theoretic strategy. This population then consists of individuals with a representation that allows them to employ mixed strategies. In practice, this means that the output of each Blue agent in a game state should be a vector of nonnegative real numbers that sum to unity; this vector is interpreted as the agent’s probability distribution for choosing between the available actions. When playing the game, the agent picks a random action using this distribution.

The Red population consists of individuals that are only able to play pure strategies, that is, in a given game state each Red individual always chooses the same action. Note that it is not necessary to devise another design and representation for this purpose. We may use the same as for Blue, and just change the interpretation of the output vector, so that the Red agent always chooses the action associated with the highest value. (In the case of ties between two or more actions, we may use an arbitrary policy for choosing between these, as long as it is consistent – this is necessary for maintaining the determinism of the agents.)

In order to ensure that the learning task for Blue gets monotonically more difficult over time, forcing it towards the optimum, we use a *hall of fame*, consisting of effective pure strategies found during training, for the Red population. This device has also been used for similar reasons in [12].

The Algorithm. The algorithm itself runs as follows: After initialising the populations with individuals having the properties described above, we use some method – such as a random draw, a heuristic or a simple tournament – for selecting a Blue individual that we designate as our nominee for the currently “best” Blue player. Then the following procedure is repeated (cf. Figure 1):

- Train the Red population for a few generations; the fitness measure for each individual is its performance against the Blue player currently nominated as best;
- Add the Red individual coming out on top after this training to the hall of fame;

- Train the Blue population for a few generations; the fitness measure for each individual is its performance against the member of the Red hall of fame which is most dangerous to that Blue individual;
- Nominate the Blue individual coming out on top after this training as the currently best Blue player.

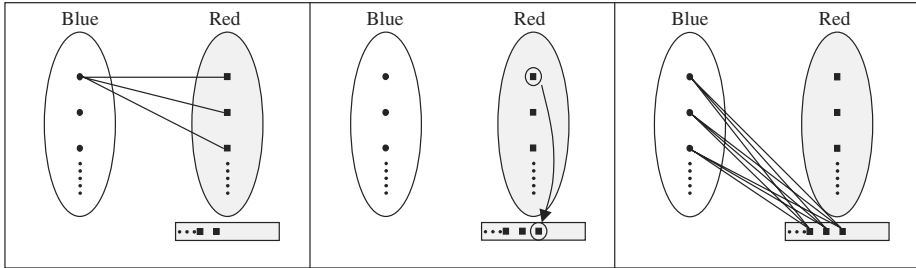


Fig. 1. Algorithm for asymmetric co-evolution

The goal of the Blue training is to find individuals that randomise between pure strategies in a way that makes it impervious to exploitation by the dangerous Red agents found; this drives the Blue agents towards the optimum. The Red training amounts to searching for a hole in the defence of the best Blue agent, thus giving the Blue population a chance to mend this flaw in the next training cycle. The metaphor of *hosts* and *parasites* [12] is particularly fitting in this setting, more so than in the symmetric cases where it is otherwise used. A host needs to guard itself against a broad variety of parasites, whereas a parasite is more than happy as long as it can break through a single host's defence. The parallel to the asymmetric layout of our algorithm should be obvious.

As for worst-case fitness (Section 3.1), it is necessary to play several games for each pair of players to obtain good performance estimates, due to the randomisation performed by the Blue agents (see also Section 5).

4 Experiments

The purpose of the experiments reported in this section is to illustrate the claims made about the different co-evolutionary designs discussed above. Therefore, we have applied the designs to a toy problem for which the solution is known, namely a modified version of the game Undercut. Furthermore, in order to factor out the effect of inaccurate performance estimates from our investigation of the designs themselves, we have used calculated expected results in our fitness assignments instead of sampled estimates.

Some standard terminology of evolutionary computation is used in the descriptions below; see e.g. [8] for definitions and explanations.

4.1 The Game of Zero-Sum Undercut

The two-player imperfect-information game of Undercut was invented by Douglas Hofstadter [3]. The rules are as follows: Each player selects a number between 1 and 5 inclusive. If the choice of one player is exactly one lower than that of the opponent (the player “undercuts” the opponent), the player receives a payoff equalling the sum of the two numbers. Otherwise, each player receives a payoff equalling his own choice. To make the game more challenging, we expand the available choices to the numbers from 1 through 30.

Undercut is clearly not zero-sum; we make a zero-sum version by changing the payoff structure somewhat. A player undercutting his opponent receives the sum of the choices from the opponent; if there is no undercut, the player with the highest choice receives the *difference* between the choices from the opponent. If, for example, Blue plays 14 and Red 22, Red wins 8 from Blue (i.e. Blue gets payoff -8 , Red gets 8); if Blue plays 26 and Red 27, Blue wins 53 from Red. As the game is symmetric, its value is clearly zero; thus, the optimal *Geq* evaluation is also zero. The worst possible *Geq* score, incidentally, belongs to the strategy of always playing 30; the most effective counter-strategy is always playing 29, and the minimum score is -59 .

The game can be solved using techniques like linear programming [14] or fictitious play [7]; the probability distribution of the solution is given in Table 1. (Choices not appearing in the table should not be played.)

Table 1. Solution of zero-sum Undercut with 30 choices

Choice	22	23	24	25	26	27	28	29	30
Probability	0.095	0.084	0.151	0.117	0.161	0.110	0.135	0.069	0.078

4.2 Experimental Setup

For the experiments reported here, the behaviour of each individual was specified by a string of 30 real numbers in $(-1,1)$. These numbers naturally represent the probability of making the corresponding choices; to map the string into a valid probability vector, all negative entries are set to zero and the rest normalised to sum to unity. (Note that this does not affect the string itself.)

The population sizes were set to 50; 500 generations were completed for each population. Tournament selection was used for selecting parents for the genetic operations. For each pair of parents a genetic operator was chosen at random to produce two offspring; the operations and probabilities used were:

- Uniform crossover (probability $\frac{1}{2}$): for each position in the string, distribute the two parent values randomly between the children;
- Average crossover (probability $\frac{1}{4}$): for each position in the string, let p and q be the two parent values, and set the offspring values to $(2p + q) / 3$ and $(p + 2q) / 3$.
- Mutation (probability $\frac{1}{4}$): the children are copies of the parents, except that each string position is changed to a random number in $(-1,1)$ with probability $1/15$.

Elitism was used; the two individuals with the highest fitness survived from one generation to the next.

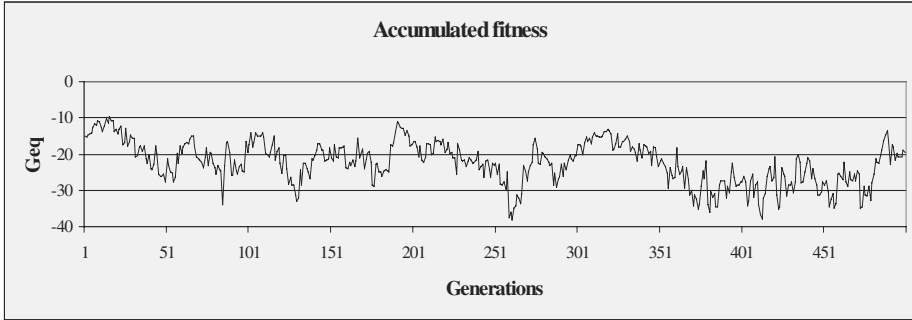


Fig. 2. *Geq* for the best individual of each Blue generation, using accumulated fitness

4.3 Results

We now present the results of applying the different co-evolutionary designs to the game of zero-sum, 30-value Undercut. We evaluate the training using the *Geq* criterion; the optimal value is then zero.

Accumulated and Worst-Case Fitness. Figure 2 shows the *Geq* of the best Blue individual of each generation when using symmetric co-evolution with accumulated fitness, averaged over five runs.

It is clear that this form of learning does not work given our goal; the reason is the lack of transitivity between strategies, as described in Section 3.1. Simply put, there is no incentive to move towards the optimum for either population, as long as the most effective strategy for exploiting the vulnerabilities of the opposing population is itself equally vulnerable.

When using worst-case fitness, the co-evolution produces better individuals than in the case of accumulated fitness, but still does not converge towards the optimum (Figure 3; notice the difference in scale compared to Figure 2).

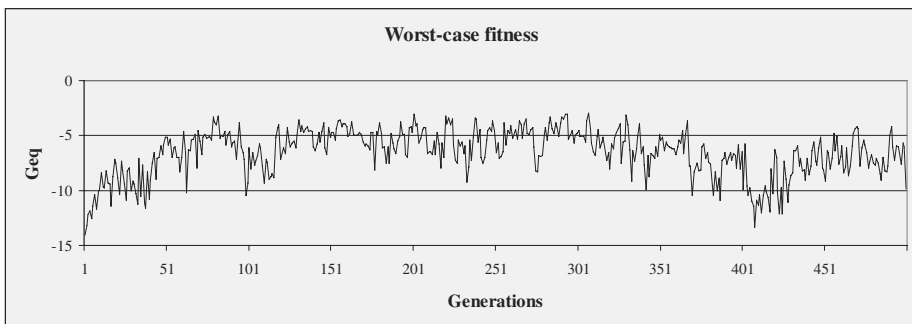


Fig. 3. *Geq* for the best individual of each Blue generation, using worst-case fitness

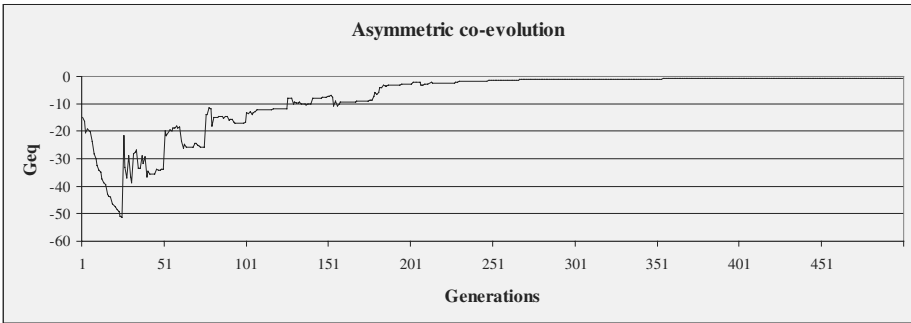


Fig. 4. *Geq* for the best individual of each Blue population, using asymmetric co-evolution

The reason for the improved performance is that worst-case fitness corresponds far better to game-theoretic evaluation than does accumulated fitness. On the other hand, the non-coerciveness of minimax play hinders a stable improvement of the agents.

Asymmetric Co-evolution. In the case of our asymmetric design of Section 3.2, we let each population train for 25 generations within each main iteration, and ran 20 of these iterations, so that the total number of generations for each side was the same as for the other designs. The results for the best Blue individual of each generation, again averaged over five runs, are shown in Figure 4.

The algorithm clearly pushes the Blue population towards better performance; the improvement gets more monotonic the more members are present in the Red hall of fame. This is due to the increased correspondence over time between Blue fitness and the *Geq* measure, as the Red hall of fame is filled with parasites that are dangerous to the various Blue strategies that may occur.

Note also that although the number of generations for each side is the same as for the symmetric designs, the total number of actual Blue–Red match-ups is much smaller than in those cases. Each Red agent always trains against *one* Blue strategy instead of a whole population, while the Blue agents are trained against a hall of fame, the size of which starts at one and increases over time. Of course, when training proceeds further, the number of opponents for each Blue agent will grow.

5 Discussion

The results of our experiments bear out what was said in Section 3 about the various designs for co-evolution of game-playing agents. In particular, they show that the idea that co-evolution works by setting up an “arms race” between the populations is not necessarily sound – for an arms race to take place and give the desired results, we require games in which there is a good correspondence between the true strength of the players (measured game-theoretically) and who beats whom. This is often not the case; in imperfect-information games this correspondence can be especially poor.

Therefore, we require a mode of fitness evaluation that enforces such a correspondence; our asymmetric design has this property.

In the experiments, we used a simple game where the solution is known, and used the calculated expected results of the match-ups in the fitness calculations, instead of results from actual game play. This was done to give a noise-free validation of our claims about the different co-evolutionary designs; for our method to be of practical interest – i.e. in games where the solution is not known – we obviously need to estimate the expected results by playing repeated games for each match-up. This, along with the fact that the number of matches in each generation increases, makes the algorithm relatively expensive in computational terms. This is, of course, a general problem with evolutionary algorithms, as these are rather blind searches compared to methods that glean information about the fitness terrain in more systematic ways.

The question, then, is when and why we should use co-evolutionary approaches, rather than more informed methods? One obvious answer is that they may be useful when other approaches fail, for instance when it is difficult to find agent representations amenable to other machine-learning techniques. Another situation in which co-evolution (and, indeed, evolution in general) is useful is when we specifically desire to use a certain representation that does not lend itself well to other approaches. As an example, we mention that we have work in progress on a far more complex game, where the individuals are small computer programs for playing the game, and the method of evolution is genetic programming [5]. The point of using this non-parametric representation is to evolve game-playing policies that are semantically understandable to humans; neural-net training, for instance, does not produce this kind of information in a readily accessible way.

All of our claims and conclusions in this paper are based on the goal of training agents that are strong in the game-theoretic sense; their ability to randomise strategies in a minimax-like way is the criterion for evaluation. We have already touched upon certain problems with this view, in particular the instabilities connected to the defensive nature of minimax strategies. This defensive approach may seem counter-intuitive to humans, as the goal of these strategies is to randomise in such a way as to be invulnerable to a possibly more intelligent opponent. Furthermore, they do not use information about their opponents, for instance information gleaned from previous games. A minimax strategy has only a weak ability of punishing vulnerable opponents; in fact, it is only expected to win if the opponent performs actions that are not a part of the optimal mixed strategy. Some other research, such as the work on poker reported in e.g. [13], has the more ambitious goal of using opponent modelling for exploiting the weaknesses of other agents. While there are certain problems with this approach, such as the lack of theoretically sound performance measures, the work is indeed very interesting. In a nutshell we can say that an agent trained in this way assumes that it can become more intelligent than its opponents, and thus be able to beat them, while a minimax-trained agent assumes that it will meet more intelligent strategies, and prepares for the worst.

6 Conclusion

We have presented an asymmetric co-evolutionary learning algorithm for imperfect-information zero-sum games. This algorithm has been designed so that the fitness of the individual agents is calculated in a way that is compatible with the goal of game-theoretic optimality. This compatibility has been somewhat lacking in previous co-evolutionary approaches, as these have often depended on unwarranted assumptions about the absolute and relative strength of players. Our algorithm is seen to work well on a toy problem for which the optimal strategy is known.

References

1. Angeline, P.J., Pollack, J.B.: Competitive environments evolve better solutions for complex tasks. In: Forrest, S. (ed.): *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo (1993) 264–270.
2. Halck, O.M., Dahl, F.A.: On classification of games and evaluation of players – with some sweeping generalizations about the literature. In: Fürnkranz, J., Kubat, M. (eds.): *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing*, Jozef Stefan Institute, Ljubljana (1999).
3. Hofstadter, D.R.: *Metamagical Themas. Questing for the Essence of Mind and Pattern*. Basic Books, New York (1985).
4. Koller, D., Pfeffer, A.: Representations and solutions for game-theoretic problems. *Artificial Intelligence* **94**(1) (1997) 167–215.
5. Koza, J.R.: *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT press, Cambridge, Massachusetts (1992).
6. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, New Brunswick (1994) 157–163.
7. Luce, R.D., Raiffa, H.: *Games and Decisions*. Wiley, New York (1957).
8. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended edition*. Springer-Verlag, Berlin–Heidelberg–New York (1996).
9. von Neumann, J.: Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen* **100** (1928) 295–320.
10. von Neumann, J., Morgenstern, O.: *Theory of Games and Economic Behavior*. Princeton University Press, Princeton (1944).
11. Pollack, J.B., Blair, A.D.: Co-evolution in the successful learning of backgammon strategy. *Machine Learning* **32**(3) (1998) 225–240.
12. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. *Evolutionary Computation* **5**(1) (1997) 1–29.
13. Schaeffer, J., Billings, D., Peña, L., Szafron, D.: Learning to play strong poker. In: Fürnkranz, J., Kubat, M. (eds.): *Proceedings of the ICML-99 Workshop on Machine Learning in Game Playing*, Jozef Stefan Institute, Ljubljana (1999).
14. Strang, G.: *Linear Algebra and Its Applications. Second Edition*. Harcourt Brace Jovanovich, Orlando (1980).

A Machine Learning Approach to Workflow Management

Joachim Herbst

DaimlerChrysler AG, Research and Technology
P.O. Box 2360, 89013 Ulm, Germany
joachim.j.herbst@daimlerchrysler.com

Abstract. There has recently been some interest in applying machine learning techniques to support the acquisition and adaptation of workflow models. The different learning algorithms, that have been proposed, share some restrictions, which may prevent them from being used in practice. Approaches applying techniques from grammatical inference are restricted to sequential workflows. Other algorithms allowing concurrency require unique activity nodes. This contribution shows how the basic principle of our previous approach to sequential workflow induction can be generalized, so that it is able to deal with concurrency. It does not require unique activity nodes. The presented approach uses a log-likelihood guided search in the space of workflow models, that starts with a most general workflow model containing unique activity nodes. Two split operators are available for specialization.

1 Introduction

The success of today's enterprises depends on the efficiency and quality of their business processes. Software based tools are increasingly used to model, analyze, simulate, enact and manage business processes. These tools require formal models of the business processes under consideration, which are called workflow models in the following. Acquiring workflow models and adapting them to changing requirements is a time consuming and error prone task, because process knowledge is usually distributed among many different people and because workflow modeling is a difficult task, that needs to be done by modeling experts (see [1,5] or [9]). Thus there has been interest in applying machine learning techniques to induce workflow models from traces of manually enacted workflow instances. The learning algorithms, we are aware of, share some restrictions, that may prevent them from being used in practice. They either apply grammatical inference techniques and are restricted to sequential workflows [5,9] or they allow concurrency but require unique activity nodes [1,6].

2 Definitions

In the following we define the terms *workflow model* and *workflow instance*. This is essential for a description of the induction task. A workflow model is a

formal explicit representation of a business process, describing how this process is (or should be) performed. It decomposes the process into elementary activities and defines their control and data flow. The activities $A = \{a_1, \dots, a_n\}$ of the process are specified in terms of their required resources and actors. Different formalisms have been proposed for workflow modeling. Within this paper we are using the ADONIS modeling language [3]. According to the ADONIS modeling language a workflow model can be defined as follows: A *workflow model* is a tuple $M = (V_M, f_M, R_M, g_M, P_M)$, where $V_M = \{v_1, \dots, v_{n_M}\}$ is a set of nodes, $\text{START}_M, \text{ACT}_M, \text{DEC}_M, \text{SPLIT}_M, \text{JOIN}_M, \text{END}_M$ is a partition of V_M , $f_M : \text{ACT}_M \rightarrow A$ is the activity assignment function, that assigns an activity to each activity node, $R_M \subseteq (V_M \times V_M)$ is a set of edges, $P_M : R_M \rightarrow [0, 1]$ assigns a transition probability to each edge and $g_M : R_M \rightarrow \text{COND}$ assigns a condition to each edge. This definition is incomplete, as it concentrates on the behavioral and functional view (see [7]) on a workflow model. For a complete definition describing also the organizational and informational view [7] as well as a discussion of additional syntactical rules and the semantics of the modeling language we refer to [3]. For our purposes the above definition and figure 1 showing the graphical representation of the node types and a brief explanation of their semantics should be sufficient. An example for a workflow model is given in figure 2.

Graphical Representation	Node Set	Explanation
	START	Starting node of a workflow model.
	ACT	An activity node.
	SPLIT	An m of n split. (m of n successors may be activated)
	JOIN	Join nodes synchronize the concurrent threads of their corresponding split nodes
	DEC	A decision node. (Exactly 1 of n successors may be activated)
	END	End node of a workflow model.

Fig. 1. ADONIS node types

A *workflow instance* is a tuple $e = (K_e, f_e(), \leq_e)$, where $K_e = \{k_1, \dots, k_{n_e}\}$ is a set of nodes, $f_e : K_e \rightarrow A$ is the activity assignment function, which assigns an activity to each node and $\leq_e$ is a partial order on K_e . Workflow instances represent a completed business cases. The nodes describe the activities, which were executed to complete a business case and the partial order describes the temporal order of their execution. For the sake of clarity, we define only those components, of a workflow instance which are relevant for this paper. Two exam-

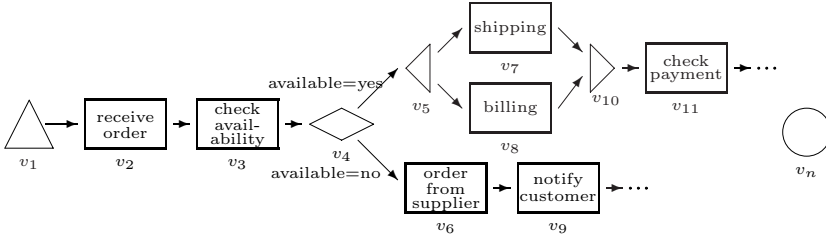


Fig. 2. Part of a simple ADONIS workflow model

ples for workflow instances are shown in figure 4. Activity nodes are represented by boxes, which are labeled with the values of the activity assignment function.

3 Inducing Workflow Models

3.1 Characterization and Decomposition of the Induction Task

The induction task to be solved can be characterized as follows: Given a multiset of workflow instances E , find a good approximation M of the workflow model M_0 , that generated E . Of course M_0 need not exist. It is simply a modeling hypothesis. We have decomposed the induction task into two subtasks:

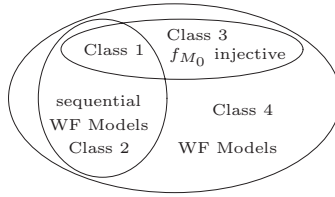
- Induction of structure - within this subtask the nodes, the edges, the activity assignment function and the transition probabilities of M are induced.
- Induction of conditions - where possible, local conditions for transitions following a split or a decision node are induced.

In this contribution we focus only on the induction of the structure. The induction of conditions can be done using standard decision rule induction algorithms such as C4.5 [12] as explained in more detail in [9].

3.2 Problem Classes

To allow a comparison between different workflow induction algorithms reported in the literature, we have defined four problem classes. These are defined in terms of two characteristics of the unknown workflow model M_0 . The first characteristic is sequentiality. A workflow model is strictly sequential, if it does not contain any split or join nodes. The second characteristic is a characteristic of the activity assignment function f_{M_0} . As we will see, it is a difference whether f_{M_0} is injective or not. If the activity assignment function is injective, then the unknown workflow model M_0 contains unique nodes for each observed activity. Using these two characteristics the four problem classes shown in figure 3 can be defined.

Actually it would be sufficient, to solve the induction task for the most general problem class, which contains all other problem classes. But we are not aware of

**Fig. 3.** Problem classes

any induction algorithm in the literature, that attempts to solve the induction task for this class. In this paper we will shortly discuss the induction of sequential workflow models and in more detail we will explain how this algorithm can be generalized to provide a solution to the problem classes three and four.

3.3 Sequential Workflow Models: Problem Classes 1 and 2

The structure of sequential workflow models can be represented by stochastic finite state automatons (SFA) and sequential workflow instances can be seen as strings over a finite alphabet. Each symbol in this alphabet corresponds to an activity of the workflow instance. Thus the problem of sequential workflow structure induction can be reduced to the problem of inducing SFAs from a positive sample of strings. This problem has already been addressed in the grammatical inference community (see e.g. [11]) and some algorithms like e.g. ALERGIA [4] or Bayesian Model Merging [14], have been proposed. In [9] we present two algorithms for sequential workflow induction. The first one follows a specific to general approach. It is a variation of the Bayesian Model Merging [14], using the log-likelihood of the workflow model as a heuristic. The second one follows a general to specific approach. For specialization it applies a split operator that splits one node into two nodes assigned to the same activity. Search starts with a most general model, containing unique nodes for each observed activity. The solution to problem class 4, which we present below, follows the same basic principle.

3.4 Concurrent Workflow Models with Unique Activity Nodes: Problem Class 3

Let's now turn to concurrent workflow models having unique activity nodes. For each activity $a_i \in \{a_1, a_2, \dots, a_n\}$ we observe, we create a unique node v_i with $f_M(v_i) = a_i$. This gives us the set of activity nodes $\text{ACT}_M = \{v_1, \dots, v_n\}$ of the workflow model M . Whenever we observe the occurrence of an activity a_i , we can identify the corresponding activity node v_i of M . This allows us to talk about the "occurrence of a node v_i within an instance e ".

But as the workflow model may contain concurrent threads we may not determine the activity node, whose completion triggered the current observed activity, as easily as in the sequential case, where we considered the immediate predecessor to be the cause for an observed activity (see [9]). This is not adequate

in case concurrent threads of control are possible. First of all the cause for an observed activity is not necessarily its immediate predecessor and also there may be more than one cause for an activity (e.g. after a join construct). This is shown in figure 4. The workflow instances e_1 and e_2 may have been generated by the workflow model M_0 . In this case the cause for activity D within instance e_2 is activity C, and not its immediate predecessor B.

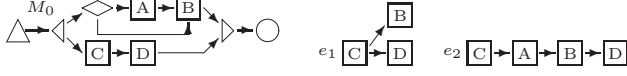


Fig. 4. Unknown model M_0 and observed workflow instances e_1 and e_2

Before we can add edges to M , we must find the cause for an observed activity. This leads us directly to the task of detecting dependencies between activities. This is done by analyzing the temporal relationships between activities. For the following definition to be well defined, we need to assume, that the unknown model M_0 is acyclic. We will later eliminate this restriction. This assumption assures that no activity occurs more than once within a workflow instance. We can now define the dependency graph as the directed graph $G_{\text{dep}} = (V_M, R_{\text{dep}})$ with

- $V_M = \text{START}_M \cup \text{ACT}_M \cup \text{END}_M$ with $\text{START}_M = \{v_0\}$, $\text{END}_M = \{v_{n+1}\}$
- $R_{\text{dep}} = \{(v_i, v_j) \mid \forall e \in E (v_i, v_j \text{ appear in } e) \Rightarrow v_i \text{ precedes } v_j \text{ in } e\}$, v_0 and v_{n+1} implicitly occur within every e . v_0 is a predecessor of any node within every e and every node occurring within e is a predecessor of v_{n+1} .

The dependency graph can be determined in one pass over the sample E . If we observed all possible instances that could be generated from M_0 the dependency graph shown in figure 5 would be found. We also define dependency graphs $G_e = (V_M(e), R_{\text{dep}}(e))$ for each instance e . G_e is the subgraph of G_{dep} containing only those nodes occurring in e and all edges between them. The dependency graphs for the instances e_1 and e_2 are depicted on the top right of figure 5. Using the dependency graphs G_e , we now determine the cause graphs $(G_e)_*$. The cause graph $(G_e)_* = (V_M, R_{\text{dep}}(e)_*)$ is the transitive reduction of G_e . The transitive reduction of a directed graph G is defined as a minimal subgraph of G having the same transitive closure as G . In this case the transitive reduction of G_e is unique because G_e is acyclic and it can be efficiently determined, because a topological ordering of the nodes in G_e is indirectly given by the partial order $\leq_e$ of the workflow instance. The cause graphs for each workflow instance can be calculated in a second pass over the sample E . The cause graphs for e_1 and e_2 are shown on the bottom left of figure 5. We can now determine the set of edges R_M of M as $R_M = \bigcup_{e \in E} R_{\text{dep}}(e)_*$.

Let's drop the assumption that M_0 is acyclic. Now an activity a_i may appear more than once within an instance e . We simply distinguish different occurrences

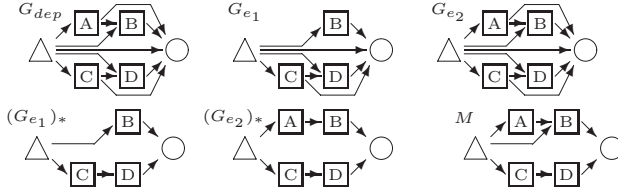


Fig. 5. Dependency graphs, cause graphs and induced model

of one and the same activity within one instance by adding an index (1st, 2nd, 3rd, ... occurrence). We then apply the same algorithm and treat different occurrences of the same activity as different activities until the edges of the model have been determined. At this point all nodes belonging to the same activity are merged to one node, that inherits the edges of all merged nodes.

To complete M we finally add explicit control flow constructs (Decision, Split and Join) to the model M where necessary. This step is not as trivial as in the sequential case (compare [9]), because dependencies between different edges must be analyzed. Within this paper we will not elaborate on this task any further.

3.5 Concurrent Workflow Models in General: Problem Class 4

For problem class 4 M_0 may contain more than one node for a specific activity. The basic idea of our solution is the same as the splitting approach presented in [9] for sequential workflows. One starts with the most general model, generated by the algorithm of the previous section, called `induceUniqueNodeModel()` in the following. This is like assuming that M_0 is in problem class 3. The most general model is specialized using split operators. The selection of the state to split and of other parameters is guided by the log-likelihood per sample. In our prototype we are using beam-search as search algorithm. A larger model (containing more nodes) is preferred over a smaller model, only if the log-likelihood per sample is larger than some user defined threshold $LLH_{\min}$.

Probability of a Sample The likelihood heuristic requires to estimate the probability of E given M . For this purpose one could describe all outgoing edges of a node v_i as a n of m selection. To distinguish decision nodes allowing only a 1 of m selection from split nodes we decided to use better estimation, that considers clusters of nodes. These are defined in a way that all nodes v_j sharing a common cause v_i within any instance are contained in a common cluster C_{ik} ¹. This idea is shown in figure 6. In the example of figure 6 the probability that the activities B and C follow the activity A would be calculated for example as $0.5 \cdot (1 \cdot 0.5 \cdot (1 - 0.5))$. The transition probabilities are estimated by the empirical counts.

¹ This is still a simplification because dependencies may be more complex. One might for example be interested in finding exclusive successors within one cluster

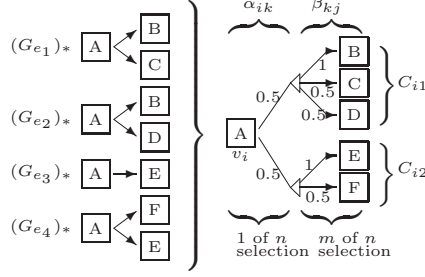


Fig. 6. Clustering of nodes

Specialization of Concurrent Workflow Models For sequential workflow induction [9] we defined the split operator as an operator on the workflow model. The effects of a split operation on a concurrent model are not restricted to the incoming and outgoing edges of the node that is split. Global effects are possible if the split operation changes the dependencies. It is thus not clear how to find a simple description for a split operator on concurrent workflow models. To prevent these difficulties, we define the split operator as an operator on the workflow instances, that introduces an artificial distinction between certain occurrences of the activity that is split. After a split operator has been applied to all instances in E , `induceUniqueNodeModel()` called with the changed instances E to return a specialized model.

Split Operators For the specialization of the workflow model we initially defined one split operator $\text{Split}_{\text{Cause}}(e, a_i, a_j, (G_e)_*)$ as:

$$\forall k \in K_e \text{ with } f_e(k) = a_i \text{ let } f_e(k) := \begin{cases} a'_i & \text{if } a_j \text{ is a cause of } a_i \\ a''_i & \text{otherwise} \end{cases}$$

While one split operator based on the cause of a node is sufficient for sequential workflows, it is sometimes not applicable for concurrent workflows. When multiple activity nodes are allowed, dependencies are often not identified correctly until the right degree of specialization has been reached. This may have the consequence that a certain cause for an activity, can not be correctly identified. If this cause is necessary to correctly distinguish different occurrences of an activity, $\text{Split}_{\text{Cause}}$ fails. To deal with this problem, we define a second split operator $\text{Split}_{\text{History}}(e, a_i, a_j)$ as:

$$\forall k \in K_e \text{ with } f_e(k) = a_i \text{ let } f_e(k) := \begin{cases} a'_i & \text{if } a_j \text{ is a predecessor of } a_i \text{ in } e \\ a''_i & \text{otherwise} \end{cases}$$

4 Related Work

In [1] an approach called process mining, based on the induction of directed graphs, is presented. It is restricted to problem class 3 and very similar to our

approach for this problem class. The main differences to our approach lie in the representation of workflow instances as strictly ordered sets of activities and in the way dependencies are defined and determined. Another approach that is also restricted to problem class 3 is presented in [6]. It uses three different metrics for the number, frequency and regularity of event sequences to estimate a model of the concurrent process. In their previous work [5] the authors applied different grammatical inference algorithms to sequential workflows.

Different approaches combining machine learning and workflow management techniques are presented by Wargitsch [15] and by Berger et. al. [2]. Both are using completed business cases to configure new workflows. While Wargitsch employs a case-based reasoning component for the selection of an appropriate historical case, Berger et. al. are using a neural network approach.

Workflow induction has some similarity with the mining of temporal patterns presented in [13] or [10]. But while we are trying to find one structure in a relatively structured event trace, these approaches are trying to find all frequent structures and they are applicable only for unstructured event traces, as their performance scales exponentially with the size of the largest structure found.

5 Prototype and Experiences

We have realized a research prototype using the business process management system ADONIS both as a front end for the generation of artificial workflow instances and as a back end for the layout generation and visualization of the induced workflow models. We applied this prototype to workflow traces generated by different types of workflow models. Some of these models are from the literature (see e.g. [1] or [6]), some of them are workflow models we have defined and others are real workflow models we have encountered within workflow projects at DaimlerChrysler. In [8] we describe the application of our approach to a simplified release process of the Mercedes Benz passenger car department. Tables 1 and 2 show comparisons with process mining [1] and with process discovery [6]. As the original samples were not available, we generated our own samples. This of course has an influence on the results.

Table 1. Workflow splitting applied to workflow models reported in [1]

Model	Nodes / Edges	Nr. splits	Nr. samples	Process Mining		Workflow Splitting	
				time	correct?	time	correct?
Upload_And_Notify	11/11	0	134	11.5s	yes	0.8s	yes
StressSleep	18/27	0	160	11.7s	yes	5.6s	yes
Pend_Block	10/11	0	121	6.3s	yes	0.8s	yes
Local_Swap	14/13	0	24	5.7s	yes	0.2s	yes
UWI_Pilot	11/11	0	134	11.8s	yes	0.8s	yes

The comparison with process mining shows that exactly the same models are found, which is not surprising as the algorithms are very similar. The improvement concerning the performance might be caused by the slightly different definition of dependency we are using, which allows a more efficient algorithm for dependency detection. Actually our approach should be less efficient, because process mining is restricted to workflow models of problem class 3 and does not try to split any nodes. The models described in [6] were initially not identified correctly by our approach. They contained some incorrect edges. The reason for these incorrect edges is that both models contain concurrent activities within cycles. With some probability only a few samples are available for those workflow instances with the highest number of iterations over a certain cycle. In this case it is likely that not all possible orderings of activities are observed for this highest iteration. This may lead to an incorrect dependency graph and as a consequence to an incorrect model. As these incorrect edges are characterized by a probability close to zero they can be identified and removed from the model. This enables our approach to induce these models correctly as well.

Table 2. Workflow splitting applied to workflow models reported in [6]

Model	Nodes / Edges	Nr. splits	Nr. samples	Process Discovery		Workflow Splitting	
				time	correct?	time	correct?
simple concurrent Process	11/12	0	300	?	yes	126.8s	(yes)
complex concurrent Process	22/26	0	150	?	yes	194.7s	(yes)

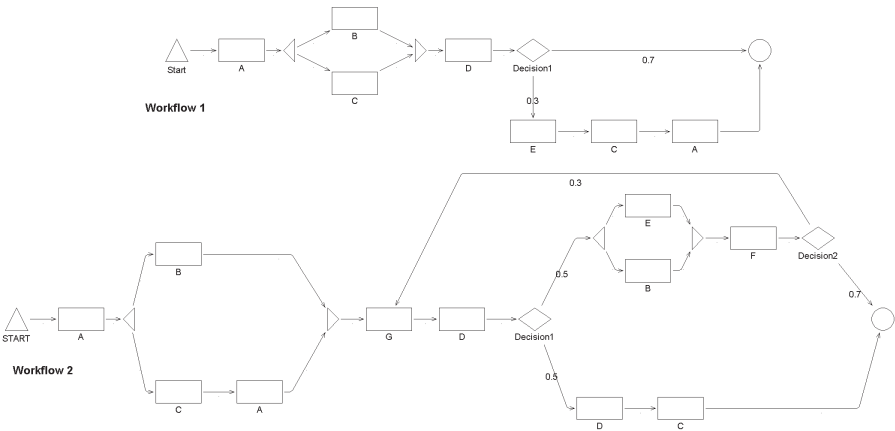


Fig. 7. Two Workflow Models used for evaluation

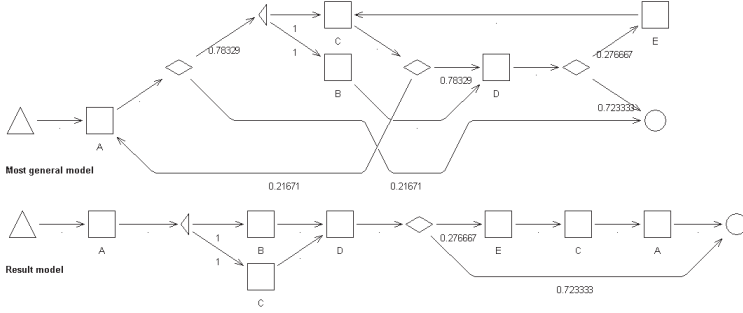


Fig. 8. Workflow 1: Most general model and result model after applying $\text{Split}_{\text{Cause}}(e, A, C, (G_e)_*)$ and $\text{Split}_{\text{Cause}}(e, C, E, (G_e)_*)$

The workflow models presented in [1] and [6] are all located in problem class 3. To evaluate the specialization procedure we also applied our approach to workflow models of problem class 4. Two examples for such workflow models are given in figures 7.

When observing a large enough sample generated from workflow 1 the most general model depicted at the top of figure 8 would be induced. Given the right choice for $\text{LLH}_{\min}$, after one intermediate step our search procedure would return the model shown at the bottom of figure 8. Any further split operations lead only to a small improvement of the log-likelihood per sample.

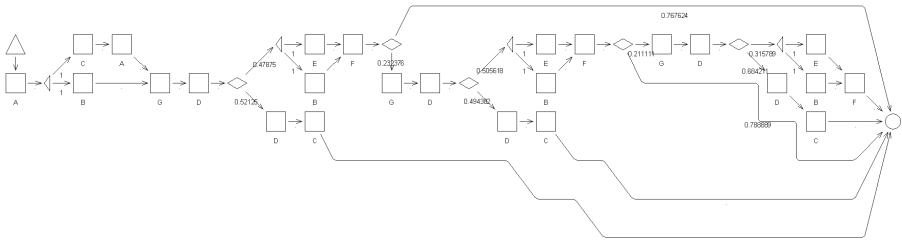


Fig. 9. Workflow 2: Overly specific model using $\text{LLH}_{\min} = 0$

The degree of specialization depends on the user defined threshold $\text{LLH}_{\min}$. Overly specialized models will for example be found if this threshold is too small. The effect of overspecialization is shown in figure 9. The cycle present within workflow 2 of figure 7 has been “unrolled”. Figure 10 shows the log-likelihood per sample for those models on the search path from the most general model to the model of figure 9. As you can see, only the first four (from left to right) split

operations significantly improve the log-likelihood per sample. After the fourth split, the log-likelihood per sample remains nearly constant. Thus the threshold $LLH_{\min}$ must be chosen within the right range close to zero, so that the search stops after the fourth split and returns the correct model shown in figure 11.

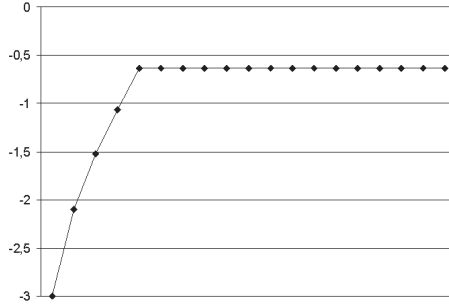


Fig. 10. Log-likelihood per sample of the models on the search path

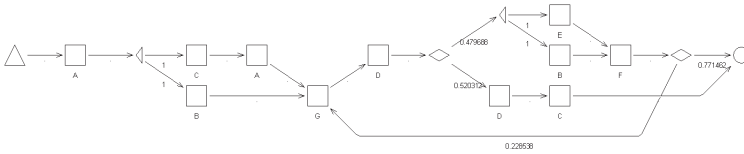


Fig. 11. Workflow 2: Result model using $LLH_{\min} = 0.1$

6 Summary and Future Work

We have presented a learning algorithm that is capable of inducing concurrent workflow models. This approach does not require unique activity nodes as other workflow induction algorithms do. We are convinced that the integration of workflow induction algorithms such as ours has the potential to provide a number of significant improvements to workflow management systems, including a shorter acquisition time for workflow models, higher quality workflow models with less errors and support for the detection of changing requirements.

Further work must be done to deal with noise, caused for example by erroneous workflow instances. Noise is especially critical if the dependency structure is affected. We are also working on algorithms that add explicit control flow constructs (Decision, Split and Join) to the induced model.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Proc. of the sixth International Conference on Extending Database Technology (EDBT)*, 1998. 183, 189, 190, 192
2. M. Berger, E. Ellmer, and D. Merkl. A Learning Component for Workflow Management Systems. In *Proceedings of the 31st Hawaii International Conference on System Sciences (HICSS-31)*, pages 754–763. IEEE, 1998. 190
3. BOC. *ADONIS Version 3.0 - Users Guide*. BOC GmbH, Vienna, 1999. 184
4. R. C. Carrasco and J. Oncina. Learning Stochastic Regular Grammars by Means of a State Merging Method. In *Second International Colloquium on Grammatical Inference and Applications (ICGI94)*, volume 862 of *Lecture Notes on Artificial Intelligence*, pages 139–152, Berlin, Germany, 1994. Springer-Verlag. 186
5. J.E. Cook and A.L. Wolf. Automating Process Discovery through Event-Data Analysis. In *Proc. of the 17th International Conference on Software Engineering*, pages 73–82. Association for Computer Machinery, 1995. 183, 190
6. J.E. Cook and A.L. Wolf. Event-Based Detection of Concurrency. Technical Report, CU-CS-860-98, Dep. of Computer Science, University of Colorado, 1998. 183, 190, 191, 192
7. B. Curtis, M.I. Kellner, and J. Over. Process Modeling. *Communications of the ACM*, pages 75–90, September 1992. 184
8. J. Herbst. Inducing Workflow Models from Workflow Instances. In *Proceedings of the 6th European Concurrent Engineering Conference*, pages 175–182. Society for Computer Simulation (SCS), 1999. 190
9. J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. In *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications*, pages 745–752. IEEE, 1998. 183, 185, 186, 188, 189
10. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997. 190
11. R. Parekh and V. Honavar. Automata Induction, Grammar Inference, and Language Acquisition. In Dale, Moisl, and Somers, editors, *Handbook of Natural Language Processing*. New York: Marcel Dekker, 1999. 186
12. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. 185
13. R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proc. of the Fifth International Conference on Extending Database Technology (EDBT)*, pages 3–17. Springer, 1996. 190
14. A. Stolcke and S. Omohundro. Best-First Model Merging for Hidden Markov Model Induction. Technical Report, TR-94-003, Int. Computer Science Institute, 1994. 186
15. C. Wargitsch. WorkBrain: Merging Organizational Memory and Workflow Management Systems. In *Workshop of Knowledge-Based Systems for Knowledge Management in Enterprises at the 21st annual German Conference on AI (KI-97)*, pages 214–219, Kaiserslautern, Germany, 1997. DFKI. 190

The Utilization of Context Signals in the Analysis of ABR Potentials by Application of Neural Networks

Andrzej Izworski¹, Ryszard Tadeusiewicz¹, and Andrzej Pasławski²

¹ University of Mining and Metallurgy, Dept. of Automatics, Laboratory of Biocybernetics
Al. Mickiewicza 30, 30-059 Cracow, Poland
{izwa,rtad}@biocyb.ia.agh.edu.pl

² University of Mining and Metallurgy, Dept. of Automatics
Al. Mickiewicza 30, 30-059 Cracow, Poland

Abstract. The elaboration of head-surface registration techniques for auditory potentials evoked from the brainstem (ABR) enabled the construction of objective research and diagnostic methods, which can be utilized in the examinations of auditory organs. The aim of the present work was the construction of a method, making use of the neural network techniques, enabling an automated detection of wave V in the ABR signals. The basic problem encountered in any attempts of automated analysis of the auditory potentials is connected with impossibility of a reliable evaluation of a single response evoked by a weak acoustic signal. It has been assumed that considerably better detection results should be obtained, when additional context information will be provided to the network's input. This assumption has been verified using complex, hybrid neural networks. As a result about 90% of correct recognitions has been achieved

1. Introduction

The registration and analysis of the ABR (Auditory Brainstem) potentials enables an objective evaluation of functions of both the mechanical part of the auditory system as well as the analysis of processes taking place in the specific levels of the neural part of that system. The registration of the ABR potentials is of particular importance in those cases when the application of the classical audiometric methods is difficult or even impossible.

The typical time dependence of the ABR potential consists of five to seven waves, labeled by the respective roman numbers (I-VII), extracted from the EEG signal by the synchronous averaging and registered within 10 or 12 ms from the application of the acoustic stimulus. In the medical evaluation of the ABR potential mostly the latency period of wave V and the I-V time distance are taken into account. The absence of any wave, particularly the wave V, is also of great diagnostic importance, and the measurement of the product of the amplitudes of the waves V and I is an important indicator used in the evaluation of regularity of processes taking place in the neural part of the auditory system. The above description of forms and methods of processing the ABR potentials in the auditory system is neither complete nor exhaustive, it is however easy to notice that the diagnostic value is mostly represented by the quantities related to the wave V, so its automated detection and localization is an important scientific challenge and a research goal of great practical importance.

2. The Research Basis

The research described in the present work have been carried out mostly in the field of analysis and processing of the ready ABR potential signals registered previously in a clinic. However for introducing the experimental conditions, to which the results described below should be referred, it is necessary to present a few information concerning the applied methodology of inducing and registration of the studied signals.

As it is known, the determination of hearing sensitivity using the ABR potentials consists of the observation of decrease of the amplitude and increase of the latency period for the wave V, for a sequence of ABR signals registered for stimuli of gradually reduced intensity (e.g. from 110 to 20 dB with 10dB step). The examination goes on till the wave V totally disappears, what denotes the situation of total lack of signal reception by the patient. The shapes of the reference ABR signals, obtained in the specified conditions for the persons with correct auditory modality, are known (see Fig.1). It is also known that the changes in recording of the ABR signal for persons with the brainstem auditory centers deficiency, consist of deformation of the shape and eventually the disappearance of the wave V.

The basic problem encountered during the attempts of automated analysis of the ABR signals is the fact that the registered signal usually considerably deviates from the reference signal shown in Fig.1.

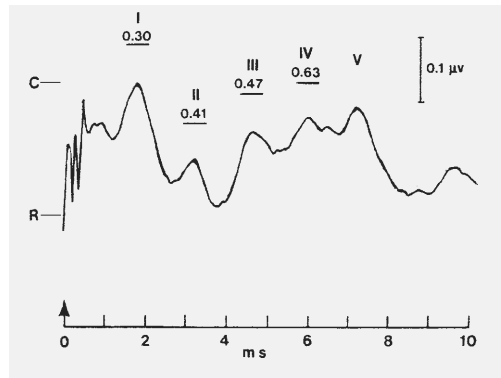


Fig. 1. Typical, singular signal of the ABR

In the case of low levels of the signal stimulating the response distinguishing between particular waves in the ABR recording can be very difficult. The previous works by the authors [4, 5] have shown that no algorithm can be constructed, able to perform the task in an automated way, and also that it is extremely difficult to construct and train a neural network, which could be able to determine from evaluation of a single recording of an ABR invoked by a weak auditory signal, the presence or absence of the wave V in the studied signal and where it is located.

It has been found that better results are obtained by physicians by evaluation of ABR result from the fact, that very often they make use of the context, i.e. evaluating a single run they make use of the neighboring runs. A working hypothesis has been

formulated, stating that artificial neural networks can also achieve considerably better correctness of the wave V detection and localization in the recordings of ABR signal, if **context information** is fed to their inputs, e.g. the ABR signal obtained for the previous (higher) amplitude of the acoustic stimulus.

The research, oriented towards verification of truth of the hypothesis formulated above, have been carried out according to the following research assumptions:

- it was assumed, that although there is a whole set of methods of ABR analysis, the present study will be concentrated exclusively on the attempt of determination whether the wave V is present or not in a given ABR signal,
- it was assumed that the tool used for detection of the wave V will be an artificial neural network of the multilayer perceptron structure,
- it was assumed that two signals will be fed to the network's input: the signal of the analyzed ABR **and** the signal used as a context,
- it was assumed that the data source will be provided by the set of several hundreds of ABR signals registered in the clinical conditions and offered for the present study due to courtesy of the Institute of Control Systems in Katowice.

In accordance to the previous research by the authors [4,5] neural networks have been used for detection of the presence of wave V. The decision followed from the fact, that neural networks are successfully applied for a long time in various, often very diverse areas [1,2]. In papers by other authors [7,8,9] their usefulness has been also proved in the field of medicine.

3. The Objective of the Study and Way to Achieve It

In table I selected results are shown of the automated classification of the ABR signals obtained in the previous research stages, oriented towards recognition of isolated signals by the artificial neural networks. The network's input have been fed with a signal describing the analyzed ABR recording (100 points) and the network's output a single logical-type signal was expected, indicating the presence or absence of the wave V in the input signal. The studied network architectures exhibited the 100-n-1 structure (where n denotes the size of the optimized hidden layer) or alternatively 100-n-m-1 structure, for the cases when networks with two hidden layer were applied.

Starting from the observation that a physician undertaking the analysis of recorded ABR signals does not analyze the signals separately but in the context of the accompanying signals (the person sees the whole series of recorded signals obtained for gradually decreasing intensity of the acoustic stimulus), an original technique of the context analysis of the considered signals have been proposed and applied. In the present work for the first time results are shown for a study, in which the authors attempted to take the context into account in the process of classification of ABR signals by the artificial neural networks.

In order to take the context into account in the described study two signals have been fed at the same time to networks input: the presently analyzed ABR signal and the previous, accompanying signal, obtained for the higher intensity of the acoustic stimulus. It could have been achieved using a simple multilayer neural network, but

additionally attempts have been made to use in the recognition process neural networks of some more complicated architecture.

4. The Considered Architectures of the Neural Networks Studied

In the task of automated recognition of the ABR potentials artificial neural networks have been applied, learned by the error backpropagation methods. Two classes of signals have been considered: the class of signals in which wave V was present and the class of signals where wave V was absent. The simplest architecture (Fig.2), which provides the possibility of making use of the context during the recognition of signals of auditory response is a neural network to the input of which two signals are fed in sequence. The considered architectures effectively exhibited the $200-n-1$ or $200-n-m-1$ structures.

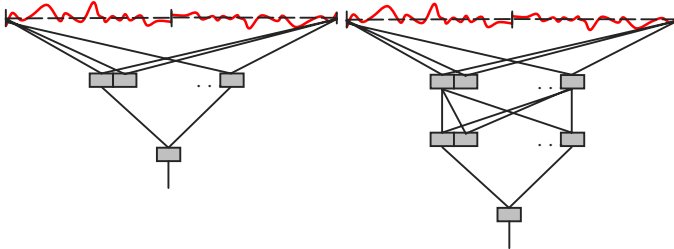


Fig. 2. The double and triple layer neural networks, in which the neurons are connected according to the "every with each other", doublets of signals are fed to the network's input

The network architecture described above have been later modified in such a way, that the first hidden layer has been split into two parts, and then two component signals of the input vector have been fed separately to each of the layers (Fig.3).

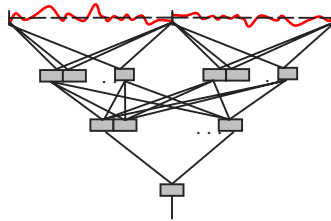


Fig. 3. Triple-layer network, the first hidden layer has been split, so that two consecutive ABR signals are fed separately to the network

Due to such a procedure the split layers of the hidden layer preliminary process the signal to be recognized and its context signal, working independently.

In the following step the first hidden layer can be split into more layers, and the **respective signal parts** of the two component signals of the input vector should be fed to individual layers. In that case each separated group of neurons can analyze the similarity of different fragments of the signal to be recognized and the context signal. The network of such architecture is presented in Fig.4.

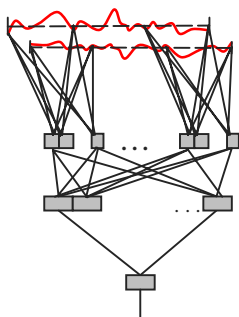


Fig. 4. Triple-layer network, the first hidden layer split into several groups of neurons, to each of them the respective parts of both signals (of equal lengths) are fed

Another possibility is to use only the information of the context signal recognition during the classification of the signal, contrary to the previous case when the whole context signal has been used. In such a situation the context signal should be previously classified by an independent network and then the output signal of that network should be fed to input of the main network together with signal to be classified. Such a network exhibiting a cascade structure is presented in Fig.5.

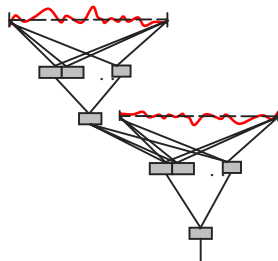


Fig. 5. System consisting of two neural networks, each of them trained separately. The first networks classifies a single signal, and the second network's input is provided with the signal itself and the information about the classification of the context signal

5. The Data Used for Evaluation of the Utility of Studied Networks

The data concerning the acquisition techniques for the ABR potentials analyzed in the present work have been as follows: the patient have been applied an acoustic stimulus in the form of a cracking noise of the intensity between 70 and 20 dB, and next from the EEG signal the ABR signal has been extracted. The original size of the signal included 1000 digitally processed values (covering the 10ms time period of the signal), but next it was reduced by the proper averaging techniques to 100 values, providing the input data for the considered networks.

The input vectors necessary for the context studies have been constructed in such a way, that to each of the ABR signals has been appended in the front part by the

preceding signal, obtained in the same measuring sequence but for the higher amplitude of the acoustic stimulus. For studied done using the network presented in Fig.5 the data sets have additionally preprocessed.. The resulting input vectors 192 points long (2 x 96 points) have been fed to the network's input.

6. The Obtained Results

In the course of the simulation the network's architecture has been optimized in order to provide the best results of the recognition. In table I several best results are shown (for comparison), obtained from the classification of single input signals. These results have already been published.

Table 1. Selected best results of the classification of single signals

No	NN architecture	Epochs	RMS error	Error of the classification [%]	
				Learning set	Test set
1	100 × 10 × 1	271	0.899	98.68	83.12
2	100 × 8 × 1	479	0.094	100.00	85.71
3	100 × 7 × 2 × 1	414	2.000	97.37	85.71
4	100 × 4 × 4 × 1	595	0.990	100.00	85.71

On the other hand the tables below the new results for the study of recognition of the ABR signals making use of the context signals.

From the completed research it follows that including the context has the strongest positive influence on the classification quality for the networks including one hidden layer, for which the improvement of the ABR recognition results was about 4-5%

The conclusion, which can be drawn is that by the addition to the recognized signal only the information about the classification of the signal preceding the analyzed signal leads to much worse effects than including the whole context signal. It has also turned out that application of more complex network architectures of the neural networks does not lead to the increase of ABR signal recognition quality.

Table 2. Selected best results of the classification for the networks presented in Fig.2 and 3

No	NN architecture	Epochs	RMS Error	Error of the classification [%]	
				Learning set	Test set
1	200 × 10 × 1	330	1.98	96.05	87.01
2	200 × 8 × 1	395	1.24	96.05	89.61
3	200 × 8 × 1	365	1.43	97.37	88.31
4	200×7×2×1	1049	3.94	94.74	85.71
5	(100+100)×(4+4)×2×1	2080	3.87	94.74	87.01
6	(100+100)×(3+3)×2×1	1794	3.88	94.74	87.01
7	(100+100)×(3+3)×2×1	2250	3.87	94.74	88.31

Table 3. Selected best result of the classification for the networks shown in Fig.4

No	NN architecture	epochs	RMS error	Error of the classification [%]	
				Learning set	Test set
1	$101 \times 10 \times 1$	1094	1.02	98.68	84.42
2	$101 \times 10 \times 1$	392	1.127	98.68	83.12
3	$101 \times 8 \times 1$	462	0.198	100.00	83.12

Table 4. Selected best results of the classification for the network shown in Fig.5

No	NN architecture	epochs	RMS error	Error of the classification [%]	
				Learning set	Test set
1	$(192) \times (3+3+3+3) \times 3 \times 1$	886	1.98	97.37	80.52
2	$(192) \times (4+4+4+4) \times 3 \times 1$	702	4.88	96.05	84.42
3	$(192) \times (3+3+3+3) \times 4 \times 1$	595	0.99	100.00	85.71

7. Conclusion

Summarizing the above considerations it can be concluded that the application of the input signal including the context of the analyzed ABR signal considerably improves the network's ability for recognition of the presence (or absence) of the wave V in the analyzed signal. It has been also found that increasing the network's complexity (by transition from triple layer to quadruple layer networks) does not lead to the expected improvement in the recognition quality, while considerably increasing the duration of the learning process. Neither have the expected results been obtained by the attempted optimization of the network's operation by splitting the hidden layer into two part analyzing separately the recognized signal and its context signal. The attempt to improve the network's operation by comparing the respective signal parts has not lead to satisfactory results either.

In spite of those - partly negative - results, it can be stated, that the application of the context signal was the reason that the considered task has found a more satisfactory solution, comparing to the case when the context was not taken into account. The neural network making use of the context data, which after the learning process were the best in classification of the ABR signals, have obtained the correct recognition in 88-89% cases. This result has to be regarded as satisfactory. It shown, that it is possible to build an automated system based on the neural networks, detecting (with the satisfactory recognition reliability) the presence of wave V in the ABR signal. It is the most important result of the study described here.

At the same time it was shown, that making use of the context signal during the automated recognition of ABR signals by artificial neural network is meaningful and leads to an effect observed as the improvement of the classification quality. This is the most essential cognitive effect of the described study.

References

1. Tadeusiewicz R.: Neural Networks, AOW RM, Warszawa (1993), (in polish)
2. Żurada J., Barski M., Jędruch W.: Artificial Neural Networks, PWN, Warszawa (1996), (in polish)
3. Aminoff M.J. (ed.): Elektrodiagnosis in Clinical Neurology, Churchill Livingstone New York, (1986)
4. Paślowski A., Izworski A.: Detection of the V wave in BAEP with use of artificial neural networks, III Conference on Acoustical Methods and Biomedical Engineering, Zakopane (1998), (in polish)
5. Izworski A.: Analysis of the Brainstem Auditory Evoked Potentials with Use Neural Networks, X Conference „Biocybernetics and Biomedical Engineering”, Warszawa, (1997) 808-813, (in polish)
6. Delgado R. E., Ozdmar O.: Automated Auditory Brainstem Response Interpretation, IEEE Engineering In Medicine And Biology, April/May (1994)
7. Tadeusiewicz R., Wszolek W., Izworski A., Modrzejewski M.: Application of Neural Networks in Diagnosis of Pathological Speech, In M. Heiss (ed.): Proceedings of the International ICSC/IFAC Symposium on Neural Computation NC '98, Vienna University of Technology, ICSC Academic Press, Canada/Switzerland, (1998) 1040-1045
8. Tadeusiewicz R., Wszolek W., Izworski A., Wszolek T.: Methods of deformed speech analysis, In: Manfredi C., Brusciaglioni P.: Models and Analysis of Vocal Emissions for Biomedical Applications, University of Firenze, (1999) 130-137
9. Tadeusiewicz R., Izworski A., Wszolek W., Wszolek T.: Processing and Classification of Deformed Speech Using Neural Networks, in: Fouke J.M., Nerem R.M., Blanchard S.M., Yoganathan A.P., (eds.): Serving Humanity, Advancing Technology, Proceedings of The First Joint BMES/EMBS Conference, Atlanta, (1999) 927-928

Complexity Approximation Principle and Rissanen’s Approach to Real-Valued Parameters

Yuri Kalnishkan*

Department of Computer Science, Royal Holloway, University of London
Egham, Surrey, TW20 0EX, United Kingdom
`yura@dcs.rhbnc.ac.uk`

Abstract. In this paper an application of the Complexity Approximation Principle to the non-linear regression is suggested. We combine this principle with the approximation of the complexity of a real-valued vector parameter proposed by Rissanen and thus derive a method for the choice of parameters in the non-linear regression.

1 Introduction

The Complexity Approximation Principle (CAP) was proposed in the paper [9] and it deals with the hypothesis selection problem. CAP is one of the implementations of the idea to trade-off the ‘goodness-of-fit’ of a hypothesis against its complexity. This idea goes back to the celebrated Occam’s razor and the scope of its implementations includes MDL and MML principles.

In this paper, we make an attempt to apply CAP to the choice of coefficients in the non-linear regression. The problem of evaluating the complexity of a real-valued vector emerges and we overcome it by adapting the approach proposed by Rissanen in [3]. We infer a formula that suggests a new estimate of the regression coefficients and it turns out to be a normalisation of the Least Squares (LS) estimate.

In Sect. 2 we formulate CAP in the form we need and describe the non-linear regression problem. In Sect. 3 we apply Rissanen’s approach and obtain the minimisation problem; in Sect. 4 and 5 we discuss possible solutions. Sect. 6 contains the description and the results of our computational experiments. We compare our results with other regression techniques.

2 Preliminaries

2.1 CAP

In this paper, the special case of CAP relevant to the batch settings and the square-loss measure of discrepancy is considered. We will now formulate CAP in this weak form.

* Supported partially by EPSRC through the grant GR/M14937 (“Predictive complexity: recursion-theoretic variants”) and by ORS Awards Scheme.

Suppose we are given a data sequences $z = ((x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)) \in (X \times \Omega)^*$, where $\Omega = [a, b]$ is the set of *outcomes* and X is the set of *signals*. Our goal is to find the decision rule $\mathcal{R} : X \rightarrow \mathbb{R}$ that suits the data best in a given class of decision rules $\mathfrak{R}$. The performance of $\mathcal{R}$ is assessed by some measure of loss or discrepancy $\lambda(\mathcal{R}(x), y)$, where y is the actual outcome which corresponds to the signal x . We assume that $\lambda(\mathcal{R}(x), y) = (\mathcal{R}(x) - y)^2$. We want $\mathcal{R}$ to perform well i.e. to suffer small loss on pairs (*signal, outcome*) $\in X \times \Omega$ that may arrive in the future.

The classical Least Squares (LS) approach suggests minimising the total square loss of $\mathcal{R}$ on the sequence z , i.e. $\text{Loss}_{\mathcal{R}}^{\text{sq}}(z) = \sum_{i=1}^l (\mathcal{R}(x_i) - y_i)^2$. A decision rule $\widehat{\mathcal{R}} \in \mathfrak{R}$ is called a *LS estimate* if the minimum

$$\min_{\mathcal{R} \in \mathfrak{R}} \text{Loss}_{\mathcal{R}}^{\text{sq}}(z) \quad (1)$$

is attained at $\widehat{\mathcal{R}}$.

LS works perfectly well in many applications unless the problem of overfitting occurs. The given data z may be influenced by noise or round-off error so following carefully all the peculiarities of our data we may end up with an estimate which makes no sense. That is why the idea to penalise the growth of complexity of $\mathcal{R}$ emerges. Instead of finding the minimum (1), one may search for $\mathcal{R}$ minimising

$$\min_{\mathcal{R} \in \mathfrak{R}} (\text{Loss}_{\mathcal{R}}(z) + \mathcal{K}(\mathcal{R})) \quad , \quad (2)$$

where $\mathcal{K}$ is some measure of complexity of $\mathcal{R}$. In the papers of Rissanen (see e.g. [3,4], or [5], various formulae analogous to (2) were investigated. These papers deal with the problem of choice of a probabilistic model and thus the measures of loss $\text{Loss}_{\mathcal{R}}(z)$ different from the square loss $\text{Loss}_{\mathcal{R}}^{\text{sq}}(z)$ are considered there.

The paper [9] provides both a motivation and a refinement to (2). In the paper [10], a value $\mathcal{K}^{\text{sq}}(z)$ called the *predictive (square-loss) complexity* of z is introduced and [9] shows that the inequality

$$\mathcal{K}^{\text{sq}}(z) \leq \text{Loss}_{\mathcal{R}}^{\text{sq}}(z) + \frac{(b-a)^2 \ln 2}{2} KP(\mathcal{R}) + C \quad (3)$$

holds for any computable decision rule $\mathcal{R}$, where KP stands for the prefix complexity (for definitions see [2]) and the constant C does not depend upon z and $\mathcal{R}$. CAP suggests minimising the right-hand side of (3), i.e. $\widetilde{\mathcal{R}}$ is called a CAP estimate if the minimum

$$\min_{\mathcal{R} \in \mathfrak{R}} \left(\text{Loss}_{\mathcal{R}}^{\text{sq}}(z) + \frac{(b-a)^2 \ln 2}{2} KP(\mathcal{R}) \right) \quad (4)$$

is attained at $\widetilde{\mathcal{R}}$.

2.2 Non-linear Regression

To construct the set of decision rules for the non-linear regression, we start with a sequence of functions $f_1, f_2, \dots$ which map X into $\mathbb{R}$. The set $\mathfrak{R}$ consists of all finite linear combinations $\theta_1 f_1 + \theta_2 f_2 + \dots + \theta_k f_k$, where $\theta = (\theta_1, \theta_2, \dots, \theta_k)^T$ is a finite-dimensional real-valued column vector. If we fix a dimension $k \in \mathbb{N}$, we will obtain the set $\mathfrak{R}_k$. Clearly, $\mathfrak{R} = \bigcup_{k=1}^{\infty} \mathfrak{R}_k$. A decision rule $\mathcal{R}$ may be identified with the corresponding θ and therefore we may identify $\mathfrak{R}$ with $\mathbb{R}^*$ and $\mathfrak{R}_k$ with $\mathbb{R}^k$. Let us introduce k -dimensional string vectors $F_i^{(k)} = (f_1(x_i), f_2(x_i), \dots, f_k(x_i))^T$, where $1 \leq i \leq l$ and $1 \leq k < +\infty$, and $(l \times k)$ -matrixes $F^{(k)}$ such that the element $F_{i,j}^{(k)}$ equals the j -th coordinate of $F_i^{(k)}$, where $1 \leq i \leq k$, $1 \leq j \leq l$, and $1 \leq k < +\infty$. Let us also introduce a column vector $Y = (y_1, y_2, \dots, y_l)^T$.

In the k -dimensional case, the LS formula (1) reads as

$$\min_{\theta \in \mathbb{R}^k} \sum_{i=1}^l (F_i^{(k)} \theta - y_i)^2. \quad (5)$$

If $l \geq k$, then the k -dimensional LS estimate $\hat{\theta}^{(k)}$ is given by the equation

$$\hat{\theta}^{(k)} = \left(\left(F^{(k)} \right)^T F^{(k)} \right)^{-1} \left(F^{(k)} \right)^T Y \quad (6)$$

(see e.g. [1]).

If we minimise (5) over k as well, we will probably either come to no solution or come to a solution corresponding to the exact fit. Another disadvantage of (5) is that it does not penalise the growth of coordinates of θ .

In the k -dimensional case, CAP formula (4) reads as

$$\min_{\theta \in \mathbb{R}^k} \left(\sum_{i=1}^l (F_i^{(k)} \theta - y_i)^2 + \frac{(b-a)^2 \ln 2}{2} KP(\theta | k) \right) \quad (7)$$

and in the case of the unbounded (*finite*) dimension we get

$$\min_{\theta \in \mathbb{R}^*} \left(\sum_{i=1}^l (F_i^{(d(\theta))} \theta - y_i)^2 + \frac{(b-a)^2 \ln 2}{2} KP(\theta) \right), \quad (8)$$

where $d(\theta)$ denotes the dimension of θ . The problem is to find a natural approximation of $KP(\theta)$ and we are discussing this problem in the next section.

3 Complexity of Real-Valued Parameters

In this section, we apply the estimate of the complexity of θ proposed in [3]. As we mentioned above, [3] deals with probabilistic models rather than with the problem of regression but the expression considered in [3] may be regarded

as a special case of general formula (2). The main result of [3] is the following approximation of the complexity of $\theta \in \mathbb{R}^k$:

$$\mathcal{K}(\theta \mid k) \approx \log^* [C(k) \|\theta\|_{\text{Loss}}^k] \quad . \quad (9)$$

Here $\log^* \alpha$ stands for $\log_2 \alpha + \log_2 \log_2 \alpha + \dots$, where only positive terms are included, $C(k)$ stands for the volume of the k -dimensional unit ball, and $\|\theta\|_{\text{Loss}}$ denotes the norm of θ induced by the second derivative of $\text{Loss}_\theta(z)$ taken at the point corresponding to the ‘maximum likelihood’ estimate, i.e.

$$\|\theta\|_{\text{Loss}} = \sqrt{\theta^T (D_\theta^2 \text{Loss}_\theta(z)|_{\theta=\hat{\theta}}) \theta} \quad , \quad (10)$$

where the minimum $\min_{\varphi \in \mathbb{R}^k} \text{Loss}_\varphi(z)$ is achieved at $\varphi = \hat{\theta}$.

As one can see, the formula is not independent of the minimisation problem (2) we are going to solve. The derivation of (10) may be outlined as follows. The estimate given by (2) is supposed to be close to the ‘maximum likelihood’ estimate $\hat{\theta}$ so $\text{Loss}_\theta(z)$ may be replaced by its second order approximation in the neighbourhood of $\hat{\theta}$. Then $\mathbb{R}^k$ is split into small rectangles such that inside each rectangle the approximation of $\text{Loss}_\theta(z)$ takes values which are sufficiently close to each other and then the rectangles are enumerated according to the ‘spiral fashion’.

We will now apply (10) to our problems (7) and (8). One may easily see that

$$D_\theta^2 \sum_{i=1}^l (F_i^{(k)} \theta - y_i)^2 = \sum_{i=1}^l \left(F_i^{(k)} \right)^T F_i^{(k)} \quad , \quad (11)$$

i.e. we obtain the sum of outer (Kronecker) products. Hence

$$\|\theta\|_{\text{Loss}}^2 = \sum_{i=1}^l \left[\theta^T \left(F_i^{(k)} \right)^T F_i^{(k)} \theta \right] = \sum_{i=1}^l \left(F_i^{(k)} \theta \right)^2 \quad . \quad (12)$$

Caring out the substitution, we obtain the following k -dimensional minimisation problem:

$$\min_{\theta \in \mathbb{R}^k} \left(\sum_{i=1}^l (F_i^{(k)} \theta - y_i)^2 + \frac{(b-a)^2 \ln 2}{2} \log^* C(k) \left[\sum_{i=1}^l \left(F_i^{(k)} \theta \right)^2 \right]^{k/2} \right) \quad . \quad (13)$$

If we approximate $KP(\theta)$ by $KP(\theta \mid k) + KP(k)$ and $KP(k)$ by $\log^*(k)$ (see [3] and [2]), where $k = d(\theta)$ is the dimension of θ , we will obtain the general formula

$$\begin{aligned} \min_{\theta \in \mathbb{R}^*} & \left(\sum_{i=1}^l (F_i^{(d(\theta))} \theta - y_i)^2 + \right. \\ & \left. \frac{(b-a)^2 \ln 2}{2} \log^* C(k) \left[\sum_{i=1}^l \left(F_i^{(k)} \theta \right)^2 \right]^{k/2} + \frac{(b-a)^2 \ln 2}{2} \log^* d(\theta) \right) \quad . \quad (14) \end{aligned}$$

The last term $\frac{(b-a)^2 \ln 2}{2} \log^* d(\theta)$ in (14) guarantees the existence of a minimum as long as long as minimums in (13) exist.

4 Minimisation

One can easily see that the parameter θ appears in (13) only in the products $F_i^{(k)}\theta$, where $1 \leq i \leq l$, so one may introduce the new vector parameter $a = F^{(k)}\theta$ ranging over the subspace $\text{Im } F^{(k)} \subseteq \mathbb{R}^l$. Therefore (13) has the form

$$\min_{a \in \text{Im } F^{(k)}} (\|a - Y\|^2 + f(\|a\|)) \quad , \quad (15)$$

where $\|a\|$ stands for the standard Euclidean norm $\|a\| = \sqrt{a_1^2 + a_2^2 + \dots + a_l^2}$ and f is a real-valued function of a real-valued parameter. It follows easily that the minimum is attained at $\tilde{a}$ collinear to the projection $\hat{Y}$ of Y on $\text{Im } F^{(k)}$. Namely if x_0 is the solution of

$$\min_{x \in \mathbb{R}} \left((x - \|\hat{Y}\|)^2 + f(x) \right) \quad , \quad (16)$$

then the minimum in (15) is achieved at $\tilde{a} = \frac{x_0}{\|\hat{Y}\|} \hat{Y}$. It follows from the definition of the LS estimate $\hat{\theta}^{(k)}$, that $\hat{Y} = F^{(k)}\hat{\theta}^{(k)}$ hence, by linearity, the minimum in (13) is achieved at

$$\tilde{\theta}^{(k)} = \frac{x_0}{\|F^{(k)}\hat{\theta}^{(k)}\|} \hat{\theta}^{(k)} \quad . \quad (17)$$

In statistics, there exists a qualitative analogy to this formula. Stein's paradox suggest normalising the Maximum Likelihood estimate in the case of the normal distribution and the square loss. See [8,7] for details.

5 Dual Variables

Suppose that the number of parameters k exceeds l the number of given examples. In this case, there is no unique LS estimate is not unique. We have many vectors θ which correspond to the exact fit. Formula (13) does not include θ unless it is multiplied by $F_i^{(k)}$ and therefore (13) does not allow us to distinguish between different sets of parameters that still give equal predictions on the training set. Hence, $\tilde{\theta}^{(k)}$ from (17) provides a solution for (13) if $\tilde{\theta}^{(k)}$ suffers zero loss on z .

It is natural to choose $\theta \in \mathbb{R}^k$ with the smallest Euclidean norm $\|\theta\|$. Such θ is given by the method of dual variables (see, e.g. [6]). According to this method, the value $\hat{\mathcal{R}}(x) = \sum_{i=1}^k \theta_i^{(k)} f_i(x)$ of the decision rule $\hat{\mathcal{R}}$ corresponding to the LS estimate with the smallest value of $\|\theta\|$ on a signal x is given by the formula

$$\hat{\mathcal{R}}(x) = Y^T \left(\mathbf{K}^{(k)} \right)^{-1} \mathbf{k}^{(k)}(x) \quad , \quad (18)$$

where $\mathbf{K}^{(k)}$ is an $(l \times l)$ -matrix such that $\mathbf{K}_{i,j}^{(k)} = K^{(k)}(x_i, x_j)$ for $1 \leq i, j \leq l$, $\mathbf{k}^{(k)}(x)$ is an l -dimensional vector such that $\mathbf{k}_i^{(k)}(x) = K^{(k)}(x_i, x)$ for $1 \leq i \leq l$,

and $K^{(k)} : X \times X \rightarrow \mathbb{R}$ is the *kernel* associated with the non-linear regression problem under consideration, i.e.

$$K^{(k)}(x', x'') = \sum_{i=1}^k f_i(x') f_i(x'') . \quad (19)$$

Note that the size of $\mathbf{K}^{(k)}$ does not increase with the increase of k .

6 Experiments and Discussion

6.1 Toy Examples

We consider the following one-dimensional toy problem. Consider the function $y = \sin(x)$ on the interval $[-A, A]$ and the Gaussian noise $\xi \sim \mathcal{N}(0, \sigma^2)$. Our approach requires tight bounds so we must bound the range of the noise. If $\sin(x) + \xi$ falls outside the interval $[-1, 1]$, we replace it by the nearest number, either 1 or -1 . Both training and test examples are taken according to the uniform distribution. We try to approximate the data by k -dimensional polynomials.

We calculate the LS estimate by (1), normalise it according to (17), and compare the difference. The main empirical result here may be formulated in the following way. Formula (17) overperforms the simple LS estimate on very ‘complicated’ and ‘noisy’ problems, i.e. cases with large values of A and σ^2 and small numbers of training examples. Otherwise our correction can only spoil the LS estimate.

Fig. 6.1 shows the squared loss on the training set of size 100 averaged over 1000 independent trials. The results correspond to the case $A = 6$, $\sigma^2 = 0.5$, the size of training sets equals 25. You may see that this case is very difficult and the best LS estimates of degree 3 perform only slightly better than those of degree 0, i.e. constant predictions.

Unfortunately, experiments with (14) failed. The graph of complexity with respect to the degree exhibits an increasing pattern and does not allow to locate the optimal degree.

6.2 Boston Housing

The Boston Housing database (available at <ftp://ftp.ics.uci.com/pub/machine-learning-databases/housing>) is often used to test different non-linear regression techniques (see e.g. [6]). The entries of this database are strings of 14 parameters which describe houses in different neighbourhoods of Boston. The last elements of these strings are prices of houses in thousands of dollars, ranging from 5 to 50. We use prices as outcomes in our experiments.

We use the polynomial kernel

$$K^{(k)}(x', x'') = \left(x'^T x'' + 1 \right)^d \quad (20)$$

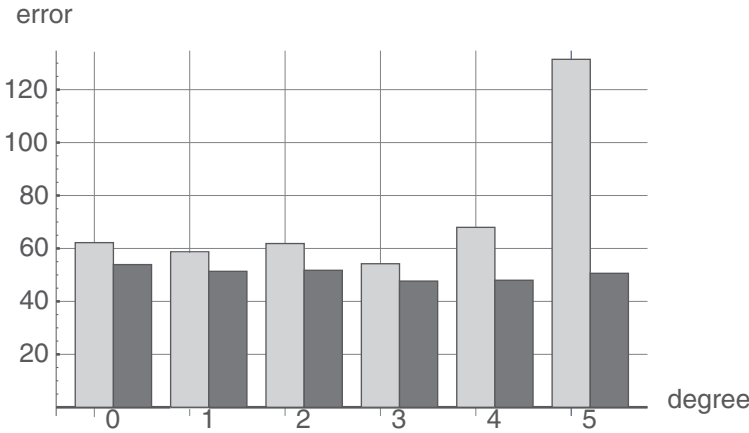


Fig. 1. The results on a toy example. The error of LS estimates is represented by light gray bars and the error of our method is represented by black ones

which correspond to the approximating of data by sums of normalised monomials of degree smaller then or equal to d . Following [6], we concentrate on $d = 5$. Our methodology is also similar to [6], we pick test sets of size 480 and training sets of size 25 randomly and repeat the procedure for 100 times.

The dimension k of the set of monomials equals $\binom{13+d}{13}$ but this natural assignment turns out to be meaningless. The correction coefficient we get is very close to 1 and it improves the performance of the algorithm by around a ten thousandth of a percent.

We may also consider using a kernel K as approximating the data by a linear combination of $\mathbf{k}_i^{(k)}(x) = K^{(k)}(x_i, x)$ (see Sect. 5). In this case, the dimension equal the size of the training set, which is much smaller.

If we make this assumption about the degree, the results become much more reasonable. Our method improves the performance by 14.7% (we obtain the average square loss over 100 trials equal to 69.1 against 81.0).

We must admit that our method is still no match to the ridge regression. The idea of the ridge regression (see, e.g. [6]) is to introduce an extra term to (18), i.e. to consider

$$\widehat{\mathcal{R}}(x) = Y^T \left(\mathbf{K}^{(k)} + aI \right)^{-1} \mathbf{k}^{(k)}(x) , \quad (21)$$

where $a > 0$ and I is the unit matrix. The paper [6] show that under the same settings the ridge regression is able to decrease the mistake down to 10.4.

Despite its theoretical justification, our method turns out to be much more rough than the ridge regression. In fact, ridge regression performs the same

task of penalising the growth of coefficients. The solution, given by the ridge regression minimises the expression $a\|\theta\| + \text{Loss}_\theta^{\text{sq}}$. This approach, motivated by empirical considerations, proves to be very sound.

7 Acknowledgements

I would like to thank Prof. V. Vovk and Prof. A. Gammerman for providing guidance to this work.

References

1. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990. 205
2. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, New York, 2nd edition, 1997. 204, 206
3. J Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983. 203, 204, 205, 206
4. J Rissanen. Stochastic complexity. *Journal of Royal Statistical Society*, 49(3):223–239, 1987. 204
5. J Rissanen. Stochastic complexity in learning. *Journal of Computer and System Sciences*, 55:89–95, 1997. 204
6. C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, 1998. 207, 208, 209
7. Mark J. Schervish. *Theory of Statistics*. Springer–Verlag, 1995. 207
8. Gábor J. Székely. *Paradoxes in Probability Theory and Mathematical Statistics*. Akadémiai Kiadó Budapest, 1986. 207
9. V. Vovk and A. Gammerman. Complexity approximation principle. *The Computer Journal*, 42(4):318–322, 1999. 203, 204
10. V. Vovk and C. J. H. C. Watkins. Universal portfolio selection. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 12–23, 1998. 204

Handling Continuous-Valued Attributes in Decision Tree with Neural Network Modeling

DaeEun Kim¹ and Jaeho Lee²

¹ Division of Informatics, University of Edinburgh
Edinburgh, EH1 2QL, United Kingdom
`daeun@dai.ed.ac.uk`

² Department of Electrical Engineering, University of Seoul
Tongdaemoon-ku, Seoul, 151-011, Korea
`jaeho@uofs.ac.kr`

Abstract. Induction tree is useful to obtain a proper set of rules for a large amount of examples. However, it has difficulty in obtaining the relation between continuous-valued data points. Many data sets show significant correlations between input variables, and a large amount of useful information is hidden in the data as nonlinearities. It has been shown that neural network is better than direct application of induction tree in modeling nonlinear characteristics of sample data. It is proposed in this paper that we derive a compact set of rules to support data with input variable relations. Those relations as a set of linear classifiers can be obtained from neural network modeling based on back-propagation. This will also solve overgeneralization and overspecialization problems often seen in induction tree. We have tested this scheme over several data sets to compare with decision tree results.

1 Introduction

Discovery of decision rules and recognition of patterns from data examples is one of challenging problems in machine learning. If data points contain numerical attributes, induction tree method needs a discretization of continuous-valued attributes with threshold values. Induction tree algorithms such as C4.5 build decision trees by recursively partitioning the input attribute space [14]. The tree traversal from the root node to each leaf leads to one conjunctive rule. Each internal node in decision tree has a splitting criterion or threshold for continuous-valued attributes to partition some part of the input space, and each leaf represents a class related to the conditions of each internal node.

Approaches based on decision tree involve the discretization of continuous-valued attributes in input space, making many rectangular divisions. As a result, it may have the inability to detect data trend or desirable classification surface. Even in the case of multivariate discretization methods which search at the same time for threshold values for more than one continuous attribute [4,12], the decision rules may not reflect data trend or the decision tree may build many

rules with support of a small number of examples, often called over-specialization problem.

A possible way is suggested to catch the trend of data. It first tries to fit a given data set for the relationship between data points using a statistical technique, generates many data points on the response surface of the fitted curve, and then induces rules with induction tree. This method was introduced as an alternative measure against the problem of direct application of induction tree to raw data [9,10]. However, it still has a problem to need many induction rules to reflect the response surface.

In this paper we suggest to investigate a hybrid technique to combine neural networks and knowledge based systems for data classification. It has been shown that neural network is better than direct application of induction tree in modeling nonlinear characteristics of sample data [3,13,15]. Neural networks have the advantage that they can deal with noisy, inconsistent and incomplete data. A method to extract symbolic rules from a neural network has been proposed to increase the performance of decision process [15]. They used in sequence a weight-decay back-propagation over a three-layer feedforward network, a pruning process to remove irrelevant connection weights, a clustering of hidden unit activations, and extraction of rules from discretized unit activations. Symbolic rules they derived from neural networks did not include input attribute relations. Also the direct conversion from neural networks to rules is related to exponential complexity when using search-based algorithm over incoming weights for each unit [5,16].

Our approach is to train a neural network with sigmoid functions and to use decision classifiers based on weight parameters of neural networks. Then induction tree selects the most relevant input variables and furthermore the desirable input variable relations for data classification. This algorithm is tested on various types of data and compared with the method based on decision tree alone.

2 Problem Statement

Induction tree is useful for a large number of examples, and it enables us to obtain proper rules from examples rapidly [14]. However, it has the difficulty in inferring relations between data points and cannot handle noisy data.

We can see a simple example of undesirable rule extraction discovered in induction tree application. Fig.1(a) displays a set of 29 original sample data with two classes. It appears that the set has four sections which have the boundaries of direction from upper-left to lower-right. A set of the dotted boundary lines is the result of multivariate classification by induction tree. It has six rules to classify data points. Even in C4.5 run, it has four rules with 6.9 % error, making divisions with attribute y . The rules do not catch data clustering completely in this example. Fig.1(b)-(c) show neural network fitting with back-propagation method. In Fig.1(b)-(c) neural networks have slopes $\alpha = 1.5, 4.0$ for sigmoids, respectively. After curve fitting, 900 points were generated uniformly on the

response surface for the mapping from input space to class, and the response values of neural network were calculated as shown in Fig.1(d). The result of C4.5 application to those 900 points followed the classification curves, but produced 55 rules. The production of many rules results from the fact that decision tree makes piecewise rectangular division for each rule, even though the response surface for data clustering has correlation between input variables.

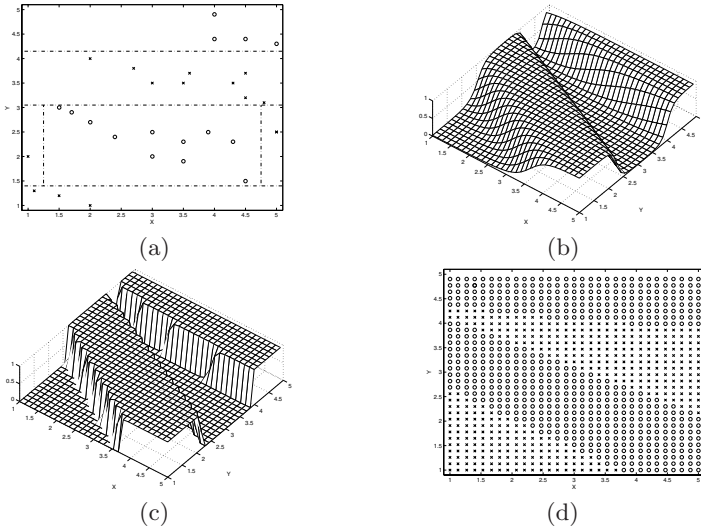


Fig. 1. Example (a) data set and decision boundary (O : class 1, X : class 0) (b)-(c) neural network fitting (d) data set with 900 points

As shown above, the decision tree has over-generalization problem for a small number of data and over-specialization problem for a large number of data. A possible suggestion is to consider or derive relations between input variables as another attribute for rule extraction. However, it is difficult to find input variable relations for classification directly in supervised learning, while unsupervised methods can use statistical methods such as principal component analysis [6].

3 Method

The goal for our approach is to generate rules following the shape and characteristics of response surface. Usually induction tree cannot trace the trend of data, and it determines data clustering only in terms of input variables, unless we apply other relation factors or attributes. In order to improve classification rules from a large training data set, we allow input variable relations for multi-attributes in a set of rules. We develop in this paper a two-phase method for rule extraction over continuous-valued attributes.

Given a large training set of data points, the first phase, as a feature extraction phase, is to train feed-forward neural networks with back-propagation and collect the weight set over input variables in the first hidden layer. A feature useful in inferring multi-attribute relations of data is found in the first hidden layer of neural networks. The extracted rules involving network weight values will reflect features of data examples and provide good classification boundaries. Also they may be more compact and comprehensible, compared to induction tree rules.

In the second phase, as a feature combination phase, each extracted feature for linear classification boundary is combined together using Boolean logic gates. In this paper, we use an induction tree to combine each linear classifier. From our results, it is shown that the two-phase method proposed is in general very effective and leads to solutions of high quality.

The highly nonlinear property of neural networks makes it difficult to describe how they reach predictions. Although their predictive accuracy is satisfactory for many applications, they have long been considered as a complex model in terms of analysis. By using expert rules derived from neural networks, the neural network representation can be more understandable. We use a neural network modeling with two hidden layers to obtain linear classification boundary. After training data patterns with neural network by back-propagation, we can have linear classifiers in the first hidden layer. To get desirable classifiers, we need to set sigmoid functions with high slope. It has been shown that a particular set of functions can be obtained with arbitrary accuracy by at most two hidden layers given enough nodes per layer [2]. Also one hidden layer is sufficient to represent any Boolean function [7]. Our neural network structure has two hidden layers, where the first hidden layer makes a local feature selection with linear classifiers and the second layer receives Boolean logic values from the first layer and maps any Boolean function. The second hidden layer and output layer can be thought of as a sum of product of Boolean logic gates. The n -th output of neural network for a set of data is $F_n = f(\sum_k^{N_2} W_{kn}^2 f(\sum_j^{N_1} W_{jk}^1 f(\sum_i^{N_0} W_{ij}^0 a_i)))$

For a node in the first hidden layer, the activation is defined as $f(\sum_i^{N_0} a_i W_{ik})$ for the k -th node where N_0 is the number of input attributes, a_i is an input, and $f(x) = 1.0/(1.0 + e^{-\alpha x})$ as a sigmoid function. When we train neural networks with back-propagation method, α , the slope of sigmoid function is increased as iteration continues. If we have a high value of α , the activation of each neuron is close to the property of digital logic gates, which has a binary value 0 or 1.

Except the first hidden layer, we can replace each neuron by logic gates if we assume we have a high slope on sigmoid function. Input to each neuron in the first hidden layer is represented as a linear combination of input attributes and weights, $\sum_i^N a_i W_{ik}$. This forms linear classifiers for data classification as a feature extraction over data distribution. When Fig.1(a) data is trained, we can introduce new attributes $aX + bY$ where a, b is a constant. We used two hidden layers with 4 nodes and 3 nodes, respectively, where every neuron node has a high slope of sigmoid to guarantee desirable linear classifiers as shown in Fig.1(c). Before applying data in Fig.1(d) to induction tree, we added four new

attributes made from linear classifiers in the first hidden layer over 900 points and then we could obtain only four rules with C4.5, while a simple application of C4.5 for those data generated 55 rules. The rules are given as follows :

- rule 1 : if $(1.44x + 1.73y \leq 5.98)$, then class 0
 rule 2 : if $(1.44x + 1.73y > 5.98)$
 and $(1.18x + 2.81y \leq 12.37)$ then class 1
 rule 3 : if $(1.44x + 1.73y > 5.98)$
 and $(1.18x + 2.81y > 12.37)$
 and $(0.53x + 2.94y < 14.11)$, then class 0
 rule 4 : if $(1.44x + 1.73y > 5.98)$
 and $(1.18x + 2.81y > 12.37)$
 and $(0.53x + 2.94y > 14.11)$, then class 1

These linear classifiers exactly match with boundaries shown in Fig.1(c), and they are more dominant for classification in terms of entropy maximization than a set of input attributes itself. Even if we include input attributes, the entropy measurement leads to a rule set with boundary equations. These rules are more meaningful than those of direct C4.5 application to raw data since their division shows the trend of data clustering and how each attribute is correlated.

Our approach can be applied to the data set which has both discrete and continuous values. If there is a set of input attributes, $Y = \{D_1, \dots, D_m, C_1, \dots, C_n\}$ where D_i is a discrete attribute and C_j is a continuous-valued attribute. For any discrete attribute D_x , it has a finite set of values available. For example, if there is a value set $\{d_{x1}, d_{x2}, d_{x3}, \dots, d_{xp}\}$ for D_x , we can have a Boolean value for each value, using the conditional equation $D_x = d_{xj}$, for $j = 1, \dots, p$. We can put this state as a node in the first hidden layer, and then one of linear classifiers obtained with neural network is $L_k = \sum_i^{m+n} A_i W_{ik} = \sum_i^n C_i W_{ik} + \sum_i^m D_i W_{ik}$ where A_i is a member of the set Y . Since we have no interest in the relation of discrete attributes whose numeric conditions and coefficient values are not meaningful in this model, the value of linear classifier L_k only depends on a linear combination of continuous attributes and weights.

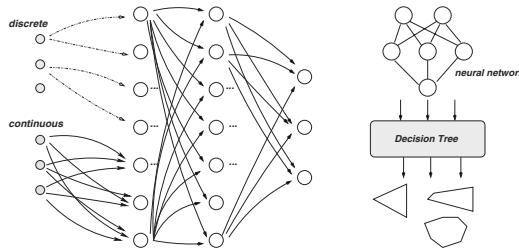


Fig. 2. Diagram for neural network and decision tree

The choice of discrete attributes in rules can be handled using induction tree algorithm more properly, without interfering the relations of continuous-valued attributes. The induction tree can do splitting any continuous value with selection of thresholds for given attributes, while it cannot derive the relation of input attributes directly. In our method, we can add to the data set of induction tree, new attributes $L_k = \sum_i^n C_i W_{ik}$ for $k = 1, \dots, r$, where r is the number of nodes in the first hidden layer for continuous-valued attributes. The new set of attributes for induction tree is $Y' = \{D_1, D_2, \dots, D_m, C_1, C_2, \dots, C_n, L_1, L_2, \dots, L_r\}$. The entropy measurement will find out the most significant classification over the new set of attributes. Also we have tested another attribute set $Y'' = \{L_1, L_2, \dots, L_r\}$ which consists of only linear classifiers generated by neural network.

4 Experiments

Our method has been tested on several sets of data in UCI depository [1]. Table 1 shows classification error rates for neural network and C4.5 [14] algorithm, and Table 2 shows error rates in our two methods. The first linear classifier $\{C + L\}$ method has a set of attributes for C4.5 classification, including both original input attributes and neural network linear classifiers together, while the second $\{L\}$ method only includes neural network linear classifiers.

Table 1. Data classification error rate result in neural network and C4.5

data	neural network				C4.5	
	pat / attr	training (%)	testing (%)	nodes	training (%)	testing (%)
wine	178 / 13	0 ± 0	1.6 ± 0.6	8 - 5	1.2 ± 0.1	6.6 ± 1.2
iris	150 / 4	0.6 ± 0.1	4.1 ± 1.3	5 - 4	1.9 ± 0.1	5.5 ± 1.1
breast-w	683 / 9	0.3 ± 0.1	4.9 ± 0.7	8 - 5	1.1 ± 0.1	4.7 ± 0.5
ion	351 / 34	0.8 ± 0.2	8.6 ± 0.8	10 - 7	1.6 ± 0.2	10.8 ± 1.3
pima	768 / 8	4.3 ± 0.5	24.7 ± 6.9	15 - 9	15.1 ± 7.6	24.7 ± 3.4
glass	214 / 9	6.5 ± 0.7	30.8 ± 2.0	15 - 8	7.0 ± 0.2	32.0 ± 2.5
bupa	345 / 6	5.9 ± 0.6	32.6 ± 1.5	10 - 7	12.9 ± 0.8	34.0 ± 1.4

Table 2. Data classification error result in our method using linear classifiers

data	linear classifier $\{C+L\}$		linear classifier $\{L\}$	
	training (%)	testing (%)	training (%)	testing (%)
wine	0.1 ± 0.1	3.7 ± 1.3	0.1 ± 0.1	3.2 ± 1.3
iris	0.7 ± 0.1	5.7 ± 1.1	0.7 ± 0.1	5.2 ± 1.7
breast-w	0.7 ± 0.1	4.4 ± 0.4	0.9 ± 0.2	4.4 ± 0.3
ion	0.8 ± 0.3	9.0 ± 0.9	1.2 ± 0.3	8.8 ± 0.9
pima	11.7 ± 1.3	27.0 ± 4.5	13.5 ± 1.2	26.6 ± 3.9
glass	5.8 ± 0.3	35.1 ± 1.7	6.7 ± 0.4	35.2 ± 2.2
bupa	10.5 ± 1.0	32.5 ± 2.6	15.3 ± 0.9	32.4 ± 2.9

The error rates were estimated by running the complete 10-fold cross-validation ten times, and the average and the standard deviation for ten runs are given in the table. Our method, adding linear classifiers into new attributes, is better than C4.5 in some sets and worse in data sets such as *glass*, *bupa* and *pima* which are hard to predict even in neural network. The result supports the fact that the method greatly depend on neural network training. If neural network fitting is not correct, then the result may mislead the result. Normally C4.5 application shows the error rate is very high for training data in Table 1. Table 3 says the number of rules using our method is smaller than that using conventional C4.5 in most of data sets. Especially when only linear classifiers from neural network are used, it is quite effective to reduce the number of rules. Most of data sets in UCI depository have a small number of data examples relative to the number of attributes. The significant difference between a simple C4.5 application and a combination of C4.5 application and neural network is not seen distinctively in UCI data unlike synthetic data in Fig.1. Information of data trend or input relations can be more definitely described when given many data examples relative to the number of attributes.

Table 3. Number of attributes and rules for C4.5 applications

data	C4.5		linear classifier ^{C+L}		linear classifier ^{L}	
	rules	attributes	rules	attributes	rules	attributes
wine	5.4 ± 0.2	13	3.0 ± 0.0	13 + 8	3.0 ± 0.0	8
iris	4.8 ± 0.2	4	3.9 ± 0.2	4+5	3.7 ± 0.2	5
breast-w	18.6 ± 0.7	9	10.5 ± 0.9	9 + 8	7.8 ± 0.8	8
ion	14.2 ± 0.6	34	7.7 ± 0.8	34 + 10	6.3 ± 0.5	10
pima	26.7 ± 2.8	8	33.7 ± 4.2	8 + 15	23.6 ± 2.9	15
glass	25.1 ± 0.6	10	24.6 ± 0.7	10 + 15	23.0 ± 0.8	15
bupa	29.1 ± 1.2	6	24.1 ± 2.0	6 + 10	15.3 ± 1.5	10

Table 1 and 2 says neural network classification is better than C4.5 applications. If we can derive easily Boolean gates directly from neural network, the combination of linear classifiers and Boolean logic gates will form a set of good rules. Each threshold logic based on neural weights is equivalent to a set of logic gates when it is applied to digital logic and will form a sum of product of Boolean logic [11]. We need to find an efficient or heuristic way to generate a set of Boolean logic gates from neural network function. If we apply a simple ID3 algorithm with linear classifiers to reduce error rate in the training sets instead of C4.5 algorithm, it may increase the performance up to the level of neural network.

5 Conclusions

This paper presents a hybrid method for constructing a decision tree from neural networks. Our method uses neural network modeling to find unseen data points

and then induction tree is applied to data points for symbolic rules, using features from neural network. The combination of neural network and induction tree will compensate for the disadvantages of one approach alone. This method has several advantages over a simple decision tree method. First, we can obtain good features for classification boundary from neural networks by training input patterns. Second, because of feature extractions about input variable relations, we can obtain a compact set of rules to reflect input patterns.

We still need further work such as applying minimum description length principle to reduce the number of attributes over linear classifiers or comparing with other methods such as regression tree methods.

References

1. C. Blake, E. Keogh, and C.J. Merz. *UCI repository of machine learning databases*. In *Proceedings of the Fifth International Conference on Machine Learning*, <http://www.ics.uci.edu/mlearn>, 1998. 216
2. G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Technical Report, Department of Computer Science, Tufts University, Medford, MA, 1988. 214
3. T.G. Dietterich, H. Hild, and G. Bakiri. A comparative study of id3 and backpropagation for english text-to-speech mapping. In *Proceedings of the 1990 Machine Learning Conference*, pages 24–31. Austin, TX, 1990. 212
4. U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of IJCAI'93*, pages 1022–1027. Morgan Kaufmann, 1993. 211
5. L. Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, New York, 1994. 212
6. S.S. Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall, Upper Saddle River, N.J., 2nd edition, 1999. 213
7. J. Hertz, R.G. Palmer, and A.S. Krogh. *Introduction to the Theory of Neural Computation*. Addison Wesley, Redwood City, Calif., 1991. 214
8. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
9. K.B. Irani and Qian Z. Karsm : A combined response surface / knowledge acquisition approach for deriving rules for expert system. In *TECHCON'90 Conference*, pages 209–212, 1990. 212
10. DaeEun Kim. Knowledge acquisition based on neural network modeling. Technical report, Directed Study, The University of Michigan, Ann Arbor, 1991. 212
11. Zvi Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, London, 1970. 217
12. W. Kweldo and M. Kretowski. An evolutionary algorithm using multivariate discretization for decision rule induction. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pages 392–397. Springer, 1999. 211
13. J.R. Quinlan. Comparing connectionist and symbolic learning methods. In *Computational Learning Theory and Natural Learning Systems*, pages 445–456. MIT Press, 1994. 212
14. J.R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Approach*, (4):77–90, 1996. 211, 212, 216

15. R. Setiono and Huan Lie. Symbolic representation of neural networks. *Computer*, 29(3):71–77, 1996. 212
16. G.G. Towell and J.W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, Oct. 1993. 212

Learning Context-Free Grammars with a Simplicity Bias

Pat Langley and Sean Stromsten

DaimlerChrysler Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304 USA
langley@isle.org
sean@psych.stanford.edu

Abstract. We examine the role of simplicity in directing the induction of context-free grammars from sample sentences. We present a rational reconstruction of Wolff’s SNPR – the GRIDS system – which incorporates a bias toward grammars that minimize description length. The algorithm alternates between merging existing nonterminal symbols and creating new symbols, using a beam search to move from complex to simpler grammars. Experiments suggest that this approach can induce accurate grammars and that it scales reasonably to more difficult domains.

1 Introduction

In this paper we focus on the task of inducing context-free grammars from training sentences. Much recent work on this topic has dealt with learning finite-state structures, but there is considerable evidence that human language involves more powerful grammatical representations. In context-free grammar induction, the learner must find not only a set of grammatical rewrite rules but also the non-terminal symbols used in those rules. For example, in addition to deciding that an English sentence can be composed of a noun phrase and a verb phrase, it must also create definitions for these intermediate concepts.

A central challenge of grammar induction involves the generative nature of language. The learner must somehow create a knowledge structure that produces an infinite number of sentences from a finite set of training cases. Typically, this requires recursive or iterative structures, which can cause overgeneralizations. Effective induction of context-free grammars requires strong constraints on search through the space of candidates. One that often recurs in the literature is a bias toward *simple* grammars.

This bias helps avoid one sort of trivial grammar that has a separate rule for each training sentence and that does not generalize at all to new sentences. However, a naive notion of simplicity leads to another sort of trivial grammar that admits any string of words and overgeneralizes drastically. A more useful variation on this idea views the grammar as a code and seeks to compress the sample sentences, minimizing the summed description length of the grammar and its derivations of training sentences. By ‘simplicity’ then, we mean that of the grammar *and* the derivations of the training sentences under the grammar.

In the following pages, we examine the extent to which this notion of simplicity can successfully direct the grammar-induction process. We explore this idea in the context of GRIDS, a rational reconstruction of Wolff's (1982) SNPR system. We first describe GRIDS' representation, performance component, learning algorithm, and evaluation function, then present experimental studies designed to evaluate the system's learning behavior. In closing, we discuss related work on grammar induction and outline directions for future research.

2 Grammar Induction Driven by Simplicity

As noted above, GRIDS represents grammatical knowledge as context-free rewrite rules, using a top-level symbol (S), a set of nonterminals, and a set of terminal symbols corresponding to words. Each rewrite rule includes one nonterminal symbol on the left-hand side and one or more symbols on the right, indicating that one can replace the former with the latter in recognizing or generating a sentence. Following VanLehn and Ball (1987), we restrict GRIDS' grammars so that no rule has an empty right-hand side, the only rules of the form $X \rightarrow Y$ are those in which Y is a terminal symbol, and every nonterminal appears in the derivation of some sentence. This restriction does not limit representational power, as one can transform any context-free grammar into this form.

The performance component of GRIDS is a top-down, depth-first parser that repeatedly substitutes the first nonterminal X in its string with the right-hand side of a rewrite rule having X on the left. We do not view this performance algorithm as part of our theoretical framework, and its implementation is far from efficient. However, it does let GRIDS determine whether a given grammar parses a given string of words, and thus whether that grammar is overly general, overly specific, or accurate for the language at hand.

2.1 Learning Operators and Search Organization

GRIDS' approach to grammar induction, as in Wolff's earlier system, relies on two learning operators. The first creates a nonterminal symbol X and an associated rewrite rule that decomposes X into its constituents. In grammars for natural languages, such symbols and their rules correspond to specific phrases and clauses. The introduction of phrasal terms should be useful when certain combinations of symbols tend to occur together in sentences. Table 1 (a) gives a simple example of this operator's effect.

The second operator involves merging two nonterminal symbols into a single symbol. The resulting sets of rules with the same left-hand side correspond, in grammars for natural languages, to word classes (e.g., nouns and verbs) and phrasal classes (e.g., noun phrases). Their introduction should be useful when certain symbols tend to occur in similar contexts within the language. We should note one important side effect of the merge operator. Given the rewrite rule $X \rightarrow Y \dots Z$, merging X and Z produces the rule $X \rightarrow Y \dots X$, which involves a recursive call. Table 1 (b) illustrates this outcome in a simple grammar, though merging can also produce indirect recursions.

Table 1. The learning operators used in GRIDS include (a) creating a new symbol and rewrite rule based on two existing symbols, and (b) merging two existing symbols, which can lead to redundant (and thus removed) rules, as well as to recursive grammars

(a) Creating symbol <i>AP1</i>	(b) Merging <i>AP1</i> and <i>AP2</i>
$NP \rightarrow ART\ ADJ\ NOUN$ $NP \rightarrow ART\ ADJ\ ADJ\ NOUN$	$NP \rightarrow ART\ AP1$ $NP \rightarrow ART\ AP2$ $AP1 \rightarrow ADJ\ NOUN$ $AP2 \rightarrow ADJ\ AP1$
$\Downarrow$	$\Downarrow$
$NP \rightarrow ART\ AP1$ $NP \rightarrow ART\ ADJ\ AP1$ $AP1 \rightarrow ADJ\ NOUN$	$NP \rightarrow ART\ AP1$ $AP1 \rightarrow ADJ\ NOUN$ $AP1 \rightarrow ADJ\ AP1$

GRIDS starts by transforming the sample sentences into an initial ‘flat’ grammar that contains only rules of the form $S \rightarrow X \dots Y$ (one for each observed sentence) and $X \rightarrow W$ (for each word W). Thus, each S rewrite rule and its associated word rules correspond to a single training instance, so that the initial grammar covers all (and only) the training sentences. Symbol creation does not change the coverage of a grammar, and symbol merging can never decrease the coverage. Thus, as GRIDS proceeds, it only considers grammars with the same or greater generality than the current hypothesis. The current version uses beam search, with a beam size of three, to control its steps through the resulting space.

The learning process in GRIDS alternates between two modes, each relying on a different operator. First the system considers all ways of merging pairs of nonterminal symbols in each current grammar, producing a set of successor grammars. When this action produces a new grammar that contains identical rewrite rules, all but one of the redundant rules are removed. Next the system uses an evaluation function, which we will discuss shortly, to select the best b grammars from the successors, breaking ties among candidates at random. If the evaluation metric indicates that at least one of the successors constitutes an improvement over the current best grammar, the new grammars become the current best set and the system continues in this mode.

However, if none of the new grammars scores better than the current best candidate, GRIDS switches from ‘merge’ mode into ‘create’ mode. Here the algorithm considers all ways of creating new terms, and their associated rules, from pairs of nonterminal symbols that occur in sequence within the grammars. GRIDS then substitutes the new term for all occurrences of the sequence in the prospective grammar. Again, it selects the best alternatives and, if some score better than the current best grammar, the best b candidates become the current set and the program continues in ‘create’ mode; if not, GRIDS changes modes

and again considers merging. The algorithm continues in this manner, alternating between modes until neither leads to improvement, in which case it halts.

2.2 Directing Search with Description Length

We have seen that GRIDS carries out a beam search through the space of context-free grammars, starting with a specific grammar based on training sentences and moving toward more general candidates. However, the space of grammars is large and the system needs some evaluation metric to direct search toward promising candidates. To this end, it applies the principle of minimum description length, measuring the simplicity of each candidate grammar G in terms of the description length for G plus that for the training sentences, encoded as derivations in G .

In this formulation, a hypothetical ‘receiver’ must know how to interpret the string of bits that encode the model and data. GRIDS encodes the rules of the grammar as strings of symbols separated by tokens of a STOP symbol. Each nonterminal token requires $\log(N+1)$ bits, where N is the number of nonterminal types, and the terminals each require $\log P_i$, where P_i is the number of words with the same part of speech. The derivations are strings of rewrite rules. The left-hand side of each is known, at each point, given the previous rules, so it need only distinguish among the R right-hand sides, which requires $\log R$ bits.

Intuitively, this measure should shun large grammars with overly specific rules, despite their short derivations, because other grammars will have smaller descriptions and do nearly as well on the derivations. The measure avoids very small, overly general grammars because they can describe too many unobserved strings, so that bits must be wasted in encoding the derivations of actual sentences just to distinguish them from these nonsentences. In general, a good code assigns long encodings to rare strings and short encodings to common ones. In our case, a good grammar may also forfeit entirely the ability to encode some (unobserved) strings in exchange for the ability to encode others (observed training sentences) more efficiently.

3 Experimental Studies of GRIDS’ Behavior

The central hypothesis in our work was that simplicity, as measured by description length, is a powerful bias for constraining the process of grammar induction. To evaluate this hypothesis, we carried out a number of experiments, which we report after considering their design and the domains used therein.

3.1 Grammatical Domains and Experimental Design

We decided to use artificial grammars in our experiments, since they let us both control characteristics of the domain and measure the correctness of the induced knowledge structures. In particular, we designed the two subsets of English grammar shown in Table 2. The first (a) includes declarative sentences with arbitrarily long strings of adjectives and both transitive and intransitive verbs, but no relative clauses, prepositional phrases, adverbs, or inflections. The second

Table 2. Two grammars used to generate training and test sentences for experiments with the GRIDS algorithm. The first grammar (a) includes arbitrary strings of adjectives, whereas the second (b) supports arbitrarily embedded relative clauses

(a)	(b)
$S \rightarrow NP VP$	$S \rightarrow NP VP$
$VP \rightarrow VERBI$	$VP \rightarrow V NP$
$VP \rightarrow VERBT NP$	$NP \rightarrow ART NOUN$
$NP \rightarrow the NOUN$	$NP \rightarrow ART NOUN RC$
$NP \rightarrow the AP NOUN$	$RC \rightarrow REL VP$
$AP \rightarrow ADJ$	$VERB \rightarrow saw$
$AP \rightarrow ADJ AP$	$VERB \rightarrow heard$
$VERBI \rightarrow ate$	$NOUN \rightarrow cat$
$VERBI \rightarrow slept$	$NOUN \rightarrow dog$
$VERBT \rightarrow saw$	$NOUN \rightarrow mouse$
$VERBT \rightarrow heard$	$ART \rightarrow a$
$NOUN \rightarrow cat$	$ART \rightarrow the$
$NOUN \rightarrow dog$	$REL \rightarrow that$
$ADJ \rightarrow big$	
$ADJ \rightarrow old$	

grammar (b) contains declarative sentences with arbitrarily embedded relative clauses, but has no adjectives, adverbs, prepositional phrases, or inflections.

These two grammars are unsophisticated compared to those required for natural languages, but they involve recursion and generate an infinite class of sentences, thus providing tests of GRIDS' ability to generalize correctly. However, one can also state both grammars as finite-state machines, which involve iteration but not recursion, so we also examined two languages that required center embedding. One involved sentences with a string of *a*'s followed by an equal number of *b*'s, whereas the other involved strings of balanced parentheses. Both languages have been used as testbeds in earlier efforts on grammar induction.

For the two English subsets, we created 20 training sets with enough strings in each for the program to reach asymptotic performance, with instances for the adjective phrase domain having a length of ten or less and those for the relative clause grammar length 15 or less. For the parenthesis-balancing and $(ab)^n$ languages, we used the same strategy to generate training sets with maximum lengths of ten and 20, respectively.

The measurement paradigms typically used for supervised learning tasks do not apply directly to grammatical domains. A grammar-induction system can infer the right word classes with relative ease, making the real test whether it forms recursive rules that let it correctly generalize to sentences longer than those in the training sample. Thus, in generating our test sets, we used maximum lengths of 15 and 20 for the adjective phrase and relative clause domains, respectively. For the parenthesis language, we generated all 65 legal strings of

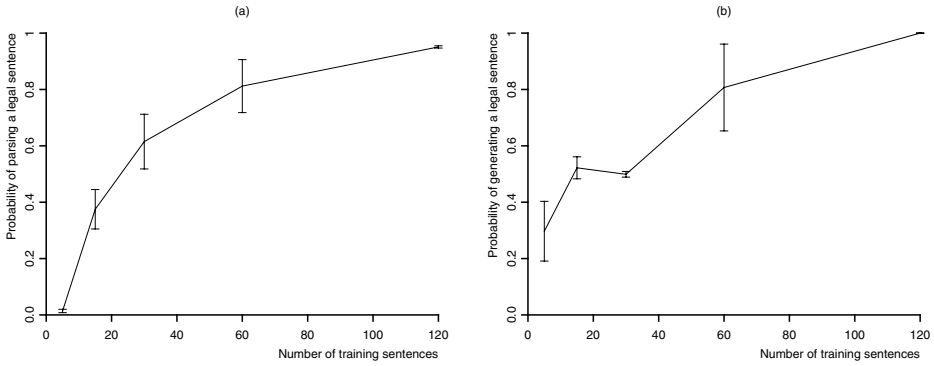


Fig. 1. Average learning curves for the adjective phrase grammar from Table 2, with (a) measuring the probability of parsing a legal test sentence and (b) the probability of generating a legal sentence

length 12 or less as positive test cases, and enumerated all 15 sentences of length 30 or less for the $(ab)^n$ language.

Another issue concerns the need to distinguish errors of omission (failures to parse sentences in the target language), which indicate an undergeneral grammar, from errors of commission (failures to generate only sentences in the target language), which indicate an overgeneral one. To estimate these terms, we used the target grammar T and each learned grammar L to generate sentence samples, and then determined their overlap. We estimated errors of omission from the fraction of sentences generated by T that were parsed by L , and errors of commission from the fraction of sentences generated by L that were parsed by T . On the average, an undergeneral grammar will produce a low score on the first measure, whereas an overgeneral one will produce a low score on the second.

3.2 Experimental Results

We intended our initial study to show that GRIDS could actually induce accurate grammars for all four domains. However, we were also interested in the *rate* of learning, so we explicitly varied the number of training sentences available to the system, at each level measuring the two accuracies of the learned grammar, averaged over 20 different training sets.

Figure 1 presents the learning curves for the adjective phrase grammar from Table 2, with (a) showing results on the first measure, the probability of parsing a legal test sentence, and (b) showing those for the second, the probability of generating a sentence parsed by the target grammar. The curves show both the average accuracy and 95% confidence intervals as a function of different numbers of training sentences. After 120 training cases, the learned grammars cover 95% of the positive test set, and all generated strings are legal.

Somewhat different results occurred with the relative clause language, as shown in Figure 2. As before, the probability of parsing the 500 legal test sentences increases with experience, though with many fewer examples, reaching

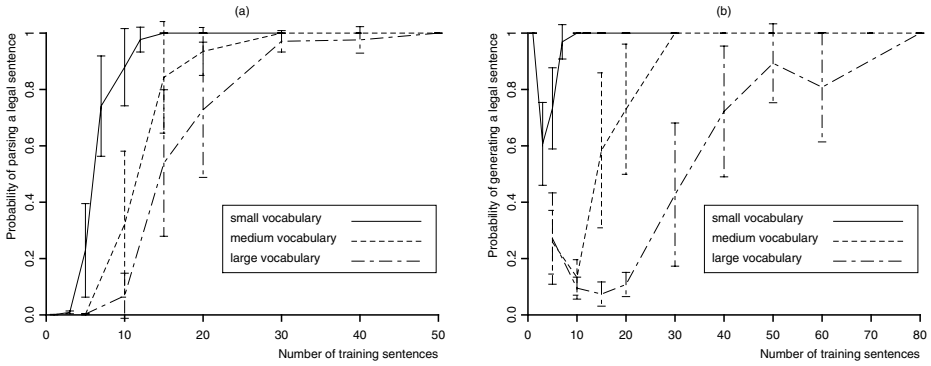


Fig. 2. Learning curves for the relative clause grammar from Table 2, and for analogous grammars that involve larger word classes, with (a) measuring the probability of parsing a legal test sentence and (b) the probability of generating a legal sentence

100% after only 15 training items. However, in this case GRIDS' probability of generating a legal sentence starts at 100%, falls to below 60% by the fourth case, then rebounds to perfect accuracy after processing 11 training sentences.

Experimental results for the parenthesis balancing language (not shown here) are analogous to those for adjective phrases, and the learning curves for the $(ab)^n$ language follow a very similar pattern, though the slopes are different. Clearly, one goal of future research should be to explain the underlying causes of these distinctive patterns, as well as the widely differing rates of learning.

Although our test grammars are simple compared to those encountered in natural languages, their complexity is comparable to others reported in the literature. Nevertheless, it would be good to understand the ability of the methods embodied in GRIDS to scale to more difficult induction tasks. To this end, we carried out an additional experiment in which we increased the size of word classes. In particular, we extended the relative clause grammar from Table 2, which included two verbs, three nouns, and one relative pronoun, by doubling and tripling the number of words in each of these categories.

Figure 2 compares the learning curves for these domains, using the two performance measures described earlier. Although increasing the size of the word classes slows down the learning process, the reduction in learning rate seems quite reasonable. Specifically, the number of training sentences required to reach perfect accuracy appears to be no more than linear in the size of the word classes. Also, this factor seems to affect both performance measures equally.

4 Discussion

Our approach to learning shares some of its central features with earlier work on grammar induction. We have already noted GRIDS' debt to Wolff's (1982) SNPR system, which also carried out heuristic search using operators for creating and merging symbols, and which used an evaluation function that traded off

a grammar's simplicity and its ability to 'compress' the training data. Cook, Rosenfeld, and Aronson's (1976) early work grammar induction also used an operator for creating nonterminal symbols, combined with hill-climbing search directed by a evaluation function similar in spirit to Wolff's.

Stolcke (1994) has carried out more recent research along similar lines, independently developing a grammar-induction algorithm that shares GRIDS' starting representation and its operations for symbol merging and creation. His system's evaluation metric also trades off a grammar's simplicity with its ability to account for observed sentences, but it learns probabilistic context-free grammars and processes training sentences incrementally. Grünwald (1996) has also developed an algorithm that uses a description-length score to direct search for 'partial' grammars, again invoking operators for term creation and merging.

The bias toward simplicity has arisen in other grammar-induction research, some quite different in overall control structure. Examples include enumerative algorithms that consider simpler grammars before more complex ones, as well as methods that start with a randomly generated grammar and invoke simplicity measures to direct hill-climbing search. Not all work on grammar induction relies on the simplicity bias, but the idea plays a recurring role in the literature. The literature also contains many formal claims about language 'learnability' under various conditions. Neither positive or negative results of this sort are relevant to our work, since we care not about guarantees but about practical methods.

Undoubtedly, we can improve the GRIDS algorithm along many fronts. For instance, it assumes that each word belongs to only one category, whereas in natural languages the same word can serve as several parts of speech. Also, an impediment to larger-scale studies is that the run time of the initial 'merge' operations increases with the square of the number of words. One strategy for dealing with the many possible merges involves trying only pairs with high scores on some heuristic measure, perhaps computed over co-occurrence statistics. Another response would be to develop an incremental version of GRIDS that processes only a few training sentences at a time and expands the grammar as necessary. We plan to explore both approaches to improving computational efficiency.

We cannot yet draw final conclusions about the role played by GRIDS' simplicity bias, as there exist other formulations of this idea not covered by our experimental evaluation. Nor can we yet tell whether other operators, or other organizations of the search process, will yield better or worse results. Clearly, more work remains to be done, but the results to date suggest the notion of simplicity has an important role to play in the acquisition of grammatical knowledge.

References

1. Cook, C. M., Rosenfeld, A., & Aronson, A. (1976). Grammatical inference by hill climbing. *Informational Sciences*, 10, 59–80.
2. Grünwald, P. (1996). A minimum description length approach to grammar inference. In S. Wermter, E. Riloff, & G. Scheler (Eds.) *Connectionist, statistical and symbolic approaches to learning for natural language processing*. Lecture Notes in Computer Science, 1040. Berlin: Springer-Verlag.

3. Stolcke, A. (1994). *Bayesian learning of probabilistic language models*. Doctoral dissertation, Division of Computer Science, University of California, Berkeley.
4. VanLehn, K., & Ball, W. (1987). A version space approach to learning context-free grammars. *Machine Learning*, 2, 39–74.
5. Wolff, J. G. (1982). Language acquisition, data compression and generalization. *Language & Communication*, 2, 57–89.

Partially Supervised Text Classification: Combining Labeled and Unlabeled Documents Using an EM-like Scheme

Carsten Lanquillon

DaimlerChrysler Research and Technology
D-89013 Ulm, Germany

Phone: +49 731 505 2809, Fax: +49 731 505 4210

carsten.lanquillon@daimlerchrysler.com

Abstract. Supervised learning algorithms usually require large amounts of training data to learn reasonably accurate classifiers. Yet, in many text classification tasks, labeled training documents are expensive to obtain, while unlabeled documents are readily available in large quantities. This paper describes a general framework for extending any text learning algorithm to utilize unlabeled documents in addition to labeled document using an Expectation-Maximization-like scheme. Our instantiation of this partially supervised classification framework with a similarity-based single prototype classifier achieves encouraging results on two real-world text datasets. Classification accuracy is reduced by up to 38% when using unlabeled documents in addition to labeled documents.

1 Introduction

With the enormous growth of on-line information available through the World Wide Web, electronic news feeds, digital libraries, corporate intranets, and other sources, the problem of automatically classifying text documents into predefined categories is of great practical importance in many information organization and management tasks.

This classification problem can be solved by applying supervised learning algorithms which learn reasonably accurate classifiers when provided with enough labeled training examples [4,14]. For complex learning tasks, however, providing sufficiently large sets of labeled training examples becomes prohibitive because hand-labeling examples is expensive. Therefore, an important issue is to reduce the need for labeled training documents. As shown in [9], a promising approach in text domains is to use *unlabeled* documents in addition to *labeled* documents during the learning process. While labeled documents are expensive to obtain, unlabeled documents are often readily available in large quantities.

Why does using unlabeled data help? As pointed out by [9] and [6], it is well known in information retrieval that words in natural language occur in strong co-occurrence patterns [13]. While some words are likely to co-occur in

one document, others are not. When using unlabeled documents we can exploit information about word co-occurrences that is not accessible from the labeled documents alone. This information can increase classification accuracy.

Nigam *et al.* [9] use a multinomial Naïve Bayes classifier in combination with the Expectation Maximization (EM) algorithm [3] to make use of unlabeled documents in a probabilistic framework. They show that augmenting the available labeled documents with unlabeled documents can significantly increase classification accuracy. In this paper we drop the probabilistic framework and extend the EM-like scheme to be used with any text classifier.

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to text classification and two traditional learning algorithms which are used later on. In Section 3, our algorithm for combining labeled and unlabeled documents in an EM-like fashion is described. Some experimental results are presented in Section 4. Section 5 lists some related work, and Section 6 concludes this paper.

2 Text Classification

The task of text classification is to automatically classify documents into a pre-defined number of classes. Each document can be in multiple, exactly one, or no class. In the experiments presented in Section 4, the task is to assign each document to exactly one class. Using supervised learning algorithms in this particular setting, a classifier can try to represent each class simultaneously. Alternatively, each class can be treated as a separate binary classification problem where each binary problem answers the question of whether or not a document should be assigned to the corresponding class [6].

2.1 Document Representation

In information retrieval, documents are often represented as feature vectors, and a subset of all distinct words or word stems occurring in the given documents are used as features. Words that frequently occur in many documents (*stop words* like "and", "or" etc.) or words that occur only in very few documents may be removed. Further, measures such as the *average mutual information* with the class labels can be used for feature selection [15]. Each feature is given a weight which depends on the learning algorithm at hand. This leads to an attribute-value representation of text. Possible weights are, e.g., binary indicators for the presence or absence of features, plain feature counts—*term frequency (tf)*—or more sophisticated weighting schemes, such as multiplying each term frequency with the *inverted document frequency (idf)* [12]. Finally, each feature vector may be normalized to unit length to abstract from different document lengths.

2.2 Learning Algorithms

A variety of text learning algorithms have been studied and compared in the literature, e.g. see [4] and [14].

Naïve Bayes Classifier For comparison we apply the multinomial Naïve Bayes classifier which uses the term frequency as feature weights as described in [9]. The idea of the Naïve Bayes classifier is to use the joint probabilities of words (features) and classes to estimate the probabilities of the classes given a document. A document is then assigned to the most probable class.

Single Prototype Classifier Further, we use a similarity-based method based on *tfidf* weights which we denote as *single prototype classifier (SPC)*. It is a variant of Rocchio’s method for relevance feedback [10] applied to text classification and is also described as the *Find Similar* algorithm in [4]. The classifier models each class with exactly one prototype computed as the average (centroid) of all available training documents. We use a scheme for setting feature weights which is denoted as *ltc* in SMART [11] notation. A document is assigned to the class of the prototype to which it has the largest cosine similarity.

3 Partially Supervised Learning

This section describes a family of partially supervised learning algorithms for combining labeled and unlabeled documents, extending the work of [9].

3.1 General Framework

A general approach for utilizing information given by unlabeled data is to apply some form of clustering. Treating the class labels of the unlabeled documents as missing values, an EM-like scheme can be applied as described below. Table 1 gives an outline of this framework.

Given a set of training documents D , for some subset of the documents $d_i \in D^l$ we know the class label y_i , and for the rest of the documents $d_i \in D^u$, the class labels are unknown. Thus we have a disjoint partitioning of our training documents into a labeled set and an unlabeled set of documents $D = D^l \cup D^u$. The task is to build a classifier based on the training documents, D , for predicting the class label of unseen unlabeled documents.

First, an initial classifier, H , is build based only on the labeled documents, D^l . Then the algorithm iterates the following three steps until the class memberships given to the unlabeled documents, D^u , by the current classifier, H , do not change from one iteration to the next. Corresponding to the **E-step**, the current classifier, H , is used to obtain classification scores for each unlabeled document. The classifier may respond with any type of classification scores, they need not be probabilistic. In order to abstract from the classifier’s response, in the next step we transform these scores into class memberships, yielding a class membership matrix, $U^u \in [0, 1]^{c \times |D^u|}$, where c is the number of classes. The sum of class memberships of a document over all classes is assumed to be one. Possible transformations are, for instance, normalizing the scores or using hard memberships, e.g. setting the largest score to one and all other scores to zero. The transformation function should depend on the classifier at hand such that it knows how to make use of the class membership matrix, U^u . Using hard memberships always

Table 1. EM-like algorithmic framework for partially supervised learning

-
- **Inputs:** Sets D^l and D^u of labeled and unlabeled documents.
 - Build initial classifier, H , based only on the labeled documents, D^l .
 - Loop while classifying the unlabeled documents, D^u , with the current classifier, H , changes as measured by the class memberships of the unlabeled documents, U^u :
 - **(E-step)** Use the current classifier, H , to evaluate classification scores for each unlabeled document.
 - Transform classification scores into class memberships of the unlabeled documents, U^u .
 - **(M-step)** Re-build the classifier, H , based on labeled documents, D^l , and unlabeled documents, D^u , with labels obtained from U^u .
 - **Output:** Classifier, H , for predicting class labels of unseen unlabeled documents.
-

allows us to use any traditional classifier. Now, provided with the class membership matrix, U^u , a new classifier, H , can be build from both, the labeled and unlabeled documents. This corresponds to the **M-step**. The final classifier, H , can then be used to predict the class labels of unseen test examples.

3.2 Instantiations

In order to apply this algorithmic framework, the underlying classification algorithm and the function for transforming classification scores have to be specified.

Naïve Bayes Classifier When using a Naïve Bayes classifier and leaving the resulting probabilistic classification scores unchanged, we end up with the algorithm given in [9]. This instantiation has a strong probabilistic framework and is guaranteed to converge to a local minimum as stated by [9].

Single Prototype Classifier Next, we will use the single prototype classifier in combination with a transformation of classification scores into hard class memberships. Hence, this instantiation of our partially supervised algorithmic framework turns out to be a variation of the well known hard *k-means* clustering algorithm [7]. The difference is that the memberships of the labeled documents remain fixed during the clustering iterations. The traditional k-means algorithm is guaranteed to converge to a local minimum after a finite number of iterations. What about our partially supervised variant?

The proof of convergence for the traditional k-means algorithm is based on the fact that there is only a finite number of hard partitionings of training documents into classes and that the sum of squared distances between prototypes and training documents, J , does not increase while iteratively updating the class memberships and the prototypes. Therefore, the algorithm must converge in a finite number of steps.

The calculation of cluster prototypes based on training documents and their hard class labels is the same in our partially supervised algorithm. Hence, this

step does not increase J . As mentioned above, the update rule for the class membership matrix in our algorithm differs from the traditional k-means algorithm. The class labels of the labeled documents remain fixed while the unlabeled documents are assigned to the closest prototype. The latter is equivalent to the traditional k-means algorithm and thus does not lead to an increase in J either. Further, note that fixed class memberships cannot cause J to change. Thus, our partially supervised algorithm will also converge to a local minimum after a finite number of steps.

4 Experimental Results

This section gives empirical evidence that combining labeled and unlabeled documents with certain text classifiers using the algorithmic framework in Table 1 can improve traditional text classifiers. Experimental results are reported on two different text corpora which are available at <http://www.cs.cmu.edu/~textlearning>. We use a modified version of the *Rainbow* system [8] to run our experiments. Following the setups in [9], we run the experiments with the partially supervised single prototype classifier as described Section 3. The results are compared to the partially supervised Naïve Bayes approach as given in [9].

4.1 Datasets and Protocol

The 20 Newsgroups dataset consists of 20017 articles divided almost evenly among 20 different UseNet discussion groups. The task is to classify an article into the one of the twenty newsgroups to which it was posted. When tokenizing the documents, UseNet headers are skipped, and tokens are formed from contiguous alphabetic characters. We do not apply stemming, but remove common stop words. While all features are used in the experiments with the Naïve Bayes classifier, for the single prototype classifier, we limit the vocabulary to the 10000 most informative words, as measured by average mutual information with the class labels. We create a test set of 4000 documents and an unlabeled set of 10000 documents. Labeled training sets are formed by partitioning the remaining 6000 documents into non-overlapping sets. All sets are created with equal number of documents per class. Where applicable, up to ten trials with disjunct labeled training sets are run for each experiment. Results are reported as averages over these trials.

The WebKB dataset contains 8145 web pages gathered from four university computer science departments. Only the 4199 documents of the classes *course*, *faculty*, *project*, and *student* are used. The task is to classify a web page into the appropriate one of the four classes. We do not apply stemming and stop-word removal. The vocabulary is limited to the top 300 words according to average mutual information with the class labels in all experiments. To test in *leave-one-university-out* fashion, we create four test sets, each containing all the pages from one of the four complete computer science departments. For each test set, an unlabeled set of 2500 pages is created by randomly selecting from

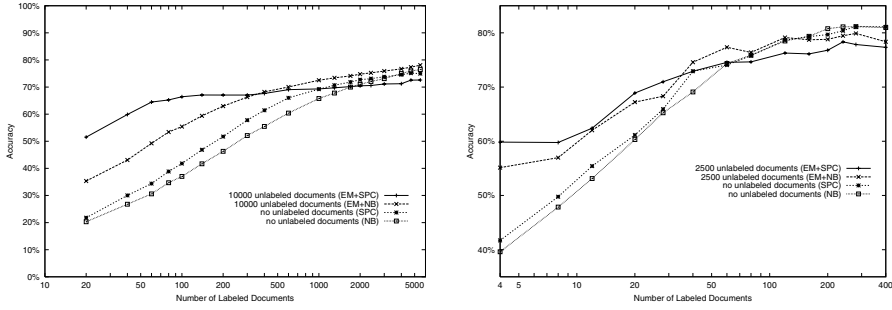


Fig. 1. Classification accuracy of the partially supervised learning framework (EM) using the Naïve Bayes classifier (NB) and the single prototype classifier (SPC) compared to the traditional classifiers on the 20 Newsgroups dataset (left) and on the WebKB dataset (right). Note the magnified vertical scale on the right

the remaining pages. Different non-overlapping labeled training sets are created from the remaining web pages. Results are reported as averages over the four different test sets.

4.2 Results

Figure 1 shows the effect of using the partially supervised learning framework with the Naïve Bayes classifier (NB) and the single prototype classifier (SPC) on the 20 Newsgroups dataset and the WebKB dataset. The horizontal axis indicates the amount of labeled training data on a log scale. Note that, for instance, 20 training documents for the 20 Newsgroups and four documents for the WebKB dataset correspond to one training document per class. The vertical axis indicates the average classification accuracy on the test sets. We vary the number of labeled training documents for both datasets and compare the results to the traditional classifiers which do not use any unlabeled documents.

In all experiments, the partially supervised algorithms perform substantially better when the amount of labeled training documents is small. For instance, with only 20 training examples for the 20 Newsgroups dataset, the partially supervised SPC reaches about 52% accuracy while the traditional SPC achieves 22%. Thus, the classification error is reduced by about 38%. For the NB, accuracy increases from 20% to about 35% when using unlabeled documents with 20 labeled training examples. For the WebKB dataset, the performance increase is much smaller, especially for the SPC. However, note that there are four times less unlabeled documents for the experiments on this dataset. As can be expected, the more labeled documents are available, the smaller the performance increase. Note that especially for the SPC, accuracy even degrades when using unlabeled documents with a lot of labeled documents. We hypothesize that when the number of labeled documents is small, the learning algorithm is desperately in need for help and makes even good use of uncertain information as provided

by unlabeled documents. However, when the accuracy is already high without any unlabeled documents, i.e. when there are enough labeled documents, adding uncertain information by means of unlabeled documents does not help but rather hurts classification accuracy.

5 Related Work

The family of Expectation-Maximization (EM) algorithms and its application to classification is broadly studied in the statistics literature. R.J.A. Little [3] mentions the idea of using an EM-like approach to improve a classifier by treating the class labels of unlabeled documents as missing values. Emde describes a conceptual clustering algorithm that tries to take advantage of the information inherent to the unlabeled data in a setting where the number of labeled data is small [5]. Blum and Mitchell [2] use co-training to make use of labeled and unlabeled data in the case that each example has at least two redundantly sufficient representations. Bensaid and Bezdek try to use information inherent to the labeled data to help clustering the unlabeled data [1]. In current work by Bensaid and the author, this approach is applied to text classification. As mentioned in Sections 1 and 3, this paper describes a generalization of the work done by Nigam *et al.* [9]. They use a multinomial Naïve Bayes classifier in combination with the EM-algorithm to make use of unlabeled documents. Joachims explores transductive support vector machines for text classification [6]. This approach uses the unlabeled test documents in addition to the labeled training documents to better adjust the parameters of the support vector machine. Although designed for classifying the documents of just this test set, the resulting support vector machine could as well be applied to classify new, unseen documents as done in this paper. However, as yet there is no empirical evidence of how well this works.

6 Conclusions and Future Work

This paper presents a general framework for partially supervised learning from labeled and unlabeled documents using an EM-like scheme in combination with an arbitrary text learning algorithm. This is an important issue when hand-labeling documents is expensive but unlabeled documents are readily available in large quantities.

Empirical results with two real-world text classification tasks and a similarity-based single prototype classifier show that this EM-scheme can successfully be applied to non-probabilistic classifiers. The applied instantiation of our framework is a variant of the traditional hard k-means clustering algorithm where the class memberships of some training documents, namely the labeled documents, are fixed. The single prototype classifier seems to be well suited for classification tasks where the number of labeled documents is very scarce. For larger numbers of labeled documents, the Naïve Bayes classifier is superior.

Adding unlabeled documents to a larger number of labeled training documents may even hurt classification accuracy when using the single prototype classifier. Future work will focus on preventing the unlabeled documents from degrading performance. An interesting approach is to introduce a weight to adjust the contribution of unlabeled documents as discussed in [9].

So far we applied only very simple learning algorithms because the successful application of more sophisticated methods seems doubtful when only very few labeled training documents are present. Nevertheless, other learning algorithms are being tested in current research. Our conjecture is that this framework works well for learning algorithms that aggregate document information for each class into a single representative like the two methods applied in this paper. By contrast, approaches like the nearest neighbor rule are likely to fail since they do not generalize and thus cannot exploit information inherent to unlabeled documents.

Acknowledgements

Thanks to Tom Mitchell, the CMU text learning group, and Amine Bensaid for help and discussions. Special thanks to Kamal Nigam for providing his results and scripts to setup the experiments.

References

1. A. Bensaid and J. Bezdek. Semi-supervised point-prototype clustering. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 12(5):625–643, 1998. 235
2. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Annual Conference on Computational Learning Theory (COLT-98)*, 1998. 235
3. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Stat. Soc., Series B*, 39:1–38, 1977. 230, 235
4. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representation for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, 1998. 229, 230, 231
5. W. Emde. Inductive learning of characteristic concept descriptions from small sets of classified examples. In *Proc. 7th European Conf. on Machine Learning*, 1994. 235
6. T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th Int. Conf. on Machine Learning*, 1999. 229, 230, 235
7. J. MacQueen. Some methods for classification and analysis of multivariate observations. In L.M. Le Cam and J. Neyman, editors, *Proc. 5th Berkeley Symp. Math. Stat. Prob.*, volume I, pages 281–297. Univ. of California Press, 1967. 232
8. A.K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996. 233
9. K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 1999. To appear. 229, 230, 231, 232, 233, 235, 236

10. J.J. Jr. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971. 231
11. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989. 231
12. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. 230
13. C. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, 1977. 229
14. Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceeding of ACM SIGIR Conference*, 1999. 229, 230
15. Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. 14th Int. Conf. on Machine Learning*, 1997. 230

Toward an Explanatory Similarity Measure for Nearest-Neighbor Classification

Mathieu Latourrette

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier,
UMR 5506 Université Montpellier II CNRS,
161, rue Ada 34392 Montpellier FRANCE
`latourre@lirmm.fr`

Abstract. In this paper, a new similarity measure for nearest-neighbor classification is introduced. This measure is an approximation of a theoretical similarity that has some interesting properties. In particular, this latter is a step toward a theory of concepts formation. It renders identical some examples that have distinct representations. Moreover, these examples share some properties relevant for the concept undertaken. Hence, a rule-based representation of the concept can be inferred from the theoretical similarity. Moreover, in this paper, the approximation is validated by some preliminary experiments on non-noisy datasets.

1 Introduction

Learning to classify objects is a fundamental problem in artificial intelligence and other fields, one which has been addressed from many sides. This paper deals with the nearest-neighbor methods (Cover and Hart [6]), also known as exemplar-based (Salzberg [8]) or instance-based learning programs (Aha et al. [1]). These algorithms classify each new example according to some past experience (a set of examples provided with their labels) and a measure of similarity between the examples. Actually, they assign to each new example the label of its nearest known example.

At first glance, similarity seems a rather intuitive notion. Examples are denoted by some properties and are similar if they have some properties in common. Thus, the more similar examples are, the more likely they share some relevant properties for the concept to learn. When the size of the dataset increases, new examples and their nearest neighbors become more and more similar. And, in the limit, classification is accurate.

Such a convergence has been studied many times. Despite positive results, such a similarity has been criticized for not being explanatory. It does not identify among the properties shared by some similar examples the ones that are relevant for the concept undertaken.

This paper is focused on the problem of explanation. The concepts to learn are assumed to have some rule-based representations. In this case, the relevant

properties are the preconditions of these rules. To explain the classification of each example, the proposed similarity measure enables to infer a rule-based representation of the concept undertaken.

For that matter, we suggest that examples are similar because they satisfy the same rules and, no longer, because the properties they are denoted by are somewhat similar. Such a similarity measure relies on the rules characterizing the concept undertaken. For a classification task, such a similarity is theoretical as the rules are unknown. However, this similarity can be approximated from each dataset and some rules inferred from this latter.

This paper is organized as follows. §2 summarizes some notations and definitions. §3 introduces the theoretical similarity. §4 is devoted to its approximation and to the resulting classifier. §5 deals with some related research.

2 Preliminaries

Let us introduce a few useful definitions. Let $F: \{f_1, f_2, \dots, f_n\}$ be a set of features, where each feature f_i can take values in its domain Dom_i : a finite unordered set. An example $x: (x_1, x_2, \dots, x_n)$ is characterized by an instantiation x_i of each feature f_i . The example x satisfies the conjunction $c_x: f_1 = x_1 \wedge f_2 = x_2 \wedge \dots \wedge f_n = x_n$. Let U denote the universe: the set of all the possible examples. Considering a finite unordered set L of labels, a concept C is a function from U to L . An exemplar e is a couple $(x, C(x))$ of an example and its label. Let E be the set of all the exemplars. A dataset D is a subset of E .

For example, for the monk1 dataset, examples are represented by 6 features. The domain of f_1, f_2 and f_4 is $\{1, 2, 3\}$. The domain of f_3 and f_6 is $\{1, 2\}$ and the domain of f_5 $\{1, 2, 3, 4\}$. The set of labels is $\{0, 1\}$. The universe contains 432 ($=3 \times 3 \times 2 \times 3 \times 4 \times 2$) examples. The concept undertaken is the boolean function $(f_1 = f_2) \vee (f_5 = 1)$. Two exemplars are:

$$e_1: ((1,1,1,1,1,1), 1) \quad \text{and} \quad e_2: ((2,2,1,3,2,1), 1)$$

Definition 1. A rule r is a partial function from U to a particular label l_r denoted by $c_r \implies l_r$. It associates to each example x such that $c_x \implies c_r$ the label l_r . c_r is a conjunction of conditions upon the values of each feature. For each feature f_i , its value is required to be in a subset (not empty) of Dom_i .

On the monk1 problem, a rule r^* is:

$$f_1 \in \{1, 2\} \wedge f_2 \in \{1, 2\} \wedge f_3 \in \{1\} \wedge f_4 \in \{1, 3\} \wedge f_5 \in \{1, 2\} \wedge f_6 \in \{1\} \implies 1$$

Let us denote such a rule by:

$$\{1, 2\}, \{1, 2\}, \{1\}, \{1, 3\}, \{1, 2\}, \{1\} \implies 1$$

An example x or an exemplar $e = (x, l)$ is covered by a rule r iff $c_x \implies c_r$. Let U_r (resp. E_r) be the subset of the examples (resp. exemplars) covered by r . An exemplar refutes r if it is covered by r but has a different label. r is coherent with the dataset D if there is no exemplar in D to refute r . r is coherent with the concept C if all the exemplars of E_r have the label of r . A rule r_1 is more specific than a rule r_2 if $U_{r_1} \subset U_{r_2}$. In this case, r_2 is more general than r_1 .

Definition 2. Let the generalization of each subset s of exemplars of the same label be $G(s)$ the most specific rule covering s and coherent with s .

Notice that the generalization of a subset of exemplars is unique. Actually, the label of a generalization $G(s)$ is the label of the exemplars in s . And, for each feature f_i , the value x_i of an example covered by $G(s)$ is required to be in the union of the values of f_i appearing in s . For example, $G(\{e_1, e_2\})$ is r^* .

The reader shall see that the operator G satisfies the two properties:

1. **(monotonicity)** The generalization of a subset of exemplars covered by a rule coherent with a concept C is coherent with C .
2. **(stability)** Let C be a concept, r a rule and e an exemplar. If $\forall e' \in E/r$, $G(\{e, e'\})$ is coherent with C then $G(\{e\} \cup E/r)$ is coherent with C .

In the reminder of this paper, these two properties will be the only ones required for G . As they are rather natural for an operator of generalization, we guess that our approach can be extended to many other representation languages.

3 Similarity with Respect to a Concept

3.1 Definition

This section is devoted to the definition of the theoretical similarity with respect to a concept C . For that matter, we assume that C is *well-defined*.

Definition 3. A well-defined concept C is a function from a universe U to a set of labels L characterized by a set of rules R . Thus, for each example x of U and each rule r ($c_r \implies l_r$) of R covering x , there is $l_r = C(x)$.

Many sets of rules characterize a concept. However, as we suggest that examples are similar because they satisfy the same rules, we have to choose these rules.

Definition 4. Let the definition of a well-defined concept C be the set of all the most general rules coherent with C . For each exemplar e , let $Def_C(e)$ be the subset of the rules covering e and defining C .

Notice that the rules defining a concept contain only relevant properties. Actually, all the conditions that could have been dropped from the maximal rules have already been. The definition of the monk1 concept is:

$\{1\}$,	$\{1\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{1,2,3,4\}$,	$\{1,2\} \implies 1$	(I)
$\{2\}$,	$\{2\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{1,2,3,4\}$,	$\{1,2\} \implies 1$	(II)
$\{3\}$,	$\{3\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{1,2,3,4\}$,	$\{1,2\} \implies 1$	(III)
$\{1,2,3\}$,	$\{1,2,3\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{1\}$,	$\{1,2\} \implies 1$	(IV)
$\{1\}$,	$\{2,3\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(V)
$\{2\}$,	$\{1,3\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(VI)
$\{3\}$,	$\{1,2\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(VII)
$\{2,3\}$,	$\{1\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(VIII)
$\{1,3\}$,	$\{2\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(IX)
$\{1,2\}$,	$\{3\}$,	$\{1,2\}$,	$\{1,2,3\}$,	$\{2,3,4\}$,	$\{1,2\} \implies 0$	(X)

Definition 5. *The neighborhood of an exemplar e with respect to a well-defined concept C is defined as follows: $N_C(e) = \{e' \in E \mid Def_C(e) \cap Def_C(e') \neq \emptyset\}$.*

Our similarity between two exemplars is measured between their neighborhoods. We choose the ratio between the numbers of exemplars common to the two neighborhoods and the number of examples belonging to one of them:

Definition 6. *Considering two exemplars e and e' , their similarity with respect to a well-defined concept C is: $Sim_C(e, e') = \frac{|N_C(e) \cap N_C(e')|}{|N_C(e) \cup N_C(e')|}$*

3.2 An Accurate Similarity for Nearest-Neighbor Classification

Let two exemplars be equivalent if and only if their similarity is 1. First of all, notice that two equivalent exemplars have the same label.

Theorem 1. *Let C be a well-defined concept and e and e' two exemplars. If e and e' are equivalent, they have the same label.*

Proof. By definition of Sim_C , e and e' are equivalent iff $N_C(e) = N_C(e')$. As e belongs to its neighborhood, e belongs to $N_C(e')$. By definition of $N_C(e')$, there is a rule r of $Def_C(e')$ that covers e . Therefore, e and e' have the label of r .

Thus, if the dataset contains an equivalent exemplar for each new example, the nearest-neighbor rule is accurate.

Definition 7. *Let C be a well-defined concept, e an exemplar. Then, the class of equivalence of e considering Sim_C is: $Eq_C(e) = \{e' \in E \mid Sim_C(e, e') = 1\}$*

The number of classes of equivalent exemplars does not depend on the dataset (theorem 2). Therefore, when the size of the dataset increases, more and more classes are represented. And, in the limit, the classifier is accurate. For the monk1 concept, there are only 13 such classes and 432 exemplars.

Theorem 2. *Let C be a well-defined concept.*

$$\forall e \in E \quad Eq_C(e) = \{e' \in E \mid Def_C(e) = Def_C(e')\}$$

Proof. If $Def_C(e) = Def_C(e')$ then $N_C(e) = N_C(e')$ and $Sim_C(e, e') = 1$. Now, assume that $Sim_C(e, e') = 1$ (i.e. $N_C(e) = N_C(e')$) and let r be in $Def_C(e)$. Each exemplar e'' covered by r belongs to $N_C(e) = N_C(e')$. Thus, $G(\{e', e''\})$ is coherent (definition of $N_C(e')$ and monotonicity). It follows that $r'' = G(\{e'\} \cup E/r)$ is coherent (stability). As r is maximal, it means that r'' is r . Therefore, r belongs to $N_C(e')$. Hence, if $Sim_C(e, e') = 1$, then $Def_C(e) = Def_C(e')$.

3.3 An Explanatory Similarity

Considering such a similarity, each exemplar is equivalent to many others. Theorem 3 states that the generalizations of some equivalent exemplars are coherent with the concept. Therefore, among the properties shared by some equivalent exemplars, some of them are relevant. This is the reason why such a similarity is somewhat explanatory.

Theorem 3. *Let C be a well-defined concept.*

$$\forall e \in E, \quad G(Eq_C(e)) \text{ is coherent with } C.$$

Proof. Theorem 2 shows that all the exemplars of $Eq_C(e)$ are covered by the same rules: $Def_C(e)$. Their generalization $G(Eq_C(e))$ is thus more specific than each of the rules of $Def_C(e)$ and, therefore, coherent with the concept C .

In the example, e_2 satisfies the rule II only and is equivalent to all the examples that satisfy only this rule. Therefore, the generalization $G(Eq_C(e_2))$ is

$$\{2\}, \{2\}, \{1,2\}, \{1,2,3\}, \{2,3,4\}, \{1,2\} \implies 1$$

It requires each covered example to satisfy $f_1 = 1$, $f_2 = 1$ and $f_5 \neq 1$. The other conditions are trivial as each value is necessary in its domain. The two first properties are relevant for the concept. However, the last one is not. It is present to prevent the exemplars covered from satisfying the rule IV.

4 Application to Nearest-Neighbor Classification

The theoretical similarity depends on the definition of the concept undertaken. In a classification task, such a definition is unknown. However, the previous similarity can be approximated from a dataset.

4.1 An Approximated Similarity Measure

The approximation relies on the ability to approximate each neighborhood by:

Definition 8. *The neighborhood of an example e with respect to a dataset D is:*

$$N_D(e) = \{e' \in D \mid G(\{e, e'\}) \text{ is coherent with } D\}.$$

Actually, for each exemplar e , the approximated neighborhood $N_D(e)$ converges toward $N_C(e) \cap D$, when the size of the dataset increases. This result follows from the proposition:

Proposition 1. *Let C be a well-defined concept, D a dataset and $e \in D$.*

1. $N_C(e) \cap D \subset N_D(e)$
2. *The probability to be in $N_D(e)$ but not in $N_C(e) \cap D$ decreases when the size of the dataset increases.*
3. *In the limit, $D=E$ and $N_D(e) \subset N_C(e) \cap D$*

Proof. Proposition 1 follows from the monotonicity of G . Proposition 2 states that each generalization is more likely to be refuted when more exemplars are provided. When all the exemplars are provided, generalizations coherent with the dataset are also coherent with the concept, which explains proposition 3.

Therefore, for each exemplar e , the size of $N_C(e)$ is approximated by the average number of exemplars of D that belong to $N_D(e)$. Let us approximate the sizes of the intersection and of the union of two neighborhoods in the same way. The theoretical similarity is, then, approximated by:

Definition 9. *Considering two exemplars e and e' , their similarity with respect to the dataset D is:* $Sim_D(e, e') = \frac{|N_D(e) \cap N_D(e')|}{|N_D(e) \cup N_D(e')|}$

4.2 IBLG Classification

On these considerations, we developed a nearest-neighbor classifier based upon the approximated similarity measure and called IBLG (Instance-Based Learning from Generalization). Each new example has several neighborhoods whether it is assumed to have a particular label or another. Hence, IBLG has to compute the nearest-neighbor for each of the possible neighborhoods and choose the nearest one. The pseudo-code of IBLG is shown below. Its complexity is $O(N^3)$ where N is the size of the dataset.

```

For each label  $l$ ,
    initialize  $N_l$  as an empty list.
For each exemplar  $e = (x, l)$  in  $D$ ,
    compute and add the neighborhood  $N_D(e)$  to  $N_l$ 

classify(example  $x$ )
    for each label  $l$ 
        let  $e$  be the exemplar  $(x, l)$ 
        compute the neighborhood  $N_D(e)$ 
        retrieve its nearest neighborhood  $N_D(e')$  in  $N_l$ 
        let  $Sim_D(e, e')$  be the similarity of  $x$  for  $l$ 
    return a label of maximal similarity

```

4.3 Some Experimental Evidences

To validate our approach, some experiments have been carried out to compare IBLG with four other classifiers: CN2 (Clark and Niblett [4]) for rule induction, PEBLS (Cost and Salzberg [5]) and SCOPE (Lachiche and Marquis [7]) for nearest-neighbor. As default classifier, the nearest-neighbor classifier based upon the Hamming distance¹ has been chosen.

As IBLG has no parameter, we have chosen the default parameters of the other algorithms. However, SCOPE has three parameters that are automatically assessed to deal with noisy datasets. Here, datasets are non-noisy and these parameters left to their theoretical values.

The experiments are summarized figure 1. IBLG appears to be less sensitive to the concept undertaken. Therefore, with respect to the other methods, IBLG performs best for complex concepts.

5 Related Research

5.1 SCOPE Classification

SCOPE (Lachiche and Marquis [7]) is a nearest-neighbor algorithm introduced in 1998. It classifies each new example according to the label of its most numerous

¹ The Hamming distance counts the number of features whose values are different.

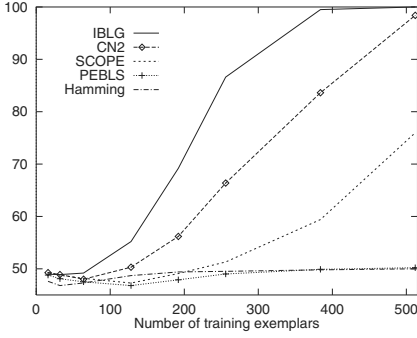
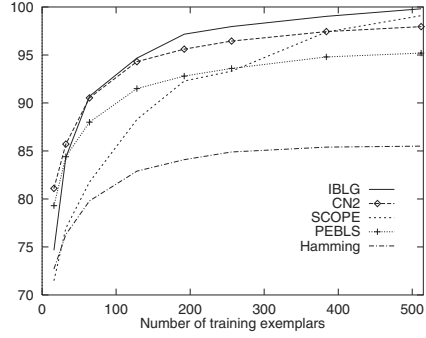
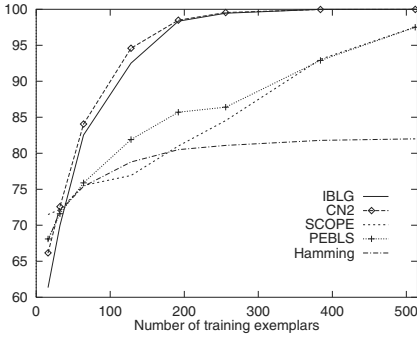
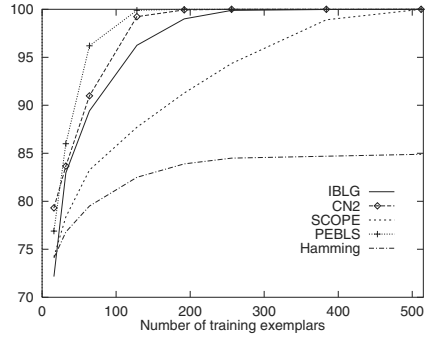
a) $C_a = \text{parity concept}$ b) $C_b = A \vee BC \vee DEF \vee GHIJ$ c) $C_c = ABCD \vee BCDE \vee CDEA \vee DEAB \vee EABC$ d) $C_d = ABC \vee BCD \vee ACD \vee ABD$

Fig. 1. Experimental learning curves for IBLG when target concepts are less and less complex boolean functions of 10 boolean features. Each measure is the average classification accuracy on the unseen examples for 25 trials. The parity concept denotes the parity of the number of features whose value is true among the five first features.

neighborhood. IBLG chooses the label of the most similar known neighborhood. The improvement may appear rather small. However, each neighborhood (a set of exemplars) carries much more information than its size. And, in this paper, this information has been shown to be relevant from both theoretical and experimental points of view.

5.2 Feature Weighting Methods

The usual similarity measure is inversely correlated to the average distance between the values of each feature. However, when too many irrelevant features describe the examples, this similarity is irrelevant as well. The most studied solution is to weight the contribution of each feature to the overall similarity.

The problem is, then, to estimate from the dataset how relevant is a feature or even a value. For example, for the context similarity measure, Biderman ([2])

emphasizes that examples sharing a particular value are perceived more similar if this value is uncommon in the dataset. However, the problem of relevance is still open. The problems raised in this research area are reviewed in (Blum and Langley [3]) and the main contributions to nearest-neighbor methods in (Wettschereck et al. [10]).

For example, PEBLS (Cost and Salzberg [5]) is one of the state-of-the-art nearest-neighbor classifiers for symbolic features. It relies on the Value Difference Metric (Stanfill and Waltz [9]) and outperforms the Hamming classifier on most of the usual datasets but not all. The poor performances of PEBLS on the parity concept (cf fig. 1a) emphasize the difficulties encountered by this approach of similarity.

6 Conclusion

In this paper, we have introduced a new way to measure the similarity between some examples. This similarity measure has some theoretical advantages over the usual ones. Firstly, it becomes more and more accurate when the size of the dataset increases. And, in the limit, similar examples do have the same label. Therefore, convergence does not follow only from the ability to retrieve more and more similar examples. Secondly, this similarity is explanatory: it allows to build a rule-based representation of the concept undertaken. Determining whether these rules make an accurate rule-based classifier will be the scope of another paper. But, preliminary results are promising.

References

1. D. Aha, D. Kibler and M. Albert, Instance-based learning algorithms, *Machine Learning*, 6(1):37-66, 1991. 238
2. Y. Biderman, A Context Similarity measure, *Lecture Note in AI: ECML'94*, Springer Verlag, 1994. 244
3. A. L. Blum and P. Langley, Selection of relevant features and examples in machine learning, *Artificial Intelligence*, 245-271, Elsevier Science BV, 1997. 245
4. P. Clark and T. Niblett, The CN2 induction algorithm, *Machine Learning*, 3(4):261-283, 1989. 243
5. S. Cost and S. Salzberg, A weighted nearest-neighbor algorithm for learning with symbolic features, *Machine Learning*, 10:57-78, 1993. 243, 245
6. T. Cover and P. Hart, Nearest-neighbor pattern classification, *IEEE Transactions on Information Theory*, 13(1):21-27, 1967. 238
7. N. Lachiche and P. Marquis, Scope classification: An instance-based learning algorithm with a rule-based characterization, *In Proceedings of ECML'98*, pages 268-279, Chemnitz, Germany, LNAI 1398, Springer, 1998. 243
8. S. Salzberg, Learning with nested generalized exemplars, *Norwell, MA: Kluwer Academic Publishers*, 1990. 238
9. C. Stanfill, D. Waltz, Toward memory-based reasoning, *Communications of the ACM*, 29:1213-1228, 1986. 245
10. D. Wettschereck, D. W. Aha and T. Mohri, A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms, *Artificial Intelligence Review*, 11:273-314, 1996. 245

Relative Unsupervised Discretization for Regression Problems

Marcus-Christopher Ludl¹ and Gerhard Widmer^{1,2}

¹ Austrian Research Institute for Artificial Intelligence, Vienna

² Department of Medical Cybernetics and Artificial Intelligence,
University of Vienna, Austria

Abstract. The paper describes a new, context-sensitive discretization algorithm that combines aspects of unsupervised (class-blind) and supervised methods. The algorithm is applicable to a wide range of machine learning and data mining problems where continuous attributes need to be discretized. In this paper, we evaluate its utility in a regression-by-classification setting. Preliminary experimental results indicate that the decision trees induced using this discretization strategy are significantly smaller and thus more comprehensible than those learned with standard discretization methods, while losing only minimally in numerical prediction accuracy. This may be a considerable advantage in machine learning and data mining applications where comprehensibility is an issue.

1 Introduction

In the area of classification learning, there has been quite some research on attribute discretization in recent years, both regarding unsupervised (class-blind) and supervised methods – see, e.g., [1,5,6,9,11,13]. Some authors have also produced detailed studies of different discretization criteria used in “on-the-fly” discretization in induction, for instance, in decision tree learning algorithms [2] or in Bayesian classifiers [3]. While discretization is strictly necessary for induction algorithms that cannot handle numeric attributes directly (e.g., decision table algorithms or simple Bayesian classifiers), it has been shown that pre-discretizing continuous attributes — even when used in induction algorithms that can actually handle continuous features — can improve both the classification accuracy and the interpretability of the induced models.

Whereas in unsupervised discretization the attribute in question is discretized with simple, class-blind procedures, supervised discretization also takes class information into account, thereby possibly constructing split points that might be missed by a class-blind algorithm. [1] gives a good overview.

Recently, there have also been some investigations into the use of discretization for a *regression-by-classification* paradigm [12], where regression is converted into a classification problem by abstracting the continuous target attribute into discrete intervals. The work presented here falls into this latter category. We describe a new, context-sensitive discretization algorithm that can be used in both supervised and unsupervised settings. We evaluate the algorithm by using

it as the basis for a regression-by-classification system. Preliminary experimental results to be presented in section 4 demonstrate that the decision trees induced using this discretization strategy are significantly smaller than those learned with standard discretization methods, while losing only minimally in prediction accuracy (measured in terms of numeric error). We think this can be a considerable advantage in machine learning and data mining applications where comprehensibility is an issue.

2 Regression via Classification

The regression problem is usually seen as the task of predicting a (more or less) exact numeric target value for previously unseen examples. Thus, the target attribute is not specified by discrete symbols, but by many distinct values from a fixed range. Specialized algorithms have been invented for this task, but the question arises whether algorithms capable of doing classification (i.e. predicting discrete symbols) couldn't possibly be applied here as well. The basic idea would be to discretize the target attribute by splitting its range into some pre-defined number of intervals, and learn to classify examples with a classification learner (like C4.5). Then, instead of just predicting the class label of an unseen example, an exact value from the according interval can be predicted (e.g. the *mean* or the *median*). [12] is one of the first detailed studies in this direction.

Unfortunately it seems that there is a theoretical limit as to what *can* be achieved by this approach to the regression problem (in terms of a lowering of the summed errors): Increasing the number of intervals usually means that the deviations *within* the intervals become smaller, but also that the accuracy of the class predictions decreases. Decreasing the number of intervals on the other hand usually goes along with higher intra-interval deviations.

What shall be shown in this paper is that by using RUDE, a method that is capable of projecting the structure of source attributes onto the continuous target attribute without demanding discreteness (as supervised methods do), we can improve the regression behaviour of the learning step in comparison to unsupervised methods. This improvement can be seen in terms of *absolute deviation* and/or *tree size* (readability).

3 RUDE – Relative Unsupervised Discretization

3.1 Goals

Originally, the algorithm RUDE described in this paper was developed as a strategy for discretizing datasets where a specified target attribute does not exist (like, e.g. when inducing *functional dependencies*) or where the target attribute itself is continuous. RUDE combines aspects of both unsupervised and supervised discretization algorithms. What sets RUDE apart from other supervised discretization algorithms is that it is not constrained to using information from only one discrete (class) attribute when deciding how to split the attribute in

question (see section 3.2). “RUDE” is actually short for **R**elative **U**nsupervised **D**iscr**E**tization, which quite exactly summarizes what this procedure does: The procedure may be called unsupervised in the sense that there is no need to specify one particular class attribute beforehand, nonetheless the split points are not constructed independently of the “other” attributes (hence “relative”).

3.2 RUDE – The Top-Level

The basic idea when discretizing a given attribute (the *target*) is to use information about the value distribution of all attributes other than the target (the *source attributes*). Intuitively, a “good” discretization would be one that has split points that correlate strongly with changes in the value distributions of the source attributes. The process that tries to accomplish this (the central component of RUDE) is called *structure projection*. Here is the top level of RUDE:

1. **Preprocessing:** Discretize (via some unsupervised method) all source attributes that are continuous (see section 3.3);
2. **Structure Projection:** Project the structure of each source attribute a onto the target attribute t :
 - (a) Filter the dataset by the different values of attribute a .
 - (b) For each such filtering perform a **clustering** procedure on values of t (see section 3.4) and gather the split points thereby created.
3. **Postprocessing:** Merge the split points found.

The time complexity of the RUDE algorithm (discretizing one continuous attribute) is $O(nm \log m)$, with n the number of attributes and m the number of examples. A complete discretization of all continuous attributes can therefore be performed in time $O(n^2 m \log m)$. Please refer to [7] for the proof.

3.3 The Main Step: Structure Projection

The intuition behind the concept of structure projection is best illustrated with an example (see Figure 1). Suppose we are to discretize a target attribute t with a range of, say, $[0..1]$, which happens to be uniformly distributed in our case. The values of t in our learning examples have been drawn along the lowest line in Figure 1. The two lines above indicate the same examples when filtered for the values 1 and 2, respectively, of some particular binary source attribute a . Given the distribution of t , any unsupervised discretizer would return a rather arbitrary segmentation of t that would not reflect the (to us) obvious distribution changes in the source attribute a . The idea of structure projection is to find points where the distribution of the values of a changes drastically, and then to map these “edges” onto the target t . The algorithm we have developed for that purpose was in fact inspired by the concept of edge detection in grey-scale image processing (see section 3.4). The basic discretization algorithm can now be stated in Fig. 2.

RUDE successively maps the “structure” of all source attributes onto the sequence of t ’s values, thereby creating split points only at positions where some

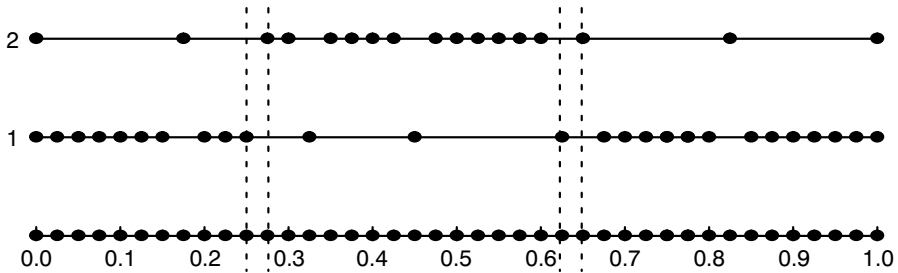


Fig. 1. Structure Projection: An Example

Given:

- a database containing our training examples;
- a set of (possibly continuous) source attributes $a_1, \dots, a_n$
- information on what attribute should be discretized (the target t);

The algorithm:

1. Sort the database in ascending order according to attribute t .
2. For each attribute a_i with $a_i \neq t$ do the following:
 - (a) If continuous, discretize attribute a_i by *equal width*
 - (b) For each symbolic value (interval) v thereby created do the following:
 - i. Filter the database for value v in attribute a_i .
 - ii. Perform *clustering* on the corresponding values of t in the filtered database.
 - iii. Gather the split points thereby created in a *split point list* for attribute t .

Fig. 2. RUDE – The basic discretization algorithm

significant distribution changes occur in some of the a_i . For pre-discretizing continuous source attributes in item 2(a) above, we have decided to use *equal-width discretization*, because it not only provides a most efficient (linear) method, but also has some desirable statistical properties (see [7] for details).

The critical component in all this is the clustering algorithm that groups values of the target t into segments that are characterized by more or less common values of some source attribute a_i . Such segments correspond to relatively densely populated areas in the range of t when filtered for some value of a_i (see Figure 1). Thus, an essential property of this algorithm must be that it tightly delimits such dense areas in a given sequence of values.

3.4 A Characterizing Clustering Algorithm

The clustering algorithm we developed for this purpose has its roots in the concept of *edge detection* in grayscale image processing ([8]). The central problem in edge detection is to find boundaries between areas of markedly different degrees of darkness. Typical edge detection algorithms amplify the *contrast* where it exceeds a certain threshold. The analogy to our clustering problem is fairly obvious and has led us to develop an algorithm that basically works by opening

Given:

- A split point list $s_1, s_2, \dots$
- A merging parameter (minimal difference s).

The algorithm:

1. Sort the sequence of split points in ascending order.
2. Run through the sequence until you find split points s_i and s_{i+1} with $s_{i+1} - s_i \leq s$.
3. Starting at $i+1$ run through the sequence until you find two split points s_j and s_{j+1} with $s_{j+1} - s_j > s$.
4. Calculate the *median* m of $[s_i, \dots, s_j]$.
 - If $s_j - s_i \leq s$ **merge** all split points in $[s_i, \dots, s_j]$ to m .
 - If $s_j - s_i > s$ **triple** the set of split points in $[s_i, \dots, s_j]$ to $\{s_i, m, s_j\}$.
5. Start at s_{j+1} and go back to step 2.

Fig. 3. Merging the split points

a “window” of a fixed size around each of the values in an ordered sequence and determining whether this value lies at an “edge”, i.e. whether one half of the window is “rather empty” and the other is “rather full”. The notions of “rather full” and “rather empty” are operationalized by some user-defined parameters. One advantage of the algorithm is that it autonomously determines the appropriate number of clusters/splits, which is in contrast to simpler clustering methods like, e.g., k-means clustering. The details of the algorithm are described in [7].

3.5 Post-processing: Merging the Split Points

Of course, due to the fact that RUDE projects multiple source attributes onto a single target attribute, usually many “similar” split points will be formed during the projections. It is therefore necessary to *merge* the split points in a post-processing phase. Figure 3 shows an algorithm for doing that. At step 3 we have found a subset of split points with successive differences lower than or equal to a certain pre-defined value s . Now, if all these split points lie closer than s (very dense), they are merged down to only one point (the median). If not, the region is characterized by the median and the two outer borders.

4 Experimental Results

Generally, evaluating discretization algorithms is not a straightforward task, as the quality of the discretization *per se* can hardly be measured. Therefore, analogously to [12], we have chosen to apply RUDE to the problem of regression. We measure the *mean average deviation* as well as the *mean tree size* that can be achieved by applying RUDE to a dataset with a continuous target attribute, learning a decision tree via C4.5 [10], and using the median of a predicted interval as the numeric class label for test examples. The results are compared to those achievable by *Equal Width* and *K-Means* discretization with the same classification learner. Table 1 summarizes the databases used for the experiments.

Table 1. The UCI datasets used in the experiments

Dataset	size	attributes	
		continuous	nominal
Abalone	4177	7	1
Auto-mpg	398	4	3
Housing	506	12	1
Machine	209	6	0
Servo	167	0	4

All results were achieved by 10-fold cross-validation. Within each of the 10 runs, a discretization of the target attribute (i.e. a split point list and the according medians) was learned on the training set, these intervals were applied to the test set, and C4.5 was run on these transformed files. We report results in terms of *mean average deviation (MAD)* and *mean tree size*.

For each method, different parameter settings were tried. Table 2 shows selected results for runs with the same number of intervals: The best RUDE run (in terms of MAD) was compared to the values achieved by equal width (EW) and k-means (KM), when set to the same number of intervals.

In table 3, the “best” results achievable by each algorithm are compared. However, simply defining the “best” runs by the lowest MAD value would have resulted in the observation that the deviations achieved by RUDE are nearly always slightly higher than with EW or KM, but the tree sizes are drastically lower! Therefore this figure shows runs with slightly higher deviations than necessary, but much better tree sizes – a good compromise was intended.

Table 2. Selected results from running EW, KM and RUDE on the same datasets, comparing values for the same number of intervals (best RUDE run) against each other. The values in bold print are the best ones (differences are not necessarily significant)

Dataset	EW MAD & Size	KM MAD & Size	RUDE MAD & Size	Intervals
Abalone	1.95 ± 0.04 871.3 ± 35.3	1.94 ± 0.06 1444.9 ± 47.3	1.93 ± 0.08 497.8 ± 408.4	7
Auto-MPG	2.76 ± 0.43 153.2 ± 8.9	2.85 ± 0.36 163.6 ± 5.4	3.47 ± 0.36 129.7 ± 15.3	8
Housing	3.08 ± 0.34 167.6 ± 4.7	3.13 ± 0.30 197.6 ± 13.5	3.32 ± 0.41 138.0 ± 28.6	9
Machine	57.91 ± 22.03 36.4 ± 10.8	61.91 ± 32.52 129.9 ± 16.2	45.59 ± 15.14 86.9 ± 25.0	7
Servo	0.44 ± 0.17 35.0 ± 0.0	0.34 ± 0.13 60.0 ± 4.0	0.39 ± 0.15 62.0 ± 4.0	6

Table 3. Comparing the “best” runs of EW, KM and RUDE

Dataset	EW		KM		RUDE	
	MAD, Size	# Ints.	MAD, Size	# Ints.	MAD, Size	# Ints.
Abalone	2.31 ± 0.05	2	2.10 ± 0.06	2	2.13 ± 0.09	4
	133.6 ± 16.2		259.1 ± 27.7		32.8 ± 58.64	
Auto-MPG	2.83 ± 0.33	6	3.59 ± 0.32	3	3.96 ± 0.34	3
	133.4 ± 6.4		76.4 ± 13.68		51.4 ± 3.32	
Housing	4.27 ± 0.36	4	4.00 ± 0.33	3	4.07 ± 0.34	4
	66.8 ± 9.04		71.0 ± 6.8		65.8 ± 6.96	
Machine	49.80 ± 13.10	4	39.63 ± 7.80	4	51.49 ± 16.25	5
	13.6 ± 9.76		59.5 ± 14.8		32.8 ± 8.72	
Servo	0.44 ± 0.16	2	0.44 ± 0.16	2	0.44 ± 0.16	3
	20.0 ± 0.0		20.5 ± 0.9		21.5 ± 2.1	

As can be seen, the mean average deviation achieved by RUDE is usually slightly higher than with the other two methods (or about equal). The reason for this could be that there is a theoretical limit as to what *can* be achieved by applying classification methods to regression problems; equal width usually achieves low numeric error, because the medians are quite equally distributed, even though the classification accuracy might not be very high (resulting in a higher tree size). With RUDE, on the other hand, tree size usually decreases significantly. This effect is apparently more visible the larger the dataset is.

In summary, RUDE seems to be able to tune the interval boundaries better than the two unsupervised methods compared here. With the same number of intervals, RUDE creates better split points (with regard to lower tree sizes and thus better understandability), even compared to k-means. Comparing the lowest MAD achieved (not caring about the number of intervals), RUDE admittedly loses. Nonetheless, even in these cases, RUDE can improve readability.

5 Discussion

What we have presented is a new method for discretizing continuous attributes by using information about the “structure” of multiple source attributes. Preliminary experimental results show that in a regression-by-classification setting, this algorithm does not improve the summed numerical error of the predictions, but can lower the tree sizes substantially, especially in large databases.

One of the main problems with the current system is that the user-specified parameters still need to be fine-tuned when dealing with a new dataset. Up to now there is no good standard set of parameter settings that works well every time. Also, unfortunately some of the parameters represent absolute values; the problem of defining relative threshold measures (like percentages) is also a current research topic.

RUDE was originally designed with *association rules* and *functional dependencies* in mind. Algorithms for inducing the latter type of knowledge can, by

definition, only work on nominal data, which makes them unsuitable for numerical databases. We are currently testing the efficacy of RUDE in this setting. Devising quantitative measures of success in such applications is a non-trivial problem, which we are currently trying to solve.

Acknowledgments

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF. We would like to thank Johannes Fürnkranz and Bernhard Pfahringer for invaluable suggestions.

References

1. Dougherty J., Kohavi R., Sahami M.: Supervised and Unsupervised Discretization of Continuous Features. In *Proceedings of the 12th International Conference on Machine Learning (ML95)*, Morgan Kaufmann, San Francisco, CA, 1995. 246
2. Fayyad U., Irani K.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, Morgan Kaufmann, San Francisco, 1993. 246
3. Friedman N., Goldszmidt M., Lee T.J.: Bayesian Network Classification with Continuous Attributes: Getting the Best of Both Discretization and Parametric Fitting. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, Morgan Kaufmann, San Francisco, CA, pp.179-187, 1998. 246
4. Jun B.H., Kim C.S., Song H.Y., Kim J.: A New Criterion in Selection and Discretization of Attributes for the Generation of Decision Trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(12), 1371- 1375, 1997.
5. Kerber R.: ChiMerge: Discretization of Numeric Attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, AAAI Press, Menlo Park, 1992. 246
6. Kohavi R., Sahami M.: Error-Based and Entropy-Based Discretization of Continuous Features. In *KDD-96: Proceedings 2nd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, pp.114-119, 1996. 246
7. Ludl M.-C.: *Relative Unsupervised Discretisation of Continuous Attributes*. Master's thesis, Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, 2000 (forthcoming). 248, 249, 250
8. Pavlidis T.: *Algorithms for Graphics and Image Processing*, Computer Science Press, Inc., Rockville, Maryland USA, 1982. 249
9. Pfahringer B.: Compression-Based Discretization of Continuous Attributes, in *Proceedings of the 12th International Conference on Machine Learning (ML95)*, Morgan Kaufmann, San Francisco, CA, 1995. 246
10. Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993. 250
11. Richeldi M., Rossotto M.: Class-Driven Statistical Discretization of Continuous Attributes. In *Machine Learning: ECML-95*, Springer, Berlin, pp.335-338, 1995. 246
12. Torgo, M., Gama, J.: Regression Using Classification Algorithms. *Intelligent Data Analysis* 1, 275-292, 1997. 246, 247, 250

13. Wang K., Goh H.C.: Minimum Splits Based Discretization for Continuous Features.
In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Morgan Kaufmann, San Francisco, CA, pp.942-951, 1997. 246

Metric-Based Inductive Learning Using Semantic Height Functions

Zdravko Markov¹ and Ivo Marinchev²

¹ Department of Computer Science, Central Connecticut State University
1615 Stanley Street, New Britain, CT 06050, U.S.A.

`markovz@ccsu.edu`

² Faculty of Mathematics and Informatics, University of Sofia
5 James Bouchier Str., 1164 Sofia, Bulgaria

`ivo@fmi.uni-sofia.bg`

Abstract. In the present paper we propose a consistent way to integrate syntactical least general generalizations (lgg's) with semantic evaluation of the hypotheses. For this purpose we use two different relations on the hypothesis space – a constructive one, used to generate lgg's and a semantic one giving the coverage-based evaluation of the lgg. These two relations jointly implement a *semantic distance measure*. The formal background for this is a height-based definition of a semi-distance in a join semi-lattice. We use some basic results from lattice theory and introduce a family of language independent coverage-based height functions. The theoretical results are illustrated by examples of solving some basic inductive learning tasks.

1 Introduction

Inductive learning addresses mainly classification tasks where a series of training examples (instances) are supplied to the learning system and the latter builds an intensional or extensional representation of the examples (hypothesis), or directly uses them for prediction (classification of unseen examples). Generally two basic approaches to inductive learning are used. The first one is based mainly on generalization/specialization or similarity-based techniques. This approach includes two types of systems – *inductive learning from examples* and *conceptual clustering*. They both generate inductive hypotheses made by abstractions (generalizations) from specific examples and differ in the way examples are presented to the system (whether or not they are pre-classified). The basic techniques used within the second approach are various kinds of *distances (metrics)* over the example space which are used to classify directly new examples (by similarity to the existing ones) or group the examples into clusters.

There exists a natural way to integrate consistently the generalization-based and metric-based approaches. The basic idea is to estimate the similarity between two objects in a hierarchical structure by the distance to their closest common parent. This idea is formally studied within the lattice theory. In ML

this is the well known *least general generalization* (*lgg*) which given two hypotheses builds their most specific common generalization. The existence of an *lgg* in a hypothesis space (a partially ordered set) directly implies that this space is a semi-lattice (where the *lgg* plays the role of infimum). Consequently some algebraic notions as finiteness, modularity, metrics etc. can be used to investigate the properties of the hypothesis space. *Lgg*'s exist for most of the languages commonly used in ML. However all practically applicable (i.e. computable) *lgg*'s are based on *syntactical* ordering relations. A relation over hypotheses is syntactical if it does not account for the background knowledge and for the coverage of positive/negative examples. For example dropping condition for nominal attributes, instance relation for atomic formulae and θ -subsumption for clauses are all syntactical relations. On the other hand the evaluation of the hypotheses produced by an *lgg* operator is based on their coverage of positive/negative examples with respect to the background knowledge, i.e. it is based on *semantic relations* (in the sense of the inductive task). This discrepancy is a source of many problems in ML, where overgeneralization is the most difficult one.

In the present paper we propose a consistent way to integrate syntactical *lggs* with semantic evaluation of the hypotheses. For this purpose we use two different relations on the hypothesis space – a constructive one, used to generate *lgg*'s and a semantic one giving the coverage-based evaluation of the *lgg*. These two relations jointly implement a *semantic distance measure*. The formal background for this is a height-based definition of a semi-distance in a join semi-lattice. We use some basic results from lattice theory and introduce a language independent coverage-based height function. We also define the necessary conditions for two relations to form a correct height function. The paper introduces a bottom-up inductive learning algorithm based on the new semantic semi-distance which is used to illustrate the applicability of the theoretical results.

The paper is organized as follows. The next section introduces the basic algebraic notions used throughout the paper. Section 3 introduces the new a height-based semi-distance. Section 4 presents an algorithm for building lattice structures and shows some experiments with this algorithm. Section 5 contains concluding remarks and directions for future work.

2 Preliminaries

In this section we introduce a height-based distance measure on a join semi-lattice following an approach similar to those described in [1] and [5] (for a survey of metrics on partially ordered sets see [2]).

Definition 1 (Semi-distance, Quasi-metric). *A semi-distance (quasi-metric) is a mapping $d : O \times O \rightarrow \mathbb{R}$ on a set of objects O with the following properties ($a, b, c \in O$):*

1. $d(a, a) = 0$ and $d(a, b) \geq 0$.
2. $d(a, b) = d(b, a)$ (symmetry).
3. $d(a, b) \leq d(a, c) + d(c, b)$ (triangle inequality).

Definition 2 (Order preserving semi-distance). A semi-distance $d : O \times O \rightarrow \mathbb{R}$ on a partially ordered set $(O, \preceq)$ is order preserving iff $\forall a, b, c \in O : a \preceq b \preceq c \Rightarrow d(a, b) \leq d(a, c)$ and $d(b, c) \leq d(a, c)$

Definition 3 (Join/Meet semi-lattice). A join/meet semi-lattice is a partially ordered set $(A, \preceq)$ in which every two elements $a, b \in A$ have an infimum/supremum.

Definition 4 (Size). Let $(A, \preceq)$ be a join semi-lattice. A mapping $s : A \times A \rightarrow \mathbb{R}$ is called a size function if it satisfies the following properties:

- S1. $s(a, b) \geq 0, \forall a, b \in A$ and $a \preceq b$.
- S2. $s(a, a) = 0, \forall a \in A$.
- S3. $\forall a, b, c \in A : a \preceq c$ and $c \preceq b \Rightarrow s(a, b) \leq s(a, c) + s(c, b)$.
- S4. $\forall a, b, c \in A : a \preceq c$ and $c \preceq b \Rightarrow s(c, b) \leq s(a, b)$.
- S5. $\forall a, b \in A$. Let $c = \inf\{a, b\}$. For any $d \in A : a \preceq d$ and $b \preceq d \Rightarrow s(c, a) + s(c, b) \leq s(a, d) + s(b, d)$.

Theorem 1. Let $(A, \preceq)$ be a join semi-lattice and s - a size function. Let $d(a, b) = s(\inf\{a, b\}, a) + s(\inf\{a, b\}, b)$. Then d is a semi-distance on $(A, \preceq)$.

Proof. 1. d is non-negative by S1.

2. $d(a, a) = s(\inf\{a, a\}, a) + s(\inf\{a, a\}, a) = s(a, a) + s(a, a) = 0$.

3. d is symmetric by definition.

4. We will show that $d(a_1, a_2) \leq d(a_1, a_3) + d(a_3, a_2)$. Let $c = \inf\{a_1, a_2\}$, $b_1 = \inf\{a_1, a_3\}$, $b_2 = \inf\{a_2, a_3\}$, $d = \inf\{b_1, b_2\}$. By S4 and S3 we have $s(c, a_1) \leq s(d, a_1) \leq s(d, b_1) + s(b_1, a_1)$. And by analogy $s(c, a_2) \leq s(d, b_2) + s(b_2, a_2)$. Then $d(a_1, a_2) = s(c, a_1) + s(c, a_2) \leq s(d, b_1) + s(b_1, a_1) + s(d, b_2) + s(b_2, a_2) \leq s(b_1, a_1) + s(b_1, a_3) + s(b_2, a_3) + s(b_2, a_2) = d(a_1, a_3) + d(a_3, a_2)$

A size function can be defined by using the so called *height functions*. The approach of height functions has the advantage that it is based on estimating the object itself rather than its relations to other objects.

Definition 5 (Height). The function h is called height of the elements of a partially ordered set $(A, \preceq)$ if it satisfies the following two properties:

- 1. For every $a, b \in A$ if $a \preceq b$ then $h(a) \leq h(b)$ (isotone).
- 2. For every $a, b \in A$ if $c = \inf\{a, b\}$ and $d \in A$ such that $a \preceq d$ and $b \preceq d$ then $h(a) + h(b) \leq h(c) + h(d)$.

Theorem 2. Let $(A, \preceq)$ be a join semi-lattice and h be a height function. Let $s(a, b) = h(b) - h(a), \forall a \preceq b \in A$. Then s is a size function on $(A, \preceq)$.

Proof. 1. $s(a, b) = h(b) - h(a) \geq 0$ by H1.

2. $s(a, a) = h(a) - h(a) = 0$.

3. Let $a, b, c \in A : a \preceq c, c \preceq b$. Then $s(a, b) = h(b) - h(a) = (h(b) - h(c)) + (h(c) - h(a)) = s(a, c) + s(c, b)$.

4. Let $a, b, c \in A : a \preceq c, c \preceq b$. Then $s(c, b) \leq s(c, b) + s(a, c) = s(a, b)$ by 3.
5. Let $c = \inf\{a, b\}$ and $d \in A : a \preceq d$ and $b \preceq d$. Then $s(c, a) + s(c, b) = (h(a) - h(c)) + (h(b) - h(c)) = h(a) + h(b) - 2h(c) = 2(h(a) + h(b)) - h(a) - h(b) - 2h(c) \leq 2(h(c) + h(d)) - h(a) - h(b) - 2h(c) = (h(d) - h(a)) + (h(d) - h(b)) = s(a, d) + s(b, d)$

Corollary 1. *Let $(A, \preceq)$ be a join semi-lattice and h be a height function. Then the function $d(a, b) = h(a) + h(b) - 2h(\inf\{a, b\})$, $\forall a, b \in A$ is a semi-distance on $(A, \preceq)$.*

3 Semantic Semi-distance on Join Semi-lattices

Let A be a set of objects and let $\preceq_1$ and $\preceq_2$ be two binary relations in A , where $\preceq_1$ is a partial order and $(A, \preceq_1)$ is a join semi-lattice. Let also GA be the set of all maximal elements of A w.r.t. $\preceq_1$, i.e. $GA = \{a \mid a \in A \text{ and } \neg \exists b \in A : a \preceq_1 b\}$. Hereafter we call the members of GA *ground elements* (by analogy to ground terms in first order logic). For every $a \in A$ we denote by S_a the *ground coverage* of a w.r.t. $\preceq_2$, i.e. $S_a = \{b \mid b \in GA \text{ and } a \preceq_2 b\}$.

The ground coverage S_a can be considered as a definition of the semantics of a . Therefore we call $\preceq_2$ a *semantic relation* by analogy to the Herbrand interpretation in first order logic that is used to define the semantics of a given term. The other relation involved, $\preceq_1$ is called *constructive (or syntactic) relation* because it is used to build the lattice from a given set of ground elements GA .

The basic idea of our approach is to use these two relations, $\preceq_1$ and $\preceq_2$ to define the semi-distance. According to Corollary 1 we use the syntactic relation $\preceq_1$ to find the infimum and the semantic relation $\preceq_2$ to define the height function h . The advantage of this approach is that in many cases there exists a proper semantic relation however it is intractable, computationally expensive or even not a partial order, which makes impossible its use as a constructive relation too (an example of such a relation is logical implication). Then we can use another, simpler relation as a constructive one (to find the infimum) and still make use of the semantic relation (to define the height function).

Not any two relations however can be used for this purpose. We will show that in order to define a correct semi-distance the two relations $\preceq_1$ and $\preceq_2$ must satisfy the following properties, which we call *coupling*.

Definition 6. $\preceq_2$ is coupled with $\preceq_1$ if both conditions apply:

1. For every $a, b \in A$ such that $a \preceq_1 b$ either $|S_a| \geq |S_b|$ or $|S_a| \leq |S_b|$ must hold. As the other case is analogous without loss of generality we can assume that $\forall a, b \in A, a \preceq_1 b \Rightarrow |S_a| \geq |S_b|$.
2. $\forall a, b \in A : c = \inf\{a, b\}$ and $\exists d = \sup\{a, b\}$ one of the following must hold:
 - C1. $|S_d| < |S_a|$ and $|S_d| < |S_b|$
 - C2. $|S_d| = |S_a|$ and $|S_d| = |S_b|$

$$C3. |S_d| = |S_b| \text{ and } |S_c| = |S_a|$$

Corollary 2. *Every partial order relation is coupled with itself.*

Theorem 3. *Let A be a set of objects and let $\preceq_2$ and $\preceq_1$ be two binary relations in A such that $\preceq_2$ is coupled with $\preceq_1$. Then there exists a family of height functions $h(a) = x^{-|S_a|}$, where $a \in A$, $x \in \mathbb{R}$ and $x \geq 2$.*

Proof. 1. Let $a, b \in A$, such that $a \preceq_1 b$. Then by the definition of coupling $|S_a| \geq |S_b|$ and hence $h(a) \leq h(b)$.
 2. Let $a, b \in A : c = \inf\{a, b\}$ and $\exists d = \sup\{a, b\}$.
 (a) Assume that $C1$ is true. Then $|S_d| < |S_a|$ and $|S_d| < |S_b| \Rightarrow |S_a| \geq |S_d| + 1$ and $|S_b| \geq |S_d| + 1 \Rightarrow -|S_a| \leq -|S_d| - 1$ and $-|S_b| \leq -|S_d| - 1$. Hence $h(a) + h(b) = x^{-|S_a|} + x^{-|S_b|} \leq x^{-|S_d|-1} + x^{-|S_d|-1} = 2x^{-|S_d|-1} \leq x \cdot x^{-|S_d|-1} = x^{-|S_d|} = h(d) \leq h(c) + h(d)$.
 (b) Assume that $C2$ is true. Then $|S_d| = |S_a|$ and $|S_c| = |S_b|$. Hence $h(a) + h(b) = h(c) + h(d)$.
 (c) Assume that $C3$ is true. Then $|S_d| = |S_b|$ and $|S_c| = |S_a|$. Hence $h(a) + h(b) = h(c) + h(d)$.

4 Experiments

To illustrate the theoretical results we use an algorithm that builds a join semi-lattice G , given a set of examples GA (the set of all maximal elements of G). The algorithm hereafter referred to as *MBI* (*Metric-based Bottom-up Induction*) is as follows:

1. Initialization: $G = GA$, $C = GA$;
2. If $|C| = 1$ then exit;
3. $T = \{h | h = \lgg(a_1, a_2) : a_1, a_2 \in C \text{ and } d(a_1, a_2) = \min\{d(b, c) | b, c \in C\}\}$;
4. $DC = \{h | h \in C \text{ and } \exists h_{min} \in T : h_{min} \preceq_2 h\}$;
5. $C = C \setminus DC$;
6. $G = G \cup T$, $C = C \cup T$, go to step 2.

There is a possible modification of this algorithm. In Step 3 instead of all, only one minimal element h from T can be used. With this modification the algorithm has a polynomial time complexity $O(n^3)$. A disadvantage of this modification is that some useful generalizations can be missed. Therefore in the practical implementations we augment the algorithm with another distance or heuristic measure used to select one of all minimal elements of T which possibly leads to the most useful generalizations.

Further in this section we discuss some experiments with the MBI algorithm with two different representation languages – atomic formulae and Horn clauses.

4.1 Atomic Formulae

The algebraic properties of the language of first order atomic formulae are studied by Reynolds in [8], where he shows that the set of atoms with the same functors and arity form a *join semi-lattice* (or *complete lattice* when the language is augmented by adding a 'universal atom' and a 'null atom'). In this framework we use $\preceq_1 = \preceq_2 = \theta$ -subsumption and by Corollary 2 we have that θ -subsumption is coupled with itself.

Figure 1 shows the top portion of the lattice G built by the algorithm, where GA consists of the 61 *positive* examples of the well-known MONK1 [9] database (the training sample) represented as atoms. Note that the produced lattice can be used both for concept learning (it contains the target hypothesis $\text{monk}(A, A, \_, \_, \_, \_)$ or $\text{monk}(\_, \_, \_, \_, \text{red}, \_)$) and for conceptual clustering since the classifications of the examples are not used (the negative examples are skipped).

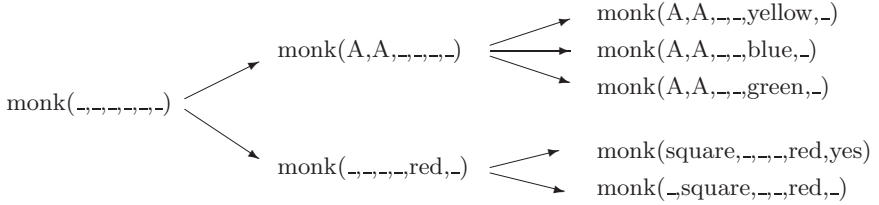


Fig. 1. Hypotheses for the MONK1 problem built by the MBI algorithm

In more complex domains however the standard version of the algorithm performs poorly with small sets of randomly selected examples. In these cases we use the augmented version of the algorithm with a syntactic distance measure to choose one element of T in Step 3. In this way we avoid the random choice and allow "cautious" generalizations only. Further heuristics can be used for this purpose, especially in the case of background knowledge.

4.2 Horn Clauses

Within the language of Horn clauses the MBI algorithm can be used with the θ -subsumption-based lgg (the constructive relation $\preceq_1$) and *logical implication* for the semantic relation $\preceq_2$. Under θ -subsumption as partial order the set of Horn clauses with same head predicates forms a semi-lattice. Furthermore, it can be shown that logical implication is coupled with θ -subsumption which makes the use of our algorithm well founded. Figure 2 shows the complete lattice build by the algorithm with 10 instances of the *member* predicate.

A major problem in bottom-up algorithms dealing with lgg_θ of clauses is the *clause reduction*, because although finite the length of the lgg_θ of n clauses can grow exponentially with n . Some well-known techniques of avoiding this problem

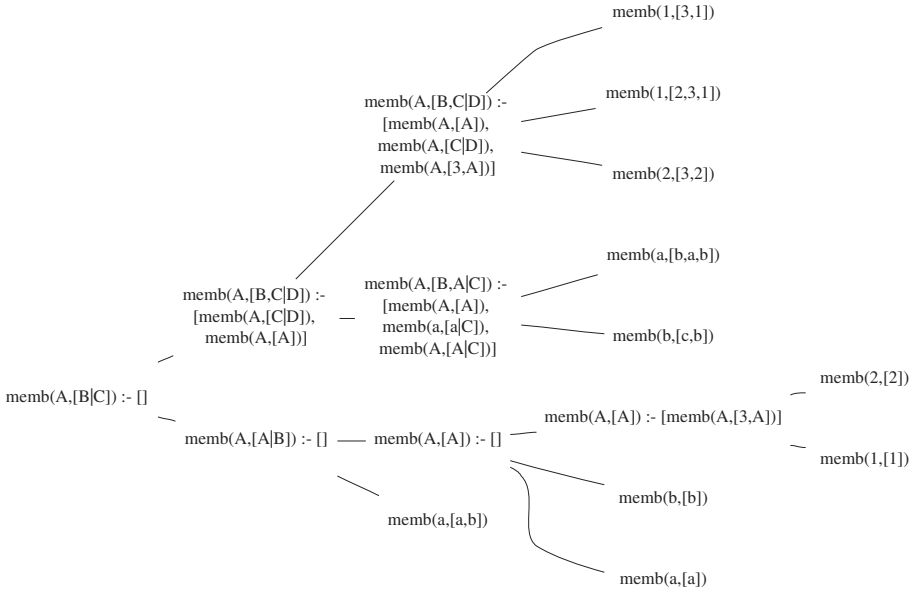


Fig. 2. ILP hypotheses for the instances of the *member* predicate

are discussed in [3]. By placing certain restrictions on the hypothesis language the number of literals in the *lgg* clause can be limited by a polynomial function independent on n . Currently we use *ij-determinate* clauses in our experiments (actually 22-determinate).

5 Conclusion

The algebraic approach to inductive learning is a very natural way to study the generalization and specialization hierarchies. These hierarchies represent hypothesis spaces which in most cases are partially ordered sets under some generality ordering. In most cases however the orderings used are based on *syntactical* relations, which do not account for the background knowledge and for the coverage of positive/negative examples. We propose an approach that explores naturally the semantic ordering over the hypotheses. This is because although based on syntactic lgg it uses a semantic evaluation function (the height function) for the hypotheses. Furthermore this is implemented in a consistent way through a height-based semi-distance defined on the hypothesis space.

As in fact we define a new distance measure our approach can be also compared to other metric-based approaches in ML. Most of them are based on attribute-value (or feature-value) languages. Consequently most of the similarity measures used stem from well known distances in feature spaces (e.g. Euclidean distance, Hamming distance etc.) and vary basically in the way the weights are computed. Recently a lot of attention has been paid to studying distance mea-

asures in first order languages. The basic idea is to apply the highly successful instance based algorithms to relational data using first order logic descriptions. Various approaches have been proposed in this area. Some of the most recent ones are [1,4,6,7]. These approaches as well as most of the others define a simple metric on atoms and then extend it to sets of atoms (clauses or models) using the Hausdorff metric or other similarity functions. Because of the complexity of the functions involved and the problems with the computability of the models these approaches are usually computationally hard. Compared to the other approaches our approach has two basic advantages. First, it is language independent, i.e. it can be applied both within propositional (attribute-value) languages and within first order languages and second, it allows consistent integration of generalization operators with a semantic distance measure.

We consider the following directions for future work. Firstly, particular attention should be paid to the clause reduction problem when using the language of Horn clauses. Other lgg operators, not based on θ -subsumption should be considered too.

The practical learning data often involve numeric attributes. In this respect proper relations, lgg's and covering functions should be investigated in order to extend the approach for handling numeric data.

Though the algorithm is well founded it still uses heuristics. This is because building the complete lattice is exponential and we avoid this by employing a hill-climbing strategy. It is based on additional distance measures or heuristics used to select the best lgg among all minimal ones (Step 3 of the algorithm). Obviously this leads to incompleteness. Therefore other strategies should be investigated or perhaps the semantic relation should be refined to incorporate these additional heuristics.

Finally, more experimental work needs to be done to investigate the behavior of the algorithm in noisy domains.

References

1. A. Hutchinson. Metrics on terms and clauses. In M. van Someren and G. Widmer, editors, *Machine Learning: ECML-97*, volume 1224 of *Lecture Notes in Artificial Intelligence*, pages 138–145. Springer-Verlag, 1997. 255, 261
2. B. Monjardet. Metrics on partially ordered sets – a survey. *Discrete Mathematics*, 35:173–184, 1981. 255
3. S. Muggleton. Inductive logic programming. In S. Muggleton, editor, *Inductive Logic Programming*, pages 3–28. Academic Press, 1992. 260
4. S.-H. Nienhuys-Cheng. Distance between herbrand interpretations: a measure for approximations to a target concept. Technical Report EUR-FEW-CS-97-05, Erasmus University, 1997. 261
5. N. Pelov. Metrics on terms and clauses and application in inductive machine learning. Master's thesis, University of Sofia, 1997. 255
6. J. Ramon, M. Bruynooghe, and W. V. Laer. Distance measure between atoms. Technical Report CW 264, Katholieke Universiteit Leuven, 1998. 261

7. J. Ramon, M. Bruynooghe, and W. V. Laer. A framework for defining a distance between first-order logic objects. Technical Report CW 263, Katholieke Universiteit Leuven, 1998. 261
8. J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, 5:135–152, 1970. 259
9. S. B. Thrun et al. The MONK's problems - a performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, Dec. 1991. 259

Error Analysis of Automatic Speech Recognition Using Principal Direction Divisive Partitioning*

David McKoskey and Daniel Boley

Department of Computer Science and Engineering
University of Minnesota, Minneapolis, 55455, USA
mckoskey@cs.umn.edu

Abstract. This paper describes an experiment performed using the Principal Direction Divisive Partitioning algorithm (Boley, 1998) in order to extract linguistic word error regularities from several sets of medical dictation data. For each of six physicians, two hundred finished medical dictations aligned with their corresponding automatic speech recognition output were clustered and the results analyzed for linguistic regularities between and within clusters. Sparsity measures indicated a good fit between the algorithm and the input data. Linguistic analysis of the output clusters showed evidence of systematic word recognition error for short words, function words, words with destressed vowels, and phonological confusion errors due to telephony (recording) bandwidth interference. No qualitatively significant distinctions between clusters could be made by examining word errors alone, but the results confirmed several informally held hypotheses and suggested several avenues of further investigation, such as the examination of word error contexts.

1 Introduction

Industrial grade speech recognition has made numerous advances in recent years, especially in corpus based implementations. Modern recognition software such as the application used for this study, often employs a sophisticated combination of techniques for matching speech utterances with their most likely or most desirable text representation (e.g. Hidden Markov modes, rule-based post-recognition processors, partial parsers, etc.). Under ideal conditions, these models enjoy a combined recognition accuracy that approaches 100%. However, word errors due to the misrecognition of an utterance are still not very well understood. Many simple factors influence word recognition accuracy, such as model parameters (e.g. language model scaling factors, word insertion penalties, etc.), speech fluency or disfluency, and items missing from the recognition model's vocabulary. Other factors are more complex, such as the influence of vocal prosody, or vowel devoicing.

Tuning these recognition tools requires extensive analysis, experimentation, and testing. One useful technique for analyzing word errors is linguistic analysis,

* This work was partially supported by NSF grant IIS-9811229.

in which one inspects the available data in search of word error exemplars that adequately represent the more general case. Filled pauses ("um" or "ah"), for example, have been successfully modeled using this technique, and have been shown to "follow a systematic distribution and well defined functions" [4]. As a result recognition accuracy for medical dictation is enhanced by representing the frequency of filled pauses in the recognition model's training data [5]. Unfortunately, other word errors, such as words mistakenly recognized as filled pauses (e.g. "um" may be mistakenly recognized as "thumb" or "arm") [4] are much more difficult to analyze because of their sparsity. In cases where word errors are sparse, error detection by inspection, while still the most accurate of any technique, becomes much more arduous and/or costly.

As part of the Web ACE Project [1], the Principle Direction Divisive Partitioning (PDDP) algorithm was originally designed to classify large collections of documents gleaned from the World Wide Web by clustering them on word frequency. Each document is encoded as a column vector of word counts for all words in the document set, and the document vectors combined into a single matrix. The clustering process recursively splits the matrix and organizes the resulting clusters into a binary tree.

The clustering process consists of four steps:

1. Assign the input matrix as the initial cluster and root of the output PDDP tree. For the initial iteration, the root node is also the only leaf node.
2. Calculate the *scatter value* for all leaf nodes in the PDDP tree and select the node with the largest scatter value.
3. For each document $\mathbf{d}$ in the selected cluster $\mathbf{C}$ containing $\mathbf{k}$ documents, assign $\mathbf{d}$ to the left or right child of $\mathbf{C}$ according to the sign of the linear discrimination function $g_{\mathbf{C}}(\mathbf{d}) = \mathbf{u}_{\mathbf{C}}^T(\mathbf{d} - \mathbf{w}_{\mathbf{C}}) = \sum_{i=1}^n u_i(d_i - w_i)$ where $\mathbf{w}_{\mathbf{C}}$ is the centroid of the current cluster and $\mathbf{u}_{\mathbf{C}}$ is the direction of maximal variance, or principle direction of $\mathbf{C}$. If $g_{\mathbf{d}} \leq 0$, then place $\mathbf{d}$ into the new left child node of $\mathbf{C}$, otherwise, place $\mathbf{d}$ into the new right child of $\mathbf{C}$.
4. Repeat from step 2

The vector $\mathbf{w}_{\mathbf{C}} \stackrel{\text{def}}{=} \frac{1}{k} \sum_j \mathbf{d}_j$ is the mean or centroid of node $\mathbf{C}$. The *scatter value* used for this study is simply the sum of all squared distances from each document $\mathbf{d}$ to the cluster centroid $\mathbf{w}$, though any other suitable criterion may be used as well. The principle direction $\mathbf{u}_{\mathbf{C}}$ corresponds to the largest eigenvalue of the sample covariance matrix for the cluster $\mathbf{C}$. This calculation is the costliest portion of the algorithm, but can be performed quickly with a Lanczos-based singular value solver. The splitting process repeats until either the maximum scatter value of any leaf node is less than the scatter of all current leaf node centroids (a stop test), or until a desired total number of leaf nodes has been reached [2].

Two strengths of the PDDP algorithm include its competitiveness with respect to cluster quality and run time. Previous analysis indicates that PDDP run time scales linearly with respect to the density of the input data matrix, not its size [2]. Studies comparing entropy measures between PDDP and other

clustering methods (such as Hypergraph or LSI) indicate that PDDP exhibits competitive performance on cluster entropy ("cluster quality") measures [2]. For these reasons, the PDDP algorithm was selected to cluster several sets of medical dictation data, clustering on the frequency of word errors in each dictation document. We had no solid hypotheses about what sort(s) of results the clustering would reveal, but hoped the cluster trees would:

- (a) reveal any linguistic regularities in the word errors of each cluster, and
- (b) indicate any relationships between specific word errors and the physician or physicians that most often make(s) them

Results from the mining process would be used to further refine the acoustic and/or language models required by the recognition software (used for this study and elsewhere), and to provide new parsing rules for error correction during post-recognition processing.

2 Data Characteristics and Processing

Modern medical practice typically includes document dictation for the sake of expediency. For example, a doctor dictates his or her patient chart notes into a recording device, and the audio is replayed for a medical transcriptionist (a proficient typist with extensive medical training). The transcriptionist types the dictation, formats the text as chart notes, and submits them to the dictating doctor for inspection. Once the notes are inspected, proofread, and approved, they are inserted into the patient's medical record. Below is an excerpt from a sample finished transcription:

< date > < name >

Subjective: patient is a 51-year-old woman here for evaluation of complaints of sore throat and left ear popping.

Objective: The patient is alert and cooperative and in no acute distress.

External ears and nose are normal.

Assessment: Upper respiratory tract infection.

Plan: treat symptomatically with plenty of fluids, a vaporizer and analgesics as needed.

Linguistic Technologies Inc. (LTI), a medical transcription company based in St. Peter, MN USA, performs recognition on medical dictation audio using an automated speech recognition application. This application is comprised of a Hidden Markov Model decoder, acoustic model, language model, and language dictionary. A rule-based post processor is also used after recognition is complete, to perform several simple parsing tasks, such as formatting numbers (e.g. "one hundred forty over eighty" becomes "140/80"). The output text is then corrected and formatted by a medical transcriptionist for final approval by the dictating

physician. The recognition output, if sufficiently accurate, significantly reduces the medical transcriptionist's workload.

For each of six physicians (henceforth talkers), two hundred finished medical dictations and their corresponding recognition output files were selected. Each set of files was sanitized to remove demographic and time stamp data. Recognition output was conditioned in order to normalize the text (downcase all words, convert numbers and punctuation to text, use standard representations for contractions and abbreviations, etc.). Normalization also included substituting tokens (called TT-words) for common words or phrases (e.g. "TT_nad" is substituted for "no acute distress"), words that require capitalization (e.g. proper names) or words that predictably required specific punctuation marks (e.g. "TT_yearold" was substituted for "year-old"). Finished dictations were treated using PLAB, a proprietary algorithm developed at LTI for inferring transcription of actual speech from formal transcription [5]. This process also included text normalization and rendered the finished dictation into a form that conformed accurately to what was actually said in the original dictation audio. For example, the above finished dictation, after sanitizing and PLAB processing, would look like this:

```
<s> dictating on paragraph TT_scolon patient is a
fifty one TT_yearold woman here for evaluation of ah
complaints of a sore throat and left ear popping period
the TT_patient alert cooperative and in TT_nad period
external ears and nose are normal period TT_acolon upper
respiratory tract infection period paragraph plan colon
will treat this symptomatically with plenty of fluids
ah ah vaporizer and analgesics as needed period </s>
```

The PLAB output and normalized/sanitized recognition output were then aligned word by word. Alignment errors were then divided into three categories:

1. Insertions: words the recognizer inserted that were not in the final dictation (e.g. the software recognized a cough, throat-clearing, or other such utterance as a word).
2. Deletions: words the recognizer deleted by mistake, the reverse scenario of an insertion error.
3. Substitutions: words the recognizer confused (e.g. "he" and "she" are easily confused).

Here is a sample excerpt from an alignment file, illustrating the three types of errors:

TT_ocolon	TT_ocolon	
the	--	DELETION
TT_patient	TT_patient	
is	---	DELETION

alert	alert	
and	---	DELETION
cooperative	cooperative	
and	and	
in	in	
---	no	INSERTION
---	acute	INSERTION
---	distress	INSERTION
TT_nad	since	SUBSTITUTION
external	external	
ears	ears	
...	...	

A matrix containing counts of each word error by document was created for each error category (insertion, deletion, and substitution). Each matrix was then clustered using the PDDP algorithm, which separated the documents in each matrix into clusters by word error. Euclidian Norm scaling was used [1], and the algorithm was halted after fifty clusters were obtained. Histograms were created for each cluster, indicating the number of documents for each talker in that cluster. The cluster's ten most common word errors were also reported, as indicated by the cluster centroid's ten highest values. If the cluster was split, then the ten highest and ten lowest principle direction word errors were also reported, indicating the word errors with the greatest contribution to the split.

3 Results

Sparsity measures for each matrix were taken by simply dividing the number of entries greater than zero by the total size of the input matrix. These measures indicated that all input matrices were between 0.15% and 0.72% fill. This is very sparse, which showed that the data and algorithm were a good match. Some clusters showed a high frequency of a single talker's documents, but significantly fewer documents from other talkers, as illustrated in Fig. 1). These clusters showed a relationship between the strongly represented talker and the word errors of that document's centroid. (In Figures 1 and 2, each colored column represents a different talker. The y-axis on the left edge of the graph contains a scale of 0 to 200, the maximum number of documents for any talker.)

Other cluster histograms showed a more equal representation among talkers, indicating that word errors reported by the centroid (and in the centroids of other, similar clusters) were of a more global character, as indicated in Fig 2.

3.1 General Characteristics

Most of the words reported at each cluster and at each split were short words and function words. "Short words" are words that contain only one or two syllables, such as "he" or "she" ("longer words" will refer to words of three or more

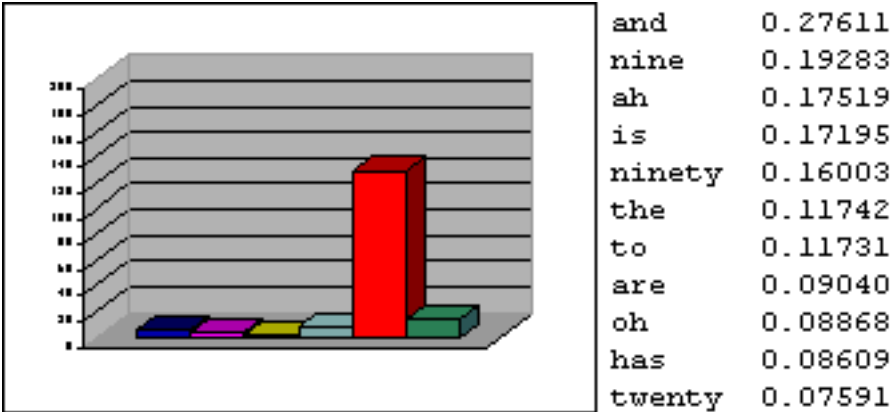


Fig. 1. This cluster and centroid words indicate a strong relationship between a particular talker and particular word errors

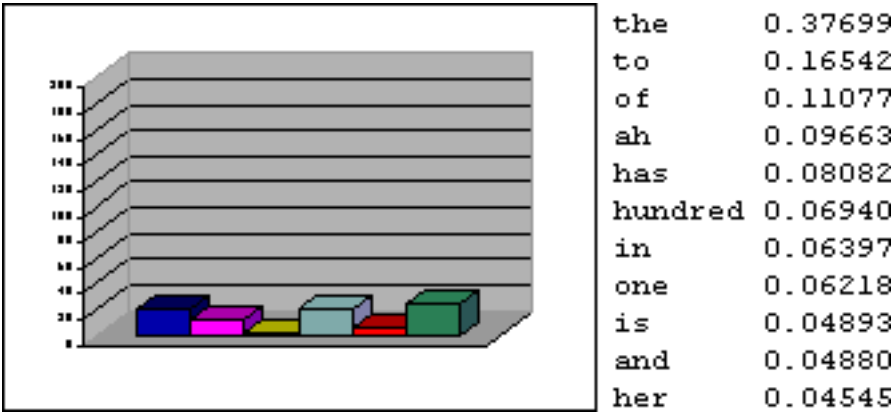


Fig. 2. This cluster suggests that the centroid values are likely global (more ubiquitous) errors

syllables). Function words have little semantic content, but have grammatical function instead, such as determiners ("a", "an", "the"), conjunctions ("and", "or", "but"), copulas ("is", "was", "were") and quantifiers (e.g. numbers). There was a great amount of overlap between these two categories, as most function words are short and many short words are function words.

3.2 Vowel Destressing and Cliticization

Notably, most short word and function word errors contained a destressed vowel. Vowel destressing often co-occurs with cliticization, in which the short word is

"attached" to one of its longer neighbor words. For example, the word "and" in the above excerpt is destressed and cliticized in the phrase "ears and nose": the "a" is destressed and deleted, and the "d" is deleted. The result is an utterance that sounds like "ears anose" or "earsanose" unless spoken very carefully. The recognition software treated most words of this type as noise or filled pauses and discarded them. Vowel destressing and cliticization for short words and function words was common throughout most of the medical dictation examined. While still an unconfirmed hypothesis, we suspect that many destressed words were located near the ends of phrases, a point at which a talker's speech is likely to accelerate.

3.3 Vowel Syncope

Other clusters showed evidence of vowel syncope, in which unstressed vowel sounds in quickly spoken words are deleted. For example, a talker might signal to the medical transcriptionist the end of one paragraph and the beginning of another simply by saying "paragraph". Even in relatively unhurried speech, though, this word was often said quickly, and in the process, the second and "a" in "paragraph" was deleted. The result was an utterance that sounded like "pair-graph". Said even more quickly, the third "a" was also deleted: "pair-grph". As a result, recognition software misidentified the word containing the syncopated vowel(s), making a substitution error (e.g. "oh" for "zero"), or treated the utterance as noise or a filled pause and discarded it, making a deletion or insertion error. Syncope was also ubiquitous throughout the medical dictations examined, though not as common as short word errors.

3.4 Telephony Interference

Cluster centroids and splits also showed some evidence of telephony bandwidth interference. Words (especially short words) that contained voiceless fricative consonants ("f", "th", "s", "sh", etc.), were easily confused, especially in cases where the fricative carries the greatest amount of word information (e.g. "he" versus "she"). These words were also easily mistaken as noise or filled pauses, though short words more frequently than longer words (words of three or more syllables).

4 Discussion / Future Work

Several conclusions can be drawn from the above results. Firstly, word errors involving short, destressed words and function words are ubiquitous throughout the medical dictations examined with the PDDP algorithm. Most often, these words were confused with other function words, brief periods of silence, background noise, or filled pauses. We hypothesized prior to the study that this was the case, but until now, had no way to easily visualize it. One task for subsequent studies would be to cluster the PDDP tree using a centroid stopping

test (described earlier), and re-agglomerate several of the leaf clusters, without regard to which side of the PDDP tree the leaves are situated. This way, clusters that were accidentally fragmented on one dimension during a split along another dimension could be reassembled.

Secondly, number words may or may not cause recognition accuracy problems, because it is known that the first twenty or so words of any dictation contain the patient name, current date, and the name of the dictating physician. These excerpts are rarely, if ever, recognized accurately. Instead, post-recognition processing (simple parsing) seems to more easily rectify problems organizing and correcting word errors involving number words. Future work will more carefully exclude the initial portion of the dictation alignment, so that clustering results will concern only number words found in the body of the dictation text.

Finally, we also noticed that several talkers were split off into their own clusters, such as the cluster shown in Fig. 1. Most often, one or two high frequency word errors were responsible for separating out a specific talker, but more generally, we were unable to discern any qualitatively significant word features that distinguished words in these clusters from word errors elsewhere in the tree. For example, a high frequency of deletion errors involving the word "and" separated out one talker, but all by itself, the word "and" isn't significantly different from the word "an", especially in telephone speech. The distinguishing factor(s), then, must reside not only in the distinguishing words themselves, but in the context in which those words were situated. One important next step for this study will be to examine context effects surrounding word errors, including word collocation and syntactic part of speech.

References

1. Boley, D.: Principal direction Divisive Partitioning. *Data Mining and Knowledge Discovery*. **2:4** (1998) 325-344 264, 267
2. Boley, D., Borst, V.: Unsupervised Clustering: A Fast Scalable Method For Large Datasets. U of MN CSE Report TR-99-029 1998 264, 265
3. Pakhomov, S.: Modeling Filled Pauses in Medical Dictations. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. (1999)
4. Pakhomov, S., Savova, G. 1999 "Filled Pause Distribution and Modeling in Quasi-Spontaneous Speech. In *Proceedings of the 14th International Congress of Phonetic Sciences (ICPS)*. (1999) 264
5. Sullivan - Pakhomov, S., Schonwetter, M.: US Patent Application For A Method and System for Generating Semi-Literal Transcripts for Speech Rercognition Systems. (2000) 264, 266

A Study on the Performance of Large Bayes Classifier

Dimitris Meretakis, Hongjun Lu, and Beat Wüthrich

Computer Science Department, Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong, China
{meretaks, luhj, beat}@cs.ust.hk

Abstract. Large Bayes (LB) is a recently introduced classifier built from *frequent* and *interesting* itemsets. LB uses itemsets to create context-specific probabilistic models of the data and estimate the conditional probability $P(c_i|A)$ of each class c_i given a case A . In this paper we use chi-square tests to address several drawbacks of the originally proposed interestingness metric, namely: (i) the inability to capture certain really interesting patterns, (ii) the need for a user-defined and data dependent interestingness threshold, and (iii) the need to set a minimum support threshold. We also introduce some pruning criteria which allow for a trade-off between complexity and speed on one side and classification accuracy on the other. Our experimental results show that the modified LB outperforms the original LB, Naïve Bayes, C4.5 and TAN.

1 Introduction

Until recently association (*descriptive*) and classification (*predictive*) mining have been considered as disjoint research and application areas. Descriptive mining aims at the discovery of strong local patterns, so-called *itemsets* [1] that hopefully provide insights on the relationships among some of the attributes of the database. Predictive mining deals with databases that consist of *labeled* tuples. Each label represents a *class* and the aim is to discover a model of the data that can be used to determine the labels (classes) of previously unseen cases.

The use of association mining techniques for classification purposes has only recently been explored. Following this route we recently proposed Large Bayes (LB) classifier [5]. LB considers each attribute-value pair as a distinct *item* and assumes that the training set is a set of transactions. During the learning phase LB employs an Apriori-like [1] association mining algorithm to discover *interesting* and *frequent labeled* itemsets. In the context of classification, we define a labeled itemset l as a set of items together with the supports $l.sup_i$ for each possible class c_i . In other words a labeled itemset provides the observed probability distribution of the class variable given an assignment of values for the corresponding attributes: $l.sup_i = P(l, c_i)$.

A new case $A = \{a_1 a_2 \dots a_n\}$, is assigned to the class c_i with the highest conditional probability $P(c_i|A) = P(A, c_i) / P(A)$. Since the denominator is constant with respect to c_i it can be ignored and the object is said to be in class c_i with the highest value $P(A, c_i)$. LB selects the longest subsets of A that are present in the set of discovered itemsets and uses them to incrementally build a *product approximation* of $P(A, c_i)$. For example, if $A = \{a_1 a_2 a_3 a_4 a_5\}$, a valid product approximation would be:

$$P(A, c_i) = P(a_2 a_5 c_i) P(a_3 | a_5 c_i) P(a_1 | a_2 a_3 c_i) P(a_4 | a_1 a_5 c_i).$$

Fig. 1 illustrates how this product approximation is incrementally generated from the set of longest itemsets by adding one itemset at each step. The formula is subsequently evaluated using the class supports of the selected itemsets and finally the class c_i with the highest probability $P(l, c_i)$ is assigned to A . Note that this process builds on the fly a local probabilistic model for the approximation of $P(A, c_i)$ that only holds for the particular classification query.

step	covered items	iset selected	product approximation	available itemsets
0	$\emptyset$	$\emptyset$	N/A	$\{a_1 a_2 a_3\}, \{a_1 a_4 a_5\}, \{a_2 a_5\}, \{a_3 a_4\}, \{a_3 a_5\}$
1	$\{a_2 a_5\}$	$\{a_2 a_5\}$	$P(a_2 a_5 c_1)$	$\{a_1 a_2 a_3\}, \{a_1 a_4 a_5\}, \{a_3 a_4\}, \{a_3 a_5\}$
2	$\{a_2 a_3 a_5\}$	$\{a_3 a_5\}$	$P(a_2 a_5 c_1)P(a_3 a_5 c_1)$	$\{a_1 a_2 a_3\}, \{a_1 a_4 a_5\}, \{a_3 a_4\}$
3	$\{a_1 a_2 a_3 a_5\}$	$\{a_1 a_2 a_3\}$	$P(a_2 a_5 c_1)P(a_3 a_5 c_1)P(a_1 a_2 a_3 c_1)$	$\{a_1 a_4 a_5\}, \{a_3 a_4\}$
4	$\{a_1 a_2 a_3 a_4 a_5\}$	$\{a_1 a_4 a_5\}$	$P(a_2 a_5 c_1)P(a_3 a_5 c_1)P(a_1 a_2 a_3 c_1)P(a_4 a_1 a_5 c_1)$	$\{a_3 a_4\}$

Fig. 1. Incremental construction of a product approximation for $P(a_1 a_2 a_3 a_4 a_5 c_i)$

The key factor in this process is the selection of interesting itemsets. In [5] we used an interestingness metric that was an adaptation of the well known cross-entropy between two probability distributions. To overcome the drawbacks of this approach we use chi-square (χ^2) tests to identify interesting itemsets. In section 4 we show experimentally that this approach leads to significant performance improvements. Moreover, we deal with the problem of setting the correct minimum support and interestingness thresholds for each data set. Although the settings we suggested in [5] work relatively well in practice, they are empirically determined and lack intuitive justification. The χ^2 test besides stemming directly from statistical theory also provides intuitive interpretation to the thresholds.

We also discuss the effect of two other pruning criteria on the performance of the classifier, namely pruning based on (a) the support and (b) the conditional entropy of the class given an itemset. Use of these criteria often leads to the generation of smaller classifiers often without significant sacrifice in the classification accuracy.

2 An Overview of Large Bayes Classifier

We will briefly outline the original LB algorithm, which is described in more details in [5]. Large Bayes is a classifier build from labeled itemsets, denoted as *itemsets* in the sequel. Consider a domain where instances are represented as instantiations of a vector $A = \{A_1, A_2, \dots, A_n\}$ of n discrete variables, where each variable A_i takes values from $val(A_i)$ and each instance is labeled with one of the $|val(C)|$ possible class labels, where C is the class-variable. A labeled itemset l with its class supports $l.sup_i$ provides the probabilities of joint occurrence $P(l, c_i)$ for l and each class c_i . The learning phase of Large Bayes aims to discover such itemsets that are frequent and interesting. As usual, an itemset is *frequent*, if its support is above the user defined minimum support threshold $minsup$: $\frac{1}{|D|} \cdot \sum_{i=1..|val(c)|} l.count_i \geq minsup$.

We can derive an estimation of the class-supports of an itemset l using two subsets of l where one item is missing. Consider for example the itemset $l = \{a_1, a_2, a_3\}$. Its class-supports $l.sup_i = P(l, c_i) = P(a_1, a_2, a_3, c_i)$ can be estimated using $l_1 = \{a_1, a_2\}$ and l_2

$= \{a_l, a_3\}$ by implicitly making certain independence assumptions: $P(a_l, a_2, a_3, c_i) = P(a_l, a_2, c_i)P(a_3|a_l, c_i) = P(l_1, c_i)P(l_2, c_i)/P(l_1 \cap l_2, c_i)$. Roughly speaking, if this estimation is accurate then l itself is not interesting, since it does not provide any more information than its subsets l_1 and l_2 . The quality of the approximation is quantified with an interestingness measure $I(l)$ that returns zero if $P(l, c_i)$ is actually equal to $P(l_1, c_i)P(l_2, c_i)/P(l_1 \cap l_2, c_i)$ and increases with their difference. An itemset is interesting if $I(l) > \tau$, where τ is a user-defined threshold. Fig. 2 presents the learning phase, which performs an Apriori-like bottom-up search and discovers the set F of itemsets that will be used to classify new cases.

```

GenItemsets(D)
In : The database D of training cases
Out: The set F of itemsets l and
      their class counts l.counti

F1 = { {aj} | aj is non class attribute }
Determine l.counti ∀ l ∈ F1, ∀ class i
for (k=2; Fk-1 ≠ ∅; k++) {
  Ck = genCandidates(Fk-1)
  For all tuples t ∈ D {
    Ci = subsets(Ck, t);
    i = class of t;
    for all candidates l ∈ Ci
      l.counti++;
  }
  Fk = selectFrequentAndInteresting(Ck) }
Return F = ∪k Fk

```

Fig. 2. Algorithm *genItemsets*

```

Classify(F, A)
In : The set F of discovered itemsets a new
      instance A
Out: The classification ci of A

cov = ∅ \ \ the subset of A already covered
nom = ∅ \ \ set of itemsets in nominator
den = ∅ \ \ set of itemsets in denominator
B = { l ∈ F | l ⊆ A and ¬∃ l' ∈ F: l' ⊆ A and l ⊂ l' }
for (k=1; cov ⊂ A; k++) {
  lk = pickNext( cov, B )
  nom = nom ∪ { lk }
  den = den ∪ { lk ∩ cov }
  cov = cov ∪ lk }
output that class ci with maximal P(A, ci):
P(A, ci) = P(ci) · ∏l ∈ nom P(l, ci) / ∏h ∈ den P(h, ci)

```

Fig. 3. Algorithm *classify*

Given a particular instance A to be classified, the set F' of the longest and most interesting itemsets in F which are subsets of A are selected. The itemsets of F' are then used to incrementally construct a product approximation for $P(A, c_i)$. The procedure *classify()* that performs this task is presented in Fig. 3 while Fig. 4 presents the selection criteria for the next itemset to be inserted in the product approximation.

```

pickNext( cov, B )
T = { l ∈ B: |l - covered| ≥ 1 };
Return an itemset lk ∈ T such that for all other itemsets lj ∈ T:
1. |lk - covered| < |lj - covered|, or
2. |lk - covered| = |lj - covered| and |lk| > |lj|, or
3. |lk - covered| = |lj - covered| and |lk| = |lj| and I(lk) > I(lj)

```

Fig. 4. Procedure *pickNext*

The resulting formula is the local model build on the fly by LB to classify A . This model implies some conditional independence assumptions among the variables but they are context-specific in the sense that different classification queries (i.e. different values of A) will produce different models making different independence assumptions. [5,6] discuss this in more detail. Finally, the formula $P(A, c_i)$ is evaluated for each c_i and A is labeled with the class c_i that maximizes $P(A, c_i)$.

3 Improving Large Bayes

A key factor affecting the performance of Large Bayes is the accurate identification of interesting itemsets. The interestingness of an itemset l is defined in terms of the error

when estimating $P(l, c_i)$ using subsets of l . Let l be an itemset of size $|l|$ and l_j, l_k be two $(|l|-1)$ -itemsets obtained from l by omitting the j^{th} and k^{th} item respectively. We can use l_j, l_k to produce an estimate $P_{j,k}(l, c_i)$ of $P(l, c_i)$:

$$P_{est}(l, c_i) = P_{j,k}(l, c_i) = \frac{P(l_j, c_i) \cdot P(l_k, c_i)}{P(l_j \cap l_k, c_i)} \quad (1)$$

Our goal is to keep those itemsets only, for which the corresponding observed probabilities differ much from the estimated ones. Information-theoretic metrics such as the cross-entropy (or Kullback-Leibler distance) are widely used [4] as a measure of the distance between the observed and the estimated probability distributions:

$$D_{KL}(P, P_{est}) = \sum_{l, c_i} P(l, c_i) \log \frac{P(l, c_i)}{P_{est}(l, c_i)} \quad (2)$$

In our case, however, the goal is to measure the distance between specific elements of the probability distribution. Consider for example a case with two variables A_1 and A_2 and $|val(A_1)| = |val(A_2)| = 4$. The corresponding sixteen 2-itemsets define the complete observed joint probability distribution $P(A_1, A_2, C)$. A high value of such metrics suggests that the class-supports of the corresponding itemsets cannot be accurately approximated *on average* by the class-supports of their subsets. To measure the accuracy of the approximation for individual itemsets in [5] we defined the interestingness $I(l|l_j, l_k)$ of l with respect to its subsets l_j and l_k as:

$$I(l|l_j, l_k) = \sum_{c_i} \left| P(l, c_i) \log \frac{P(l, c_i)}{P_{est}(l, c_i)} \right| \quad (3)$$

This ad-hoc measure presents certain drawbacks with respect to its ability to identify interesting local patterns. Consider for example a domain with two classes and an itemset l for which $P(l, c_1) = 0$, $P(l, c_2) = 0.15$ and $P_{est}(l, c_1) = 0.1$, $P_{est}(l, c_2) = 0.15$. Although this is indeed a very interesting itemset since the estimated probability for c_1 greatly differs from observed one, $I(l|l_j, l_k) = 0$ and l is discarded as non-interesting. In addition, our interestingness measure (but also every information-theoretic measure) suffers from the fact that it ignores the sample size and assumes that the sample probability distribution is equal to the population probability distribution thus ignoring the possibility that the differences occurred purely because of chance.

In the sequel we describe the application of chi-square (χ^2) tests to overcome these problems. We reduce the problem of deciding whether an itemset is interesting to applying a hypothesis-testing procedure on the following hypotheses:

H_0 : $P(l, c_i) = P_{est}(l, c_i)$, i.e. l is not-interesting

H_1 : $P(l, c_i) \neq P_{est}(l, c_i)$, i.e. l is interesting

To test the hypotheses we calculate the χ^2 test statistic with $|C|$ degrees of freedom. ($|D|$ is the database size, $|C|$ the number of classes):

$$\chi_l^2 = \sum_{j=1}^{|C|} \frac{(P(l, c_j) \cdot |D| - P_{est}(l, c_j) \cdot |D|)^2}{P_{est}(l, c_j) \cdot |D|} = \sum_{j=1}^{|C|} \frac{(P(l, c_j) - P_{est}(l, c_j))^2}{P_{est}(l, c_j)} * |D| \quad (4)$$

If $\chi_l^2 > \chi_{p, |C|}^2$ the null hypothesis H_0 is rejected and l is considered interesting. The statistical-significance threshold p is user-defined but should in general be high i.e. $p < 0.05$ since discovering non-interesting itemsets does not improve the accuracy and unnecessarily increases the complexity of the resulting classifier. The degrees of freedom for the test are $|C|$ since the sums of the expected and the observed

frequencies of an itemset are generally different [8]. If the degrees of freedom are two or less, Yates correction is applied (subtracting 0.5 from the absolute difference in Eq. (4) before squaring, when this difference exceeds 0.5).

A problem associated with χ^2 tests is that the estimated frequencies $P_{est}(l, c_i) \cdot |D|$ in each term of in Eq. (4) should be not too small otherwise the test is sensitive to errors. To overcome this problem we apply a merging step before calculating the χ^2 -statistic. During this step the class with the smallest frequency is merged with the immediate larger class to form a composite class containing the sum of the frequencies. The corresponding observed frequencies are merged also and the degrees of freedom (df) are reduced by one. The merging phase stops when all expected frequencies are large enough or when all class-frequencies are merged. Following standard statistical practice we set the minimum value of an expected frequency to 5 if $df = 2$ and 3 if $df > 2$, otherwise it is merged.

As a result of the merging step, each itemset l has a value χ_l^2 that refers to different degrees of freedom df_l . To compare these values with the minimum required threshold $\chi_{p, |C|}^2$ we need a degrees-of-freedom-independent test. For that reason we take advantage of the fact that the value $t = \sqrt{2 \cdot \chi^2} - \sqrt{2 \cdot df - 1}$ approximately follows the normal distribution and therefore the modified requirement for an itemset to be interesting becomes:

$$t = \sqrt{2 \cdot \chi_l^2} - \sqrt{2 \cdot df_l - 1} > \sqrt{2 \cdot \chi_{p, |C|}^2} - \sqrt{2 \cdot |C| - 1} \quad (5)$$

Note that although t can now take negative values as well, the requirement for an itemset to be interesting remains that its t value is bigger than the threshold of equation (5) which is determined by the required statistical significance level p and the number of classes $|C|$.

3.1 Pruning Criteria: Trading off Accuracy for Simplicity and Speed

A difficult challenge in the design of classifiers is preventing overfitting and generating simple models. Simple models are not only easily interpretable but also generalize better in unseen data and are faster to build and evaluate. In LB overfitting translates to the discovery of “too many itemsets” and this is particularly true in domains with many multi-valued attributes, where the search space is huge. χ^2 tests significantly reduce the number of discovered itemsets to a tractable amount. However, there are some other pruning criteria that can potentially reduce the number of itemsets and accelerate both the learning and classification phase.

Support-based pruning is used by many classification methods including decision trees, where a leaf is not expanded if it contains less than a minimum number of cases. In section 4 we evaluate the effect of support pruning on the accuracy of LB.

A somehow more effective pruning criterion is the conditional entropy of the class C given an itemset l :

$$H(C|l) = \sum_{i=1}^{|C|} P(c_i|l) \cdot \log P(c_i|l)$$

Conditional entropy takes values ranging from zero (if l only appears with a single class) to $\log(|C|)$, if l 's appearances are uniformly distributed among the classes. If $H(C|l)$ is very small l bears almost certainty about a class and therefore needs not be expanded. In the next section we show that the

introduction of a relatively low conditional entropy threshold often reduces the size of the classifier without significantly affecting its accuracy.

4 Experimental Results

To evaluate the performance of LB with the χ^2 tests (LB-chi2), we use 23 data sets from the UCI ML Repository [7] with a special preference on the largest and more challenging ones in terms of achievable classification accuracy. We compared LB-chi2 with the originally proposed version of LB, the Naïve Bayes classifier [2] (since in the extreme case if only 1-itemsets are used LB reduces to NB), Quinlan’s Decision Tree classifier C4.5 [9], and TAN [4]; a Bayesian Network classifier that relaxes the independence assumptions of NB by using some pairs of attributes.

Accuracy was measured either using 10-fold cross validation (CV-10) for small data sets or the holdout method (training and testing set split) for the larger ones. The train and test set splits and the cv-folds were the same for all results reported. Since all methods except of C4.5 only deal with discrete attributes, we used entropy-based [3] discretization for all continuous attributes. No discretization was applied for C4.5.

The factor most affecting the results is the p-value of the χ^2 tests. We experimented with 0.01, 0.025, 0.005, 0.001 and 0.0005, and selected 0.005 as the most effective one. Higher p-values slowly deteriorated the accuracy and tended to produce more complex, larger and slower classifiers. This is natural since high p-values cause more itemsets to be characterized as interesting. On the other hand, values below $p=0.005$ caused most of the itemsets to be rejected as non-interesting and generated simplistic classifiers with poor accuracy. The effects of the varying p-values on the average accuracy and classifier size can be seen on figure 5.

Table 1 provides a comparison of the algorithms in the 23 data sets according to five criteria. LB-chi2 outperforms all others according to all criteria indicating that it is indeed a very accurate classifier. The criteria used are: (1) Average Accuracy of the classifiers, (2) Average Rank (Smallest values indicate better performance on average), (3) The number of wins–losses of LB-chi2 against other algorithms, and the statistical significance of the improvement of LB-chi2 against each algorithm using (4) a one-sided paired t-test and (5) a Wilcoxon paired, signed, one-sided, rank test.

Table 1. Comparison of the classifiers according to various criteria

		NB	C4.5	TAN	LB	LB-chi2
1	Average Accuracy	0.8187	0.8147	0.8376	0.8332	0.8434
2	Average Rank	3.695652	3.73913	2.73913	2.652174	1.913043
3	No wins vs.:	19 – 4	19 – 4	16 – 6	15–7	--
4	1-side Paired t-test	0.9995	0.9991	0.9940	0.9828	--
5	Wilcoxon paired signed rank test	>0.995	>0.995	>0.99	>0.975	--

Table 2. Summary Table of datasets and results. $|A|$ = number of attributes, $|I|$ =number of distinct items (attribute-value pairs) after discretization, $|C|$ = number of classes, Miss = presence of missing values. Last two colums indicate the training/testing time of LB-chi2 in sec

Data Set	Data set Properties						Accuracy					Time (s) LB-chi2	
	$ A $	$ I $	$ C $	Mis	# Train	# Test	NB	C4.5	TAN	LB	LB-chi2	Train	Test
1 Adult	14	147	2	Yes	32561	16281	0.8412	0.854	0.8571	0.8511	0.8668	48.81	37.04
2 Australian	14	48	2	No	690	CV-10	0.8565	0.8428	0.8522	0.8565	0.8609	0.18	0.03
3 Breast	10	28	2	Yes	699	CV-10	0.97	0.9542	0.9671	0.9686	0.9714	0.11	0.02
4 Chess	36	73	2	No	2130	1066	0.8715	0.995	0.9212	0.9024	0.9418	1.99	2.20
5 Cleve	13	27	2	Yes	303	CV-10	0.8278	0.7229	0.8122	0.8219	0.8255	0.07	0.01
6 Flare	10	27	2	No	1066	CV-10	0.7946	0.8116	0.8264	0.8152	0.818	0.18	0.03
7 German	20	60	2	No	999	CV-10	0.741	0.717	0.727	0.748	0.75	0.42	0.07
8 Heart	13	17	2	No	270	CV-10	0.8222	0.7669	0.8333	0.8222	0.8185	0.05	0.01
9 Hepatitis	19	32	2	Yes	155	CV-10	0.8392	0.8	0.8188	0.845	0.8446	0.05	0.01
10 Letter	16	146	26	No	15000	5000	0.7494	0.777	0.8572	0.764	0.8594	109.29	56.90
11 Lymph	18	49	4	No	148	CV-10	0.8186	0.7839	0.8376	0.8457	0.8524	0.07	0.02
12 Pendigits	16	151	10	No	7494	3499	0.8350	0.923	0.9360	0.9182	0.9403	44.23	22.58
13 Pima	8	15	2	No	768	CV-10	0.759	0.711	0.7577	0.7577	0.7564	0.06	0.02
14 Pima Diabetes	8	14	2	No	768	CV-10	0.7513	0.7173	0.7656	0.7669	0.763	0.07	0.02
15 Satimage	36	384	6	No	4435	2000	0.818	0.852	0.872	0.839	0.8785	392.60	83.75
16 Segment	19	147	7	No	1540	770	0.9182	0.958	0.9351	0.9416	0.9429	2.28	1.16
17 Shuttle-small	9	50	7	No	3866	1934	0.987	0.995	0.9964	0.9938	0.9948	1.40	0.78
18 Sleep	13	113	6	No	70606	35305	0.6781	0.7310	0.7306	0.7195	0.7353	476.71	621.97
19 Splice	59	287	3	No	2126	1064	0.9464	0.933	0.9463	0.9464	0.9408	3.24	3.07
20 Vehicle	18	69	4	No	846	CV-10	0.6112	0.6982	0.7092	0.688	0.7187	1.24	0.23
21 Vote Records	16	48	2	No	435	CV-10	0.9034	0.9566	0.9332	0.9472	0.9334	0.13	0.04
22 Waveform-21	21	44	3	No	300	4700	0.7851	0.704	0.7913	0.7943	0.7913	0.1	2.724
23 Yeast	8	18	10	No	1484	CV-10	0.5805	0.5573	0.5721	0.5816	0.5816	0.15	0.04

Table 2 provides information about the data sets, lists the accuracies of the classifiers and shows the training and testing time of LB-chi2 on all data sets (Measured on a 400MHz Pentium WinNT PC). Noticeably, the biggest improvements in accuracy against the original LB came mostly from the largest data sets; this indicates the inability of the originally used interestingness metric in such cases.

The p-value for chi2-LB was set to 0.005 and to facilitate more accurate χ^2 tests the minimum support was set to $\max\{10, 2*|c|\}$. Although this is a minimum requirement in order for the test statistic to be accurate this can be further increased in order to reduce both the training time and the size of the classifier as discussed below.

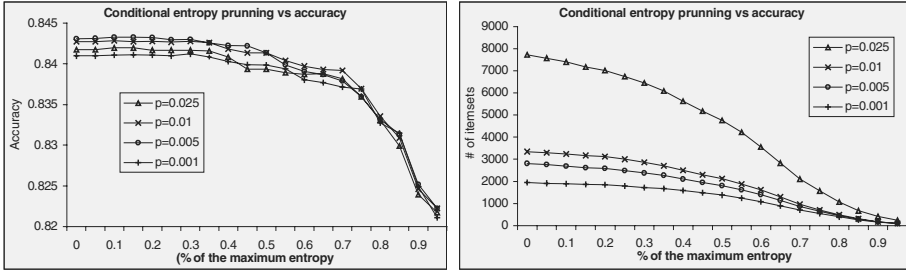


Fig. 5. Effect of conditional entropy pruning on (a) accuracy and (b) size of classifier for four different p -value thresholds. x-axis contains the threshold as a percentage of the maximum conditional entropy $\log|C|$

Figure 5 illustrates the effect of conditional entropy pruning on the average accuracy (5a) and size (5b) of LB. Since the number of classes is different among the datasets the minimum threshold $\min H$ is expressed as a percentage of the maximum conditional entropy $\log|C|$. In a 4-class domain, for example, a value of 0.2 implies that $\min H = 0.2 \cdot \log 4 = 0.4$. Values for $\min H$ of up to $0.3 \cdot \log|C|$ have little impact on the accuracy while at the same time reducing the size of the classifier. The rightmost values of the graph correspond to maximum pruning where only 1-itemsets are used and therefore represent the accuracy and size of Naïve Bayes classifier.

Support pruning has a more drastic effect on the size of the classifier as can be seen in Figure 6. This is particularly true on large data sets like “sleep” where 10 occurrences for an itemset l represent a probability $P(l)=0.0001$. Increasing the minsup threshold to 0.005 in this data set reduced the number of itemsets discovered from 31000 to 7500 while the accuracy fell only slightly, from 0.7336 to 0.727.

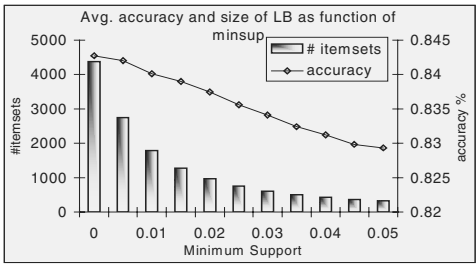


Fig. 6. Effect of support pruning on average accuracy and size of LB

References

1. R. Agrawal, R. Srikant, Fast algorithms for mining association rules, *VLDB-94*, 1994.
2. R.Duda, P.Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.
3. U.M.Fayyad, K.B.Irani, Multi-Interval discretization of continuous-valued attributes for classification learning, *13th IJCAI*, 1022-1027, 1993.

4. N.Friedman, D. Geiger, M. Goldszmidt, Bayesian Network Classifiers, *Machine Learning*, 29, 131-163, 1997.
5. D. Meretakis, B.Wüthrich, Extending Naïve Bayes Classifiers Using Long Itemsets, *KDD-99*, pp 165-174, San Diego, USA, 1999.
6. D.Meretakis, B.Wüthrich, Classification as Mining and Use of Labeled Itemsets, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (DMKD'99), Philadelphia, USA, 1999.
7. C.J.Merz, P.Murphy, UCI repository of machine learning databases, 1996 (<http://www.cs.uci.edu/~mlearn/MLRepository.html>).
8. W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *Numerical Recipes in C*, 2nd Ed, Cambridge University Press, 1992.
9. J.R.Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

Dynamic Discretization of Continuous Values from Time Series^{*}

Llanos Mora López¹, Inmaculada Fortes Ruiz², Rafael Morales Bueno¹, and Francisco Triguero Ruiz¹

¹ Dpto. Lenguajes y C.Computacion. E.T.S.I.Informtica.
Campus de Teatinos. 29071 Málaga, Spain
`{llanos,morales}@lcc.uma.es`
`triguero@ctima.uma.es`

² Dpto. Matematica aplicada. E.T.S.I.Inf.
C. Teatinos. 29071 Málaga, Spain
`ifortes@ctima.uma.es`

Abstract. Two methods to assign discrete values to continuous values from time series, using dynamic information about the series, are proposed. The first method is based on a particular statistic which allows us to select a discrete value for a new continuous value from the series. The second one is based on a concept of significant distance between consecutive values from time series which is defined. This definition is based on qualitative changes in the time series values. In both methods, the conversion process of continuous values into discrete values is dynamic in opposition to static classical methods used in machine learning. Finally, we use the proposed methods in a practical case. We transform the daily clearness index time series into discrete values. The results display that the series with discrete values obtained from the dynamic process captures better the sequential properties of the original continuous series.

1 Introduction

The goal of data analysis by time series is to find models which are able to reproduce the statistical characteristics of the series. Moreover, these models allow us to predict next values of the series from its predecessors.

One of the most detailed analysis of statistic methods for the research of time series has been done by Box & Jenkins [2]. The mathematical model for a time series is the concept of discrete-time stochastic process. It is supposed that the observed value of the series at time t is a random sample of size one from a random variable X_t , for $t \in \{1, \dots, n\}$. A time series of length n is a random sample of a random vector like this $(X_1, \dots, X_n)$. The random vector is considered as part of a discrete-time stochastic process, and observed values of the random variables are considered as the evolution of the process. The process

^{*} This work has been partially supported by project FACA number PB98-0937-C04-01 of the CICYT, Spain. FACA is a part of the FRESCO project

is completely known if the joint probability distribution function of each random vector is known; when the series is Gaussian, the process is completely known when all first and second order moments are known.

The following steps are pursued in the analysis of data using time series theory: identification of the model, estimation of parameters, diagnosis of the model and prediction of new values. The identification of the model can be achieved either in the time domain, using the sample and partial autocorrelation functions, or in the frequency domain, using spectral analysis. In both cases, a previous selection of the possible models which can be used to fit the data must be done. This can be a restriction in the final results. Another important restriction is the following: once the model has been identified and the parameters have been estimated it is supposed that the relation between the parameters is constant along the time. However, in many time series this can not be true.

On the other hand, the analysis of time series and stochastic process has also been analyzed from machine learning techniques. Some of these techniques have solved successfully the restrictions noted above. Two of the most important works in this line have been developed by D.Ron et al., [10,11], where the use of probabilistic finite automata is proposed. In [11], a subclass of probabilistic finite automata has been used for modeling distributions on short sequences that correspond to objects such as single handwritten letters, spoken words, or short protein sequences. In [10], another subclass of probabilistic finite automata, called probabilistic suffix automata, has been used to describe variable memory length Markov processes. Other works arise from the work developed by Dagum [3], based on belief network models; in [3], it is proposed the use of dynamic network models, which are a compromise between belief network models and classical models of time series. They are based on the integration of fundamental methods of Bayesian analysis of time series. However, almost all models used for time series from machine learning are restricted to input features with known discrete values, not allowing continuous valued features as input. For this reason, before any of this method is used, it is necessary to transform the observed continuous values into discrete values. Any method to obtain discrete values must have the following two features: first, it must be known how many different discrete values can appear in the series; second, it must be able to quantify how different two or more consecutive values of the series are.

Let us consider, as an example, the following time series: $\{..., 0.80, 0.82, 0.95, 0.94, 0.96, 0.94, 0.96, \dots\}$, which corresponds to measures of cloudiness index (fraction of overcast sky) for consecutive days. The possible values of this index range from 0 to 1 (clear sky and completely overcast, respectively). With this parameter precision 100 different values can be obtained in the series. However, in most applications, a few different values will suffice to characterize this index and then obtain significant information about the cloudiness of the sky. For instance, the former series can be described as: {overcast, overcast, completely overcast, completely overcast, completely overcast, completely overcast, completely overcast}; that is, the qualitative values {clear sky, ..., almost completely overcast, completely overcast} -corresponding to {from 0.0 to 0.1, from

0.1 to 0.2, ..., from 0.9 to 1.0}- give us all the information we need. Therefore, a possible way to transform into discrete values consists of using fixed-size intervals. We refer to this transformation as static discrete conversion. In this paper we propose a transformation which we will refer to as dynamic discrete conversion. The static discrete conversion methods group a set of items into a hierarchy of subsets whose items are related in some meaningful way. Typically, these algorithms perform the conversion into discrete values according to statistical stationary properties of the values, not taking into account the evolution of these values. This procedure has various problems. Consider, for example, the following cloudiness index series: {0.71, 0.89, 0.89, 0.91, 0.89}. Using the proposed static discrete conversion, we obtain the series: {half overcast, almost completely overcast, almost completely overcast, overcast, almost completely overcast}. However, if we observe the series -or this situation in the real world-, we will probably not consider as different situations those when the cloudiness index take value 0.89 or 0.91. To circumvent this problem, a new approach is developed in this paper to transform continuous values into discrete ones. We refer to it as dynamic qualitative discrete conversion: dynamic because it takes into account the evolution of the series; and qualitative because the selection of the discrete value is based on a significant distance which is defined below.

With this type of discretization, we come closer to the form of knowledge of nature phenomena we have in our mind, and we overcome the limitation of a static arithmetic concept. Machine learning techniques and qualitative reasoning allow us to overcome the rigid arithmetic concepts underlying in any equation and to come closer to the language of the brain which, as Neumann said, “is not the mathematical language”.

In Section 2 we develop our two approaches to obtain discrete values. We explain the static conversion and the two dynamic conversion that we propose. In Section 3 a practical case is described using the algorithms developed in this paper. Finally, conclusions and possible extensions are summarized in Section 4.

2 Dynamic Qualitative Discretization

In this section we explain the basic idea of this work: the development of an alternative dynamic discrete conversion method. The goal is to develop an effective and efficient method to transform continuous values into discrete ones using the overall information included in the series and, when possible, feedback with the learning system. To do this, the discrete value which corresponds to a continuous value is calculated using qualitative reasoning, taking into account the evolution of the series.

Qualitative models have been used in different areas in order to get a representation of the domain based on properties (qualities) of the systems which, additionally, allows us to avoid the use of complex mathematical models, [5], [6]. One of the objectives which has been pursued is to develop an alternative physics in which the concepts are derived from a far simpler, but nevertheless formal,

qualitative basis. Qualitative reasoning can also be used to predict the behavior of systems, [7].

On the other hand, any process of discretization has some psychological plausibility since in many cases humans apparently perform a similar preprocessing step representing temperature, weather, speed, etc., as nominal (discrete) values. Following [13], the desirable attributes for a discretization method are:

- Measure of classification “goodness”
- No specific closeness measure
- No parameters
- Globality rather than locality
- Simplicity
- Use of feedback
- Use of *a priori* knowledge
- Higher order correlations
- Fast

The more typical way to deal with numeric variables is to create a partition $\mathcal{P}$ in the range of possible values of the variable, and treat each subset of the partition as a single discrete value. If $\mathcal{P}$ is chosen too thick then important distinctions are missed; if $\mathcal{P}$ is chosen too fine, the data are over-partitioned and the probability estimates may become unreliable. On the other hand, the best partition $\mathcal{P}$ depends on the size of the series which is to be partitioned. A partition with subsets of fixed size is a static partition; if a discrete value is assigned take into account precedent values then we obtain a dynamic partition.

When using a static discrete conversion method, the continuous values are transformed into s discrete values through s intervals of same length. Specifically, the width w_X of a discretized interval is given by:

$$w_X = \frac{\max\{X_t\} - \min\{X_t\}}{s}, \quad (1)$$

where, hereafter, $\max$ and $\min$ are always considered for $t \in \{1, \dots, n\}$. The discrete value v_i corresponding to a continuous value X_i of the series is an integer from 1 to s which is given by:

$$v_i = \text{discretize}(X_i) = \begin{cases} s & \text{if } X_i = \max\{X_t\} \\ [(X_i - \min\{X_t\})/w_X] + 1 & \text{otherwise} \end{cases}, \quad (2)$$

where $[A]$ means the integer part of A . After deciding upon s and finding w_X , it is straightforward to transform the continuous values into discrete ones using this expression.

In this paper we propose the use of a qualitative dynamic discrete conversion method. It is dynamic because the discrete value associated to a particular continuous value can change along the time: that is, the same continuous value can be discretized into different values, depending on the previous values observed in the series. It is qualitative because only those changes which are qualitatively

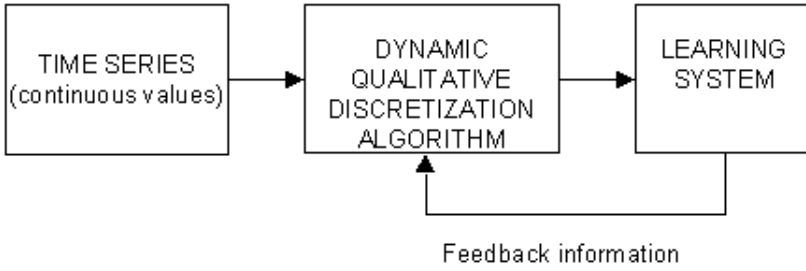


Fig. 1. Dynamic qualitative discretization model

significant appear in the discretized series. Moreover, with this dynamic method the information from the learning model can be take into account.

The process to generate the discrete values is described in Figure 1.

The dynamic qualitative discretization algorithm (DA) is connected to some learning system (LS). Information is fed forward from the DA to the LS. The LS generates feedback for the DA in order to improve the discretization of the continuous inputs. For example, when predicting using variable memory order Markov models, we can decide what discrete value is associated to a particular continuous value either using the probabilities estimated in the model or asking information to the LS about the number of states used to construct the model. Based on this feedback, the DA may perform the adjust of the discrete values corresponding to continuous ones.

With these ideas we propose two procedures to obtain time series with discrete values taking into account the preceding values for the discretization of each value. For both methods, we first justify its use and then propose an algorithm to implement it.

2.1 Using a t Statistic

The idea behind this method is to use statistical information about the preceding values observed from the series to select the discrete value which corresponds to a new continuous value of the series. A new continuous value will be associated to the same discrete value as its preceding values if the continuous value belongs to the same population. Otherwise, the static discrete conversion method will assign a new discrete value to this new continuous value. To decide if a new continuous value belongs to the same population as the previous ones, a statistic with Student's t distribution is computed. The method is formally described below.

Given a set of observations, $X_1, \dots, X_n, X_{n+1}$, it is possible to examine whether X_{n+1} belongs to the same population as the previous values using the statistic:

$$t_{observed} = \frac{X_{n+1} - \bar{X}}{\sqrt{\hat{\sigma}^2(1 + 1/n)}}, \quad (3)$$

where $\bar{X} = n^{-1} \sum_{i=1}^n X_i$, and $\hat{\sigma}^2 = (n-1)^{-1} \sum_{i=1}^n (X_i - \bar{X})^2$. As it is proved in the Appendix, if certain statistical conditions are met, when X_{n+1} comes from the same population as the previous values the statistic $t_{observed}$ has Student's t distribution with $n-1$ degrees of freedom. This property suggests the following algorithm:

Algorithm to discretize continuous values from time series using the t statistic

Input:

continuous time series $\{X_t\}$

α : significance level, t_α

s : number of intervals

Method:

$v_1 \leftarrow discretize(X_1)$

$ini = 1$

for $i = 2, \dots, n$ do

if $((i - ini) > 1)$ then

$$\bar{X} \leftarrow \left(\sum_{j=ini}^{i-1} X_j \right) / (i - ini)$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{j=ini}^{i-1} (X_j - \bar{X})^2}{(i - ini) - 1}}$$

$$t_{observed} \leftarrow |X_i - \bar{X}| / \hat{\sigma}$$

$$v_i = \begin{cases} discretize(X_i) & t_{observed} > t_{\alpha/2} \\ v_{i-1} & \text{otherwise} \end{cases}$$

else

$$v_i = discretize(X_i)$$

if $v_{i-1} \neq v_i$ then

$$ini = i$$

end

Output:

Discrete time series: $\{v_i\} = \{v_1, \dots, v_n\}$

2.2 Using Qualitative Reasoning

This method is based on the ideas of qualitative reasoning. In order to characterize the evolution of the system and select discrete values, we propose to use distance functions. These distance functions measure the relationship between consecutive values. They have been used in Instance-Based learning, to determine how close a new input vector is to each stored instance, and use the nearest instance or instances to predict the output class. Therefore, distances are often normalized by dividing the distance for each attribute by the range (i.e. the difference between maximum and minimum) of that attribute, so that the distance for each attribute is in the approximate range $[0, 1]$. It is also common to use

standard deviation instead of range in the denominator. Domain knowledge can often be used to decide which method is most appropriate.

Using some of these ideas we have defined the concept of significant distance between values of the series: two consecutive continuous values correspond to the same discrete value when the distance between them is smaller than a threshold significant distance. This significant distance can be absolute (ASD) -the same for all the sequence- or relative (RSD) to the values which are being compared. We propose the use of the following expressions for these two distance functions:

$$ASD = \frac{|X_i - X_j|}{range\{X_t\}}, \quad (4)$$

$$range\{X_t\} = \max\{X_t\} - \min\{X_t\}, \quad (5)$$

$$RSD = \frac{|X_i - X_j|}{|X_i|}. \quad (6)$$

The proposed expression for the ASD is based on the euclidean metric distance, [14]. The new discrete value is determined depending on how far it is from the preceding values. Changes above the threshold involve changes in the discrete value. When this procedure is used, smooth changes may not be detected, especially if the time series evolves slowly but always in an increasing or decreasing way. For instance, in the time series $\{0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96\}$ all continuous values would be assigned to the same discrete value. To solve this problem we propose to consider only the most recent values of the series to estimate the significant distance.

The first continuous value of the time series is used as reference value. The next values in the series are compared with this reference. When the distance between the reference and a specific value is greater than the threshold (there is a significant difference between them), the comparison process stops. For each value between the reference and the last value which has been compared, the following distances are computed: distance between the value and the first value of the interval, and distance between the value and the last value of the interval. If the former one is lower than the latter one, the discrete value assigned is the one corresponding to the first value; otherwise, the discrete value assigned is the one corresponding to the last value. We now formally describe the algorithm which implements this dynamic qualitative discrete conversion process.

Algorithm to discretize continuous values from time series using flags

Input:

continuous time series $\{X_t\}$,
absolute (relative) significant distance, ADF (RSD)
 s : number of intervals

Method:

$v_1 \leftarrow discretize(X_1)$

```

ref  $\leftarrow X_1$ 
marked  $\leftarrow 1$ 
for  $i = 2, \dots, n$  do
    dist  $\leftarrow |X_i - \text{ref}|$ 
     $v_i = \begin{cases} v_{i-1} & \text{if } \text{dist} < ADS \text{ (or } RDS) \\ discretize(X_i) & \text{otherwise} \end{cases}$ 
    if  $v_{i-1} \neq v_i$  then
        ref  $\leftarrow X_i$ 
         $j \leftarrow i$ 
         $k \leftarrow 1$ 
        while  $|z_i - z_{i-k}| < |z_{i-k} - z_{j-\text{marked}}|$ 
             $v_{j-k} \leftarrow v_j$ 
             $k \leftarrow k + 1$ 
        end
        marked  $\leftarrow 1$ 
    else
        marked  $\leftarrow \text{marked} + 1$ 
    end
end
Output:
Discrete time series:  $\{v_n\}$ 

```

3 Experimental Results

We have used the proposed dynamic discretization algorithms to obtain discrete series from continuous ones. The input data we have used are daily clearness index time series recorded in 10 Spanish stations. The daily clearness index is a climatic parameter obtained from the normalization of daily global radiation received in the surface of the earth. The normalization factor is the extraterrestrial radiation. The values of clearness index range between 0 and 1. In Figure 2 we can observe a fragment of these series, corresponding to data of Málaga in May, 1993 (the total number of observations from Málaga is approximately 2500).

For this series we have used the following method of discretization:

1. Static discretization
2. Dynamic qualitative discretization using the t statistic
3. Dynamic qualitative discretization using flags.

With the first method, the resulting discretized series has jumps which do not correspond to significant changes in the continuous values. The other two methods show more accurately the changes which are observed in the continuous time series.

To analyze whether the continuous series $\{X_t\}$ and the discrete series $\{v_t\}$ -obtained using any of the tree discretization methods- are similar, statistical tests were used to compare their means, variances and cumulative probability distribution functions (cpdf). In all cases, we can accept the hypothesis that the discretized series has the same mean, variance and pdf as the original one

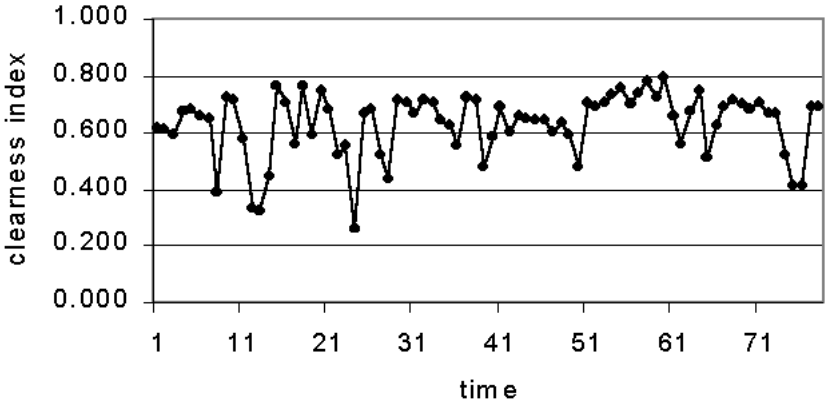


Fig. 2. Fragment of a series of daily clearness index

Table 1. Number of different sequences which appear in discretized series

Method	Order(length)			
	2	3	4	5
Static discret.	91	596	2627	6373
Dynamic disc. (t-statistic)	84	520	2157	4699
Dynamic disc.(qualitative reasoning)	86	571	2230	4875

(tests have been carried out with 0,95 as significance level; the pdf's have been compared using the Kolmogorov-Smirnov two-sample statistic -see [12], pp.401-403, for a description of this statistic). Thus, the original time series and the discretized one seem to have similar statistical characteristics. In Figure 3, it is depicted the cpdf of the original series from Málaga and the cpdf of the three discretized series which have been obtained from the original one.

On the other hand, clearness index series have been studied using time series models, such as Markov models, [1,8]. To use Markov models, the input data are sequences built with a set of terminal symbols. In this case, the terminal symbols are the different discrete values. Basically, Markov models of order m analyze the probability of appearance of sequence with length m . Using this information, a model is built for the series (for instance, probabilistic finite automata). With this idea in mind, we have analyzed the number of different sequence which appear in our discretized series for order $m = 2, 3, 4$ and 5. In Table 1 we show the results which have been obtained:

We observe that the number of different sequences which appear is lower when dynamic models are used to discretize the series (compare rows two and three in Table 1 with row one). This is a logical result because when the evolution of the series is taking into account, the real changes which take place in the original series are detected more precisely.

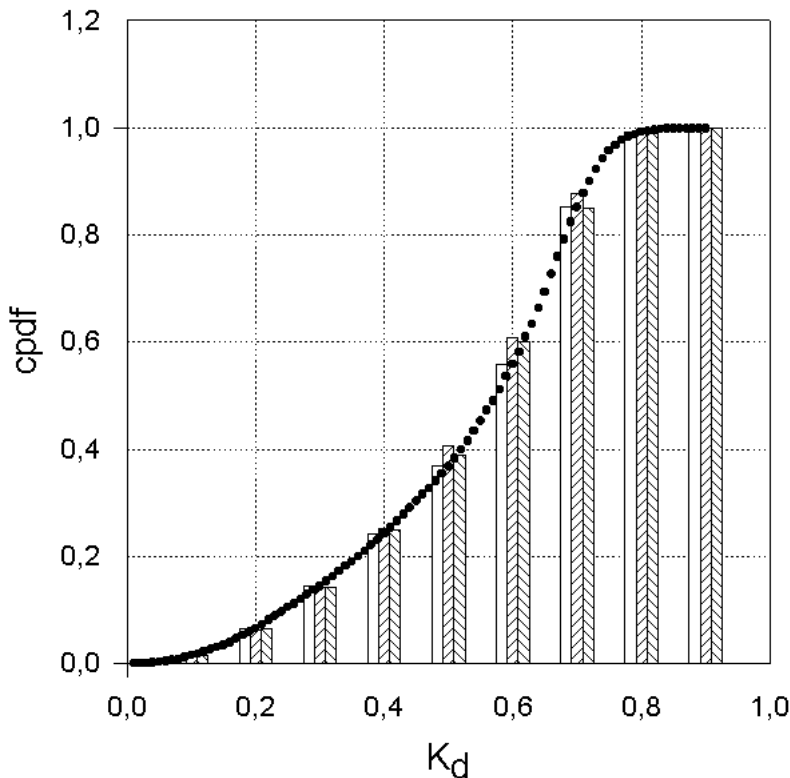


Fig. 3. cpdf for continuous (*points*) and discretized series (*rectangles; from left to right: static discretization, dynamic discretization using the t -statistic and dynamic qualitative discretization using flags*)

4 Conclusions and Future Work

Most of the machine learning models used for time series only work with discrete values. For this reason, before using any of these methods, it is necessary to convert them into discrete values. We have developed two dynamic methods to discretize continuous input values from time series. The main contribution of the work is that, with these methods, the evolution of the time series can be taken into account in the discretization process.

The algorithms proposed have been used in a practical case: the discretization of a climatic parameter -namely, the daily clearness index. Our results show that if a dynamic method is used to discretize continuous values from a time series, then the resultant series captures more accurately the true evolution of the series.

A fixed number of intervals has been used in the discretization process. It would be interesting to design a discretization process which decides, for each

specific case, what is the maximum and minimum number of intervals which can be used to obtain a discretized series which behaves similarly as the original one. We will evaluate our methods with another data sets in a immediately future.

References

1. AGUIAR, R.J., COLLARES-PEREIRA, M., CONDE J.P. Simple procedure for generating sequences of daily radiation values using a library of Markov Transition Matrix. *Solar Energy*, 40, 269-279, 1988. 288
2. BOX, G.E.P., JENKINS, G.M. *Time Series Analysis forecasting and control*. USA: Prentice Hall, 1976. 280
3. DAGUM, P., GALPER, A. Time series prediction using belief network models. *Int. Journal Human-Computer Studies*, 42, 617-632, 1995. 281
4. FELLER, W. *An Introduction to Probability Theory and Its Applications*. Vol. 1. Chapman & Hall, London; John Wiley & Sons, New York, 3rd ed., 1968. 290
5. FORBUS, K.D. Qualitative Process Theory, *Artif. Intell.* 24: 85-168, 1984. 282
6. KLEER, J. BROWN, J.S. A qualitative physics based on confluences, *Artif. Intell.* 24: 7-83, 1984. 282
7. KUIPERS, B. Commonsense reasoning about causality: deriving behavior from structure, *Artif. Intell.* 24: 169-203, 1984. 283
8. MORA-LÓPEZ, L., SIDRACH-DE-CARDONA, M. Characterization and simulation of hourly exposure series of global radiation. *Solar Energy*, 60, No.5, 257-270, 1997. 288
9. RABINER, L.R. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the Seventh Annual Workshop on Computational Learning Theory*, 1994.
10. RON, D., SINGER, Y., TISHBY, N. Learning Probabilistic Automata with Variable Memory Length. *Proceedings of the Seventh Annual Workshop on Computational Learning Theory*, 1994. 281
11. RON, D., SINGER, Y., TISHBY, N. On the Learnability and Usage of Acyclic Probabilistic Finite Automata. *Journal of Computer and System Sciences*, 56, 133-152, 1998. 281
12. SHORACK, G.R., WELLNER, J.A. *Empirical Processes with Applications to Statistics*. Wiley Series, New York, 1986. 288
13. VENTURA, D., MARTINEZ, T.R. BRACE: A Paradigm for the Discretization of Continuously Valued Data. *Proceedings of the Seventh Florida Artificial Intelligence Research Symposium*, 117-21, 1994. 283
14. WILSON, D.R., MARTINEZ, T.R. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, 6, 1-34, 1997. 286

Appendix

Assume that $X_1, \dots, X_n$ are independent and identically distributed observations from a Gaussian population. It is well-known (see [4]) that:

$$\bar{X} \sim N(\mu, \sigma^2/n); \quad \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{\sigma^2} \sim \chi_{n-1}^2, \quad (7)$$

where $\bar{X} = n^{-1} \sum_{i=1}^n X_i$, $\mu = E[X_i]$, $\sigma^2 = \text{var}(X_i)$ and χ_{n-1} denotes the chi-square distribution with $n-1$ degrees of freedom; moreover these two distributions are independent. Hence, if X_{n+1} is another observation from the same Gaussian population, and independent of the previous ones, then:

$$\frac{X_{n+1} - \bar{X}}{\sqrt{\sigma^2(1 + 1/n)}} \sim N(0, 1); \quad \frac{(n-1)}{\sigma^2} \hat{\sigma}^2 \sim \chi_{n-1}^2, \quad (8)$$

where $\hat{\sigma}^2 = (n-1)^{-1} \sum_{i=1}^n (X_i - \bar{X})^2$, and these statistics are also independent. Therefore:

$$\frac{\frac{X_{n+1} - \bar{X}}{\sqrt{\sigma^2(1 + 1/n)}}}{\sqrt{\frac{(n-1)}{\sigma^2} \cdot \frac{\hat{\sigma}^2}{(n-1)}}} \sim t_{n-1}, \quad (9)$$

where t_{n-1} denotes Student's t distribution with $n-1$ degrees of freedom. The left-hand member of this last expression is precisely our statistic t_{observed} .

Using a Symbolic Machine Learning Tool to Refine Lexico-syntactic Patterns^{*}

Emmanuel Morin and Emmanuelle Martienne

Institut de Recherche en Informatique de Nantes (IRIN)

Université de Nantes

2, rue de la Houssinière - BP 92208

44322 Nantes Cedex 03 - France

{morin,martien}@irin.univ-nantes.fr

Abstract. Acquisition of patterns for information extraction systems is a common task in Natural Language Processing, mostly based on manual analysis of text corpora. We have developed a system called PROMÉTHÉE, which incrementally extracts lexico-syntactic patterns for a specific conceptual relation from a technical corpus. However, these patterns are often too general and need to be manually validated.

In this paper, we demonstrate how PROMÉTHÉE has been interfaced with the machine learning system EAGLE in order to automatically refine the patterns it produces. The empirical results obtained with this technique show that the refined patterns allows to decrease the need for the human validation.

1 Introduction

As the amount of electronic documents (corpora, dictionaries, newspapers, news-wires, etc.) become more and more important and diversified, there is a need to extract information automatically from texts. Extracting information from text is an important task for Natural Language Processing researchers. In contrast to text understanding, information extraction systems do not aim at making sense of the entire text, but are only focused on fractions of the text that are relevant to a specific domain [6]. In information extraction, the data to be extracted from a text is given by a *syntactic pattern*, also called a *template*, which typically involves recognizing a group of entities, generally noun phrases, and some relationships between these entities.

In recent years, through Message Understanding Conferences, several information extraction systems have been developed for a variety of domains. However, many of the best-performing systems are difficult and time-consuming to build. They also generally contain domain-specific components. Therefore, their success is often tempered by their difficulties to adapt to new domains. Having the use of specialists' abilities for each domain is not reasonable.

^{*} We would like to thank C. Jacquemin and M. Quafafou for helpful discussions on this work.

In order to overcome such weakness, we have developed the PROMÉTHÉE system, dedicated to the extraction of lexico-syntactic patterns relative to a specific conceptual relation, from a technical corpus [10]. However, based on our experience, we believe that such patterns are too general: indeed, without using manual constraints, their coverage is satisfying but their precision¹ is low. In order to refine these patterns, we propose to use a learning system, called EAGLE [8], which is based on the *Inductive Logic Programming* paradigm [11]. This latter extracts intensional descriptions of concepts, from their extensional descriptions including their ground examples and counter-examples, as well as a prior knowledge of the domain. The learned definitions, expressed in a logic-based formalism, are further used in recognition or classification tasks.

This paper is organized as follows. Section 2 presents a description of the information extraction system PROMÉTHÉE. Next, section 3 presents the interfacing between the PROMÉTHÉE and EAGLE systems. Section 4 presents and evaluates some results obtained on some patterns of the hyponymy relation. Section 5 discusses related work in applying symbolic machine learning to information extraction. Finally, section 6 concludes the paper and suggests future work.

2 The Prométhée System

In the last few years, several information extraction systems have been developed to extract patterns from text. AutoSlog [13,14] creates a dictionary of extraction patterns by specializing a set of general syntactic patterns. CRYSTAL [15] is another system that generates extraction patterns dependant on domain-specific annotations. LIEP [7] also learns extraction patterns, but relies on predefined keywords, a sentence analyzer to identify noun and verb groups, and an entity recognizer to identify entities of interest (people, company names, and management titles).

Our approach to extract patterns is based on a different technique which makes no hypothesis about the data to be extracted. The information extraction system PROMÉTHÉE uses only pairs of terms linked by the target relation to extract specific patterns, but relies on part-of-speech tag, and on local grammars. For instance, the following sentence of the [MEDIC] corpus²: *we measured the levels of aspartate, glutamate, gamma-aminobutyric acid, and other amino acids in autopsied brain of 6 patients* contains a pair of terms, namely (*aspartate*, *amino*

¹ The precision of a pattern is the percentage of sentences matching the pattern which really denote the conceptual relation modeled by this pattern.

² All the experiments reported in this paper have been performed on [AGRO]: a 1.3-million words French agronomy corpus and on [MEDIC]: a 1.56-million words English medical corpus. These corpus are composed of abstracts of scientific papers owned by INIST-CNRS.

acid), linked by the hyponymy³ relation. From this sentence, the following pattern modeling the relation is extracted: NP {, NP}* and other NP⁴.

2.1 Overview of the Prométhée Architecture

The PROMÉTHÉE architecture is divided into three main modules:

1. *Lexical Preprocessor*. This module starts by reading the raw text. The text is divided into sentences which are individually tagged⁵, *i.e.* noun phrases, acronyms, and a succession of noun phrases are detected by using regular expressions. The output is formatted under the SGML (Standard Generalized Markup Language) formalism.
2. *Lexico-syntactic Analyzer*. This module extracts lexico-syntactic patterns modeling a semantic relation from the SGML corpus. Patterns are discovered by looking through the corpus, and by using a bootstrap of pairs of terms linked by the target relation. This procedure which consists of 7 steps is described in the next section.
3. *Conceptually Relationship Extractor*. This module extracts pairs of conceptually related terms by using a database of patterns, which can be either the output of the lexico-syntactic analyzer or manually specified patterns.

2.2 Lexico-syntactic Analyzer

The lexico-syntactic analyser extracts new patterns by looking through a SGML corpus. This procedure, inspired by Hearst [4,5], is composed of 7 steps.

1. Select manually a representative conceptual relation, *e.g.* the hyponymy relation.
2. Collect a list of pairs of terms linked by the previous relation. This list of pairs of terms can be extracted from a thesaurus, a knowledge base or manually specified. For example, from a medical thesaurus and the hyponymy relation, we find that *glutamate* IS-A *amino acid*.
3. Find sentences where conceptually related terms occur. Thus, the pair (*glutamate, amino acid*) allows to extract from the corpus [MEDIC] the sentence: *we measured the levels of asparate, glutamate, gamma-aminobutyric acid, and other amino acids in autopsied brain of 6 patients.*
4. Find a common environment that generalizes the sentences extracted at the third step. This environment indicates a candidate lexico-syntactic pattern.
5. Validate candidate lexico-syntactic patterns by an expert.
6. Use new patterns to extract more pairs of candidate terms.
7. Validate candidate terms by an expert, and go to step 3.

³ According to [9], a lexical term L_0 is said to be a hyponym of the concept represented by a lexical item L_1 if native speakers of English accept sentences constructed from the frame *An L_0 is a (kind of) L_1* . Here, L_0 (resp. L_1) is the hyponym (resp. hypernym) of L_1 (resp. L_0).

⁴ NP is part of speech tag for a noun phrase.

⁵ We thanks Évelyne Tzoukermann (Bell Laboratories, Lucent Technologies) for having tagged and lemmatized the corpus [AGRO].

2.3 Lexico-syntactic Expressions and Patterns

At the third step of the lexico-syntactic analyzer, a set of sentences is extracted. These sentences are lemmatised, and noun phrases are identified. So, we represent a sentence by a lexico-syntactic expression. For instance, the following element of the hyponymy relation: (*neocortex, vulnerable area*) allows to extract from the corpus [MEDIC] the sentence: *Neuronal damage were found in the selectively vulnerable areas such as neocortex, striatum, hippocampus and thalamus*. From this sentence, we produce the lexico-syntactic expression: NP be find in NP such as LIST⁶.

A lexico-syntactic expression is composed of a set of elements, which can be either lemmas, punctuation marks, numbers, symbols (*e.g.* §, <, π , etc.) or words with specific part of speech tags, such as NP, LIST, CRD, etc. Through this simplification process, we have a more generic representation of relevant sentences, and comparing these sentences is easier.

A lexico-syntactic pattern is a generalization of a set of lexico-syntactic expressions. For example, with the previous expression, and at least another similar one, the following lexico-syntactic pattern is deduced [10] : NP such as LIST.

2.4 Limitations of this Technique

Using this technique, some lexico-syntactic patterns are extracted. However, these patterns are too general: indeed without using manual constraints, their coverage is satisfying but their precision is low. The low precision can be explain by general patterns which cover a set of more rarely specific patterns. Too general patterns do not prevent the further extraction of pairs of terms which are not linked by the target relation. At present, a human validation (the step 5 of the lexico-syntactic analyzer procedure) is necessary to exclude the patterns which are considered as too general. Through the interfacing of PROMÉTHÉE and EAGLE, we aim at automatically acquiring some knowledge refining these patterns, in order to decrease the need of human validation.

3 Interfacing Prométhée with Eagle

The goal of interfacing PROMÉTHÉE with EAGLE is to use the latter as a tool for refining too general patterns. Thus, EAGLE fits between the steps 5 and 6 of the previous methodology (see Section 2.2).

For a specific pattern, the lexico-syntactic analyzer extracts sentences from the SGML corpus. An expert classifies these sentences between examples (*i.e.* sentences where pairs of terms are conceptually related) and counter-examples (*i.e.* sentences where pairs of terms are not conceptually related). From this extensional description of the patterns and the prior knowledge consisting of a lexicon, the EAGLE system extracts some intensional descriptions of these patterns. Interpreted as syntactic or logic constraints on the general form of the

⁶ LIST is part of speech tag for a succession of noun phrases.

patterns, these descriptions allow to refine them and to decrease the need for human validation.

Interfacing the two systems requires the translation of PROMÉTHÉE’s lexico-syntactic analyzer output sentences into EAGLE’s logic-based formalism. Here, a sentence is basically viewed as a lexico-syntactic expression including two main conceptually related noun phrases called NP1 and NP2. In EAGLE, the representation of such a sentence in the prior knowledge consists in describing, by means of predicates, how it is organized around NP1 and NP2, *i.e.* which terms precede or follow them, together with the corresponding separation depths. Given a noun phrase and a particular element in the sentence, the depth is defined here as the distance, *i.e.* the number of elements, which separate the noun phrases from the given element. Additional predicates are used in the prior knowledge to indicate the part of speech tags (verb, adjective, etc) of the terms in the lexicon.

4 Experimental Results

In this experimentation, we have focused on the hyponymy relation. For this relation, PROMÉTHÉE incrementally extracted 11 lexico-syntactic patterns from the corpus [AGRO]. We are particularly interested in two of them, namely: NP comme LIST (NP such as LIST in English), and NP (LIST), which model respectively exemplification and enumeration structures [2]. Some sentences instantiating these patterns were produced from a 43,000 sentences corpus [AGRO], and split into examples and counter-examples. The following clause $\text{Pattern}(x) \leftarrow \text{Succ}(x, \text{NP1}, y, z) \wedge \text{Crđ}(y)$ is an example of the results produced by EAGLE. It defines a constraint according to which a pattern x models an hyponymy relation if (1) its noun phrase NP1 is followed by a term y at a depth equal to z , and (2) y is a cardinal number.

4.1 Exemplification Structure Pattern

Among the 36 sentences instantiating the pattern NP comme LIST, the expert retained a sample of 28 sentences which denoted a hyponymy relation, *i.e.* the examples, and 8 sentences which did not, *i.e.* the counter-examples. In a first experimentation, constraints were induced by using the whole prior knowledge associated with the 36 sentences. But the resulting constraints were not satisfying in the sense that they focused on tool words (*e.g.* preposition, article, etc.). In order to improve the results, some predicates regarding tool words have been ignored from the prior knowledge. The constraints which were learned from the next experimentation can be split into two main categories: (1) the hyperonym term can be preceded by an undefined adjective, such as *différents* (*different*), *certain*s (*some*) and *d’autres* (*others*), and (2) the hyperonym term can be preceded by the expression *chez d’autres*. It appears that sentences matching these constraints have a high level of reliability, and do not require validation by a expert. This is illustrated on Table 1.

Before learning, the pattern NP comme LIST is too general, since its precision is equal to 77.7%. As a consequence, all the 36 matching sentences must be

Table 1. Exemplification structure patterns accuracies before and after learning process

	Pattern	Matching sent.	Good sent.	False sent.
Before learning	NP comme LIST	36	28	8
After learning	chez d'autres NP comme LIST	2	2	0
	{certains différents d'autres ...} NP comme LIST	8	8	0
	NP comme LIST	26	18	8

manually validated. After learning, two patterns have a precision of 100.0%, which allows to remove the matching sentences from the manual validation. Consequently, only 26 matching sentences must be manually validated. With these new constraints, around 27% ($100 - (26/36) * 100$) of matching sentences are automatically acquired.

4.2 Enumeration Structure Pattern

Among the 603 sentences instantiating the pattern NP (LIST), the expert retained a sample of 21 sentences which denoted a hyponymy relation, *i.e.* the examples, and 16 sentences which did not, *i.e.* the counter-examples. As in the previous experimentation, some restrictions have been applied in the prior knowledge. Here, two categories of constraints have been acquired: (1) as previously the hyperonym term can be preceded by an undefined adjective, and (2) the cardinal before the hyperonym term must be equal to the number of elements of the list LIST. This is illustrated on Table 2.

Before learning the precision of the pattern NP (LIST) is equal to 56.8% on 37 matching sentences. Once again, learning allows to decrease the number of matching sentences to be manually validated (*i.e.* 27 *vs* 37). Again, with these specific constraints, around 27% ($100 - (27/37) * 100$) of matching sentences are automatically acquired.

Table 2. Enumeration structure patterns accuracies before and after learning process

	Pattern	Matching sent.	Good sent.	False sent.
Before learning	NP (LIST)	37	21	16
After learning	NP (LIST)	27	11	16
	{certains différents d'autres ...} NP (LIST)	4	4	0
	CRD1 NP (LIST-CRD2)	6	6	0
	CRD1 = CRD2			

5 Related Work

Previous research involving Machine Learning methods and Natural Language Processing has been devoted to the learning of syntactic patterns, such as noun phrases [12,1], name phrases [16], or specific-domain patterns [15,13,14,7,3]. Machine learning has the potential to significantly assist the acquisition of lexico-syntactic patterns.

Several information extraction systems, dedicated to the acquisition of patterns, are based on the use of machine learning techniques. AUTOSLOG [13] system uses a training corpus to generate candidate patterns, and rely on an expert to verify and reject each candidate pattern. CRYSTAL [15] is one of the first systems to automatically induce a dictionary of information extraction rules, by generalizing patterns identified in the text by an expert. However, a training corpus is not often available for most information extraction tasks. The RAPIER [3] system uses relational learning to construct unbounded pattern-match rules. LIEP [7] learns information extraction patterns from example texts containing events. A user can choose which combinations of entities signify events to be extracted. These positive examples are used by LIEP to build a set of extraction patterns. The general methodology is similar to EAGLE's, but PROMÉTHÉE, like AUTOSLOG, does not try to recognize relationships between multiple constituents.

EAGLE system is used by the PROMÉTHÉE system only to provide more information about the general forms of the patterns. Thus, it is involved only in a small part of the acquisition process. Consequently, few training examples are needed to produce syntactical constraints : around forties are enough to achieve good performance, rather than hundreds or thousands. Moreover, the constraints produced by EAGLE provide some readable logical and syntactical information about lexico-syntactic-patterns. This is not the case of other systems only extract syntactical information.

6 Conclusion and Future Work

In this paper, we have proposed an approach for refining lexico-syntactic patterns, based on the use of a machine learning tool. This technique interfaces an information extraction system PROMÉTHÉE with an inductive logic programming system EAGLE, which allows for refining the lexico-syntactic patterns produced by PROMÉTHÉE.

The empirical results obtained with this technique show that the refined patterns allows to decrease the need for the human validation.

From a Natural Language Processing point of view, the use of a machine learning technique highlights some knowledge which usually required manual data mining. From a Machine Learning point of view, it illustrates the usefulness of an inductive learning technique on a real-world problem.

In future work, we plan to investigate the usefulness of EAGLE to extract constraints by using PROMÉTHÉE's syntactical and morphological information which allowed to generate lexico-syntactic expressions.

References

1. Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. A memory-based approach to learning shallow natural language patterns. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, Montreal, Canada, 1998. 298
2. Andr  e Borillo. Exploration automatis  e de textes de sp  cialit   : rep  rage et identification de la relation lexicale d'hyponymie. *LINX*, 34/35:113–124, 1996. 296
3. Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Computational Natural Language Learning (CoNLL'97)*, pages 9–15, Madrid, Spain, July 1997. 298
4. Marti A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 539–545, Nantes, France, 1992. 294
5. Marti A. Hearst. Automated Discovery of WordNet Relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 131–151. MIT Press, Cambridge, MA, 1998. 294
6. Jerry R. Hobbs, Douglas E. Appelt, John S. Bear, David J. Israel, and W. Mabry Tyson. FASTUS : A system for extracting information from natural language text. Technical Report 515, SRI International, Menlo Park, CA, november 1992. 292
7. Scott B. Huffman. Learning information extraction patterns from examples. In *Workshop New Approaches to Learning for Natural Language Processing at IJCAI'95*, pages 127–133, Montreal, 1995. 293, 298
8. Emmanuelle Martienne and Mohamed Quafafou. Learning Logical Descriptions for Document Understanding: a Rough Sets-based Approach. In *Proceedings of the first International Conference on Rough Sets and Current Trends in Computing (RSCTC'98)*, pages 22–26, Warsaw, Pologne, june 1998. 293
9. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Miller Katherine. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3:235–244, 1990. 294
10. Emmanuel Morin. *Extraction de liens s  mantiques entre termes    partir de corpus de textes techniques*. Th  se en informatique, Universit   de Nantes, December 1999. 293, 295
11. Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8:295–318, 1991. 293
12. Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 811–816, 1995. 298
13. Ellen Riloff. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 811–816, Menlo Park, CA, USA, July 1993. 293, 298
14. Ellen Riloff. Automatically generating extraction from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 1044–1049, august 1996. 293, 298
15. Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1314–1319, august 1995. 293, 298
16. Marc Vilain and David Day. Finite-state phrase parsing by rule sequences. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 274–279, Copenhagen, Denmark, 1996. 298

Measuring Performance when Positives Are Rare: Relative Advantage versus Predictive Accuracy – A Biological Case-Study

Stephen H. Muggleton¹, Christopher H. Bryant¹, and Ashwin Srinivasan²

¹ Computer Science, University of York
York YO10 5DD, UK

{stephen,bryant}@cs.york.ac.uk
<http://www.cs.york.ac.uk/aig/>

² Computing Laboratory, Oxford University
Wolfson Building, Oxford OX1 3QD, UK
ashwin@comlab.oxford.ac.uk

Abstract. This paper presents a new method of measuring performance when positives are rare and investigates whether Chomsky-like grammar representations are useful for learning accurate comprehensible predictors of members of biological sequence families. The positive-only learning framework of the Inductive Logic Programming (ILP) system CProgol is used to generate a grammar for recognising a class of proteins known as human neuropeptide precursors (NPPs). Performance is measured using both predictive accuracy and a new cost function, *Relative Advantage* (*RA*). The *RA* results show that searching for NPPs by using our best NPP predictor as a filter is more than 100 times more efficient than randomly selecting proteins for synthesis and testing them for biological activity. Predictive accuracy is not a good measure of performance for this domain because it does not discriminate well between NPP recognition models: despite covering varying numbers of (the rare) positives, all the models are awarded a similar (high) score by predictive accuracy because they all exclude most of the abundant negatives.

1 Introduction

This paper presents a new method of measuring performance when positives are rare and attempts to answer, by way of a case-study, the question of whether grammatical representations are useful for learning from biological sequence data. We address the question by refuting the following null hypothesis.

Null hypothesis: The most accurate comprehensible multi-strategy predictors of biological sequence families do not employ Chomsky-like grammar representations.

The performance of each model is measured using a new cost function, *Relative Advantage* (*RA*). Section 2 defines *RA* and explains why it is used in preference to predictive accuracy.

The domain of the case study is the recognition of a class of proteins known as human neuropeptide precursors (NPPs). These proteins have considerable therapeutic potential and are of widespread interest in the pharmaceutical industry. Our most accurate comprehensible multi-strategy predictor of NPPs employs a Chomsky-like grammar representation.

Multi-strategy learning [4] aims at integrating multiple strategies in a single learning system, where strategies may be inferential (e.g. induction, deduction etc) or computational. Computational strategy is defined by the representational system and the computational method used in the learning system (e.g. decision tree learning, neural network learning etc).

We refute the null hypothesis as follows. A grammar is generated for a particular class of biological sequences. A group of features is derived from this grammar. Other groups of features are derived using other learning strategies. Amalgams of these groups are formed. A recognition model is generated for each amalgam using C4.5 and C4.5rules. The null hypothesis is refuted because:–

1. the best performance achieved using any of the models which include grammar-derived features is higher than the best performance achieved using any of the models which do not include the grammar-derived features;
2. this increase is statistically significant;
3. the best model which includes grammar-derived features is sufficiently more comprehensible than the best ‘non-grammar’ model.

2 Relative Advantage

NPPs are identified either through purely biological means or by screening genomic or protein sequence databases for likely NPPs, followed by biological evaluation. If we wish to go beyond using sequence homology to find new members of the (generally small) NPP families, we need a recognition model for NPPs in general. However if this recognition model is poor then it may not be much better than random sampling of sequence databases and the cost-benefit of any experimental evaluation of NPPs found by such a procedure would be prohibitively small.

In developing a general recognition model for human NPPs, we are faced with three significant obstacles.

1. The number of known NPPs in the public domain databases of protein sequence (e.g. SWISS-PROT [3]) is very small in proportion to the total number of sequences. When we developed our method of estimating *RA* (May 1999), SWISS-PROT contained 79,449 sequences, of which some 57 could definitely be identified as human NPPs.
2. There is no guarantee that all the human NPPs in SWISS-PROT have been properly identified. We estimate there may, in fact be up to 90 NPPs in SWISS-PROT.

3. There is no benchmark method for NPP recognition that can be used to compare any new methods. We must therefore compare our recognition model with random sampling to evaluate success.

This domain requires a performance measure which addresses all of these issues. For domains in which positives are rare, predictive accuracy, as it is normally measured in Machine Learning (assuming equal misclassification costs):–

- gives a poor estimate of the performance of a recognition model. For instance, if a learner induces a very specific model for such a domain, the predictive accuracy of the model may be very high despite the number of true positives being very small or even zero.
- does not discriminate well between models which exclude most of the (abundant) negatives but cover varying numbers of (the rare) positives. (This is illustrated later in this paper – see Table 4.)

For domains in which there is no guarantee that all positives can be identified as such, assigning misclassification costs does not suffice (see Sect. 2.1).

Therefore we define a *relative advantage* (RA) function which predicts the reduction in cost in using the model versus random sampling. (In the following, ‘the model’ refers to a recognition model for predicting whether a sequence is a NPP.) $RA = \frac{A}{B}$ where

A = the expected cost of finding one NPP by repeated independent random sampling from SWISS-PROT and performing a laboratory analysis of each protein.

B = the expected cost of finding one NPP by repeated independent random sampling from SWISS-PROT and analysing only those proteins which are predicted by the learned model to be a NPP.

In contrast to other measures of performance, this ratio is both relevant and meaningful to experts in the domain.

RA can be defined in terms of probability as follows. Let C = the cost of testing the biological activity of one protein via wet-experiments in the laboratory; NPP = Sequence is a NPP; Rec = Model recognises sequence as a NPP.

$$RA = \frac{C/Pr(NPP)}{C/Pr(NPP | Rec)} = \frac{Pr(NPP | Rec)}{Pr(NPP)} \quad (1)$$

Let testing the model on test data yield the 2×2 contingency table shown in Table 1a with the cells n_1 , n_2 , n_3 , and n_4 . Let $n = n_1 + n_2 + n_3 + n_4$ be the number of instances in the test-set.

If the proportion of NPPs in the test-set was known to be the same as the proportion of NPPs in the database then we could estimate $Pr(NPP)$ to be $(n_1 + n_3)/n$ and $Pr(NPP | Rec)$ to be $n_1/(n_1 + n_2)$. These estimates cannot be used with our method because we cannot assume that the proportion of NPPs is the same in the test-set and database.

In order to derive a formula for estimating RA given both a set of positives and a set of randoms, we estimate $Pr(NPP)$ and $Pr(NPP | Rec)$ as follows.

Table 1. 2×2 Contingency table for a) the test-set and b) SWISS-PROT. The axes of each 2×2 matrix are labelled by the sets NPP sequences, Random sequences, H (Hypothesis predictions) and $\overline{H}$ (complement of H). The cells of each matrix represent the cardinalities of the corresponding intersections of these sets. $n_1 + n_2 + n_3 + n_4 = n$, where n is the number of instances in the test-set. The total of the counts/frequencies in the four cells of the contingency table for SWISS-PROT = S , where S is the total number of sequences in the SWISS-PROT database.

a) test-set			b) SWISS-PROT	
	Set of test NPP seq.s	Set of test Random seq.s	NPP sequences in SWISS-PROT	Random sequences in SWISS-PROT
H	n_1	n_2	$\left(\frac{n_1}{n_1+n_3}\right) M$	$\left(\frac{n_2}{n_2+n_4}\right) (S - M)$
$\overline{H}$	n_3	n_4	$\left(\frac{n_3}{n_1+n_3}\right) M$	$\left(\frac{n_4}{n_2+n_4}\right) (S - M)$

Let S be the total number of sequences in the database, of which M are NPPs.

$$Pr(NPP) = \frac{\text{no. of NPPs in the database}}{\text{no. of sequences in the database}} = M/S \quad (2)$$

$$Pr(NPP \mid Rec) = F/G \quad (3)$$

where F = no. of NPPs in db which are recognised by model
and G = no. of sequences in db which model predicts to be NPP.

Table 1b shows the expected result of using the learned recognition model on the entire SWISS-PROT database. From Equation 3 and Table 1b it follows that:

$$Pr(NPP \mid Rec) \simeq \frac{\left(\frac{n_1}{n_1+n_3}\right) \times M}{\left(\frac{n_1}{n_1+n_3}\right) M + \left(\frac{n_2}{n_2+n_4}\right) (S - M)} = (Mp_1)/(Mp_1 + (S - M)p_2) \quad (4)$$

where $p_1 = n_1/(n_1 + n_3)$ and $p_2 = n_2/(n_2 + n_4)$. Substituting Equations 2 and 4 into Equation 1 gives

$$RA = \frac{(Mp_1)/(Mp_1 + (S - M)p_2)}{M/S} = \frac{Sp_1}{Sp_2 + M(p_1 - p_2)} \quad (5)$$

2.1 Estimating Relative Advantage

In the following Relative Advantage over the entire population is represented by RA in capital letters where as Relative Advantage over a sample is denoted by lower case i.e. ra . As the value of M is not known, we estimate $\sum_{M=57}^{90} RA$. Therefore we integrate Equation 5 with respect to M . The lower limit of M is equal to the number of known NPPs in SWISS-PROT. The upper limit of M is

the most probable maximum number of NPPs in SWISS-PROT i.e. a total of the known NPPs and those proteins which have yet to be scientifically recognised as a NPP.

$$\begin{aligned} \sum_{M=57}^{90} RA &\simeq Sp_1 \times \int_{M=57}^{90} \frac{1}{(p_1 - p_2)M + Sp_2} \partial M + RA(57) \\ &= \frac{Sp_1}{(p_1 - p_2)} \ln \frac{90(p_1 - p_2) + Sp_2}{57(p_1 - p_2) + Sp_2} \end{aligned} \quad (6)$$

We estimate $\sum_{M=57}^{90} RA$ by summing an estimate of the $\sum_{M=57}^{90} RA$ for each instance in the test-set as follows, where n is the number of instances in the test-set. This method has the advantage that it allows the significance of the difference between the RA of two models to be gauged (see Sect. 2.2). From the contingency table it follows that:

$$\sum_{M=57}^{90} ra = \frac{1}{n} \sum_{i=1}^4 \left(n_i \sum_{M=57}^{90} ra_i \right) \quad (7)$$

Each $\sum_{M=57}^{90} ra_i$ is estimated by substituting $p_1 = \frac{a}{a+c}$ and $p_2 = \frac{b}{b+d}$ into Equation 6. The values of a , b , c and d are determined by three steps.

1. Whatever the i value, a , b , c and d are initially given the values of the corresponding counts/frequencies in the contingency table for the test-set (see Table 1a).
2. Each one of a , b , c and d , is decremented providing that the value before subtraction is greater than 1.
We do not decrement when the value before subtraction is zero because this can result in p_1 or p_2 having negative values; this does not make sense because p_1 and p_2 are probabilities. We do not decrement when the value is one because this can cause p_1 or p_2 to have the value zero, which in turn has a *highly disproportionate* effect on the value of $\sum_{M=57}^{90} ra_i$.
3. The value of either a , b , c or d is incremented to reflect the classification of an instance in the cell n_i .

For instance, if $i = 2$ and all the counts in the contingency table are greater than one then $a = n_1 - 1$, $b = n_2$, $c = n_3 - 1$, $d = n_4 - 1$.

Note that Steps 1 and 2 assign the same prior probability to each instance because the effect of each step is not dependent upon which cell the current instance belongs to. Therefore this method of estimating $\sum_{M=57}^{90} RA$ has the properties of a) producing identically distributed random variables representing the outcome for each instance; b) having a sample mean which approaches the population mean in the limit and c) having a relatively small sample variance.

The final step of our method for estimating RA is to take the mean of the summed values.

$$Mean\ RA = \frac{\sum_{M=57}^{90} ra_i}{90 - (57 - 1)} = \frac{\sum_{M=57}^{90} ra_i}{34} \quad (8)$$

Table 2. 4×4 Contingency Table. The rows of the 4×4 matrix are labelled by the cells of the 2×2 contingency table for H_1 . The columns of the 4×4 matrix are labelled by the cells of the 2×2 contingency table for H_2 . The cells of the 4×4 matrix represent the cardinalities of the corresponding intersections of these sets. $\sum_{i=1}^4 \sum_{j=1}^4 n_{i,j} = n$, where n is the number of instances in the test-set

	n_1	n_2	n_3	n_4
n_1	$n_{1,1}$	$n_{1,2}$	$n_{1,3}$	$n_{1,4}$
n_2	$n_{2,1}$	$n_{2,2}$	$n_{2,3}$	$n_{2,4}$
n_3	$n_{3,1}$	$n_{3,2}$	$n_{3,3}$	$n_{3,4}$
n_4	$n_{4,1}$	$n_{4,2}$	$n_{4,3}$	$n_{4,4}$

2.2 Assessing the Significance of the Difference between the RA of Two Models

We compare the performance of two recognition models, H_1 and H_2 , by comparing their $\sum_{M=57}^{90} RA$ values. Let d be difference in $\sum_{M=57}^{90} RA$ values over the entire population, i.e. for all the proteins in SWISS-PROT, and $\hat{d}$ be the observed difference on the test-set.

$$d = \sum_{M=57}^{90} RA_{H_1} - \sum_{M=57}^{90} RA_{H_2} \quad (9)$$

$$\hat{d} = \sum_{M=57}^{90} ra_{H_1} - \sum_{M=57}^{90} ra_{H_2} \quad (10)$$

$\hat{d}$ is an unbiased estimator for the true difference because it is calculated using an independent test set. To determine whether the observed difference is statistically significant we address the following question. What is the probability that $\sum_{M=57}^{90} RA_{H_1} > \sum_{M=57}^{90} RA_{H_2}$, given the observed difference, $\hat{d}$.

If D is a random variable representing the outcome of estimating d by random sampling then, according to the Central Limit Theorem, $\hat{\mu}_D$ is normally distributed in the limit. It has an estimated mean $\hat{d}$ and has an estimated variance of $\hat{\sigma}_D^2/n$. The variance of a random variable, X , is $\sigma_X^2 = E((X)^2) - (E(X))^2$. Therefore, since D is a random variable: $\hat{\sigma}_D^2 = \hat{\mu}_{D^2} - \hat{\mu}_D^2$. We calculate $\hat{\mu}_{D^2}$ as follows. Let testing the model on test data yield the 4×4 contingency table shown in Table 2 with the cells $n_{i,j}$. (Note that only those cells shown in bold font can have a count greater than zero because an instance cannot be both an NPP and a Random.)

$$\hat{\mu}_{D^2} = \frac{1}{n} \sum_{i=1}^4 \sum_{j=1}^4 \left(n_{i,j} \left(\sum_{M=57}^{90} ra_i - \sum_{M=57}^{90} ra_j \right)^2 \right) \quad (11)$$

Given that $p(\sum_{M=57}^{90} RA_{H_1} > \sum_{M=57}^{90} RA_{H_2}) = p(\sum_{M=57}^{90} RA_{H_1} - \sum_{M=57}^{90} RA_{H_2} > 0)$ we evaluate our null hypothesis by estimating $p(d < 0)$ using the Central

Limit Theorem.

$$\int_{x=-\infty}^0 Pr(d=x)dx = \int_{x=-\infty}^0 \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \quad (12)$$

where $\mu = \hat{\mu}_D$ and $\sigma = \hat{\sigma}_D/\sqrt{n}$.

3 Sequence Data in Biology

Research in the biological and medical sciences is being transformed by the volume of data coming from projects which will reveal the entire genetic code (genome sequence) of *Homo sapiens* as well as other organisms that help us understand the genetic basis of human disease. A significant challenge in the analysis and interpretation of genetic sequence data is the accurate recognition of patterns that are diagnostic for known structural or functional features within the protein. Although regular expressions can describe many of these features they have some inherent limitations as a representation of biological sequence patterns. In recent years attention has shifted towards both the use of neural network approaches (see [1]) and to probabilistic models, in particular hidden Markov models (see [2]). Unfortunately, due to the complexity of the biological signals, considerable expertise is often required to 1) select the optimal neural network architecture or hidden Markov model prior to training and 2) understand the biological relevance of detailed features of the model.

A general linguistic approach to representing the structure and function of genes and proteins has intrinsic appeal as an alternative approach to probabilistic methods because of the declarative and hierarchical nature of grammars. While linguistic methods have provided some interesting results in the recognition of complex biological signals [9] general methods for learning new grammars from example sentences are much less developed.

We considered it valuable to investigate the application of Inductive Logic Programming methods to the discovery of a language that would describe a particularly interesting class of sequences – neuropeptide precursor proteins (NPP). Unlike enzymes and other structural proteins, NPPs tend to show a lower overall sequence similarity despite some evidence of common ancestry within certain groups. This confounds pattern discovery methods that rely on multiple sequence alignment and recognition of biological conservation. Our approach is to generate the context-free definite-clause-grammar shown in Fig. 1. We represent protein sequences using the alphabet $\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$, where each letter represents a particular amino acid residue.

4 Experiment

This section describes an experiment which tries to refute the null hypothesis (see Sect.1). It describes the materials used in the experiment and the three steps of the experimental method and presents the results.

4.1 Materials

Data was taken from the annotated protein sequence database SWISS-PROT. Our data-set¹ comprises a subset of positives i.e. known NPPs and a subset of randomly-selected sequences. It is not possible to identify a set of negative examples of NPPs with certainty because there will be proteins which have yet to be recognised scientifically as a NPP. The subset of positives contains all of the 44 known NPP sequences that were in SWISS-PROT at the time the data-set was prepared. 10 of the 44 precursors were reserved for the test set. These sequences are unrelated by sequence similarity to the remaining 34. The subset of randoms contains all of the 3910 full length human sequences in SWISS-PROT at the time the data-set was prepared. 1000 of the 3910 randoms were reserved for the test-set.

4.2 Method

The method may be summarised as follows. A grammar is generated for NPP sequences using CProlog [5] version 4.4 (see Sect.3). A group of features is derived from this grammar; other groups of features are derived using other learning strategies (see Sect.3). Amalgams of these groups are formed. A rule-set is generated for each amalgam using C4.5 (Release 8) [8] and C4.5rules² and its performance is measured using *MeanRA* (see Sect.3). The null-hypothesis (see Sect.1) is then tested by comparing the *MeanRA* achieved from the various amalgams.

During both the generation of the grammar using CProlog and the generation of propositional rule-sets using C4.5 and C4.5rules we adopt the background information used in [6] to describe physical and chemical properties of the amino acids.

Table 3 summarises how some of the properties SWISS-PROT changed over the duration of the experiments described in this paper and the subsequent preparation of this paper. All the *MeanRA* measurements in this paper are based on the properties as they stood at May 99; these were the most up-to-date values available at the time the measurements were made.³

Grammar Generation. A *NPP grammar* contains rules that describe legal neuropeptide precursors. Fig. 1 shows an example of such a grammar, written as a Prolog program. This section describes how production rules for signal peptides and neuropeptide starts, middle-sections and ends were generated using CProlog. These were used to complete the context-free definite-clause-grammar structure shown in Fig. 1. CProlog was used because it is the only general purpose ML system which is both capable of generating a grammar and tolerant to the

¹ Available at <ftp://ftp.cs.york.ac.uk/pub/aig/Datasets/neuropeps/>.

² The default settings of C4.5 and C4.5rules were used.

³ When measuring performance using *MeanRA* there is no requirement that the size of the test data-set is equal to the number of known human NPPs in SWISS-PROT.

Table 3. Properties of sequences in SWISS-PROT at the time the data-set described in Sect.4.1 was prepared and in May 1999

	At the time the data- May '99 set was prepared	
Number of sequences	64,000	79,449
Number of known human NPPs	44	57
Most probable maximum number of human NPPs	Not known	90

noise present in real-world data-sets. Approaches which are specific to grammar induction do not tolerate noise.

The grammar to be learnt by CProgol contains dyadic non-terminals of the form $p(X,Y)$, which denote that property p began the sequence X and is followed by a sequence Y . To learn production rules for these non-terminals from the training set, CProgol was provided with the following. 1) Extensional definitions of these non-terminals. 2) Definitions of the non-terminals **star/2** and **run/3**. **star/2** represents some sequence of unnamed residues whose length is not specified. **run/3** represents a run of residues which share a specified property. 3) Production rules for various domain-specific subsequences and patterns. This natural inclusion of existing biochemical knowledge illustrates how the grammar-based approach presents a powerful method for describing NPPs.

Certain restrictions were placed on the length of NPPs, signal peptides and neuropeptides because pilot experiments had shown that they increased the accuracy of the grammar. These constraints only affect the values of features derived from the grammar. They do not constrain the value of the sequence length feature described at the end of Sect.3.

Feature Groups. 1) **The grammar features** Each feature in this group is a prediction about a NPP sequence made by parsing the sequence using the grammar generated by CProgol. 2) **The SIGNALP features** Each feature in this group is a summary of the result of using SIGNALP on a sequence. SIGNALP [7] represents the pre-eminent automated method for predicting the presence and location of signal peptides. 3) **The proportions features** Each feature in this group is a proportion of the number of residues in a given sequence which either are a specific amino-acid or which have a specific physicochemical property of an amino-acid. 4) **The sequence length feature** This feature is the number of residues in the sequence.

Propositional Learning. The training and test data sets for C4.5 were prepared as follows.

1. Recall from Sect.4.1 that our data comprises 44 positives and 3910 randoms. 40 of the 44 positives occur in the set of 3910 randoms. As C4.5 is designed

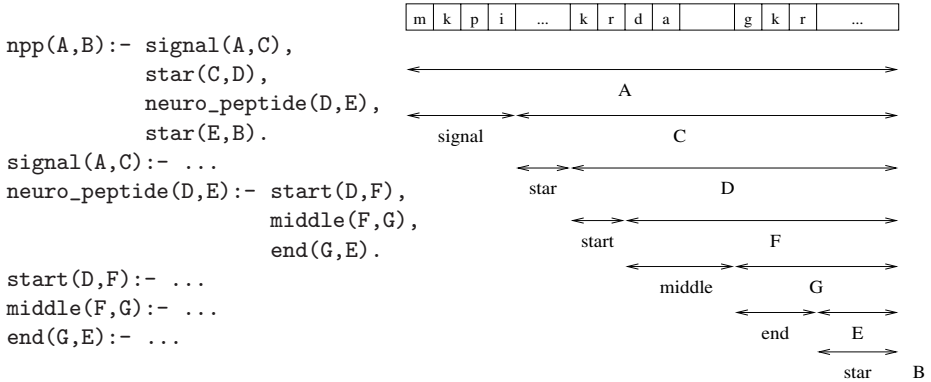


Fig. 1. Grammar rules describing legal NPP sequences. The rules comply with Prolog syntax. *npp*(*X*,*Y*) is true if there is a precursor at the beginning of the sequence *X*, and it is followed by a sequence *Y*. The other dyadic predicates are defined similarly. *star*(*X*,*Y*) is true if, at the beginning of the sequence *X*, there is some sequence of residues whose length is not specified and which is followed by another sequence *Y*. Definitions of the predicates denoted by ‘...’ are to be learnt from data of known NPP sequences

to learn from a set of positives and a set of negatives, these 40 positives were removed from the set of randoms. Of the 40 positives which are in the set of randoms, 10 are in the test-set. Hence the set of $(3910 - 40)$ sequences were split into a training set of $(2910 - 30 = 2880)$ and a test set of $(1000 - 10 = 990)$.

2. Values of the features were generated for each training and test sequence. Each sequence was represented by a data vector comprised of these feature values and a class value (‘1’ to denote a NPP and ‘0’ otherwise).
3. Finally to ensure that there were as many ‘1’ sequences as ‘0’ sequences a *training* set of 2880 NPPs was obtained by sampling with replacement. Thus the training data-set input to C4.5 comprised (2×2880) examples. (No re-adjusting was done on the test data.)

Amalgams of the feature groups described in the previous section were formed. The amalgams are listed in Table 4. The following procedure was followed for each one:– (1) training and test sets were prepared as described above; (2) a decision tree was generated from the training set using C4.5; (3) a rule-set was generated from this tree using C4.5rules; (4) a 2×2 contingency table was drawn-up based on the predictions of this rule-set on the test-set; (5) *MeanRA* was estimated from this contingency table.

The refutation of the null hypothesis was then attempted as described in Sect.1.

Table 4. Estimates of *MeanRA* and predictive accuracy of the amalgams of the feature groups

Amalgam	<i>Mean RA</i>	Predictive Accuracy (%)
Only props	0	96.7 $\pm$ 0.6
Only Length	1.6	91.8 $\pm$ 0.9
Only SignalP	11.7	98.1 $\pm$ 0.4
Only Grammar	10.8	97.0 $\pm$ 0.5
Props + Length	49.0	98.6 $\pm$ 0.4
Props + SignalP	15.0	98.3 $\pm$ 0.4
Props + Grammar	31.7	98.2 $\pm$ 0.4
SignalP + Grammar	0	98.6 $\pm$ 0.4
Length + Grammar	0	96.2 $\pm$ 0.6
Length + SignalP	34.4	98.7 $\pm$ 0.4
Length + SignalP + Grammar	0	98.0 $\pm$ 0.4
Props + Length + SignalP	29.2	98.7 $\pm$ 0.4
Props + Length + Grammar	33.2	98.5 $\pm$ 0.4
Props + SignalP + Grammar	15.0	98.3 $\pm$ 0.4
Props + Length + SignalP + Grammar	107.7	99.0 $\pm$ 0.3

4.3 Results and Analysis

Table 4 shows the *MeanRA* and predictive accuracy for each amalgam of feature groups. The highest *MeanRA* (107.7) was achieved by one of the grammar amalgams, namely the ‘Proportions + Length + SignalP + Grammar’ amalgam. The best *MeanRA* achieved by any of the amalgams which do not include the grammar-derived features was the 49.0 attained by the ‘Proportions + Length’ amalgam. This difference is statistically significant: $p(d < 0)$ is well below 0.0001.

Table 4 shows that predictive accuracy is not a good measure of performance for this domain because it does not discriminate well between the amalgams: despite covering varying numbers of (the rare) positives, all the models are awarded a similar (high) score by predictive accuracy because they all exclude most of the abundant negatives.

5 Discussion

This paper has shown that the most accurate comprehensible multi-strategy predictors of biological sequence families employ Chomsky-like grammar representations.

The positive-only learning framework of the Inductive Logic Programming (ILP) system CProgol was used to generate a grammar for recognising a class of proteins known as human neuropeptide precursors (NPPs). As far as these authors are aware, this is both the first biological grammar learnt using ILP and

the first real-world scientific application of the positive-only learning framework of CProgol.

If one searches for a NPP by randomly selecting sequences from SWISS-PROT for synthesis and subsequent biological testing then, at most, only one in every 2408 sequences tested is expected to be a novel NPP. Using our best recognition model as a filter makes the search for a NPP far more efficient. Approximately one in every 22 of the randomly selected SWISS-PROT sequences which pass through our filter is expected to be a novel NPP.

The best ‘non-grammar’ recognition model does not provide any biological insight. However the best recognition model which includes grammar-derived features is broadly comprehensible and contains some intriguing associations that may warrant further analysis. This model is being evaluated as an extension to existing methods used in SmithKline Beecham for the selection of potential neuropeptides for use in experiments to help elucidate the biological functions of G-protein coupled receptors.

The new cost function presented in this paper, Relative Advantage (RA), may be used to measure performance of a recognition model for any domain where 1) the proportion of positives in the set of examples is very small; 2) there is no benchmark recognition method and 3) there is no guarantee that all positives can be identified as such. In such domains, the proportion of positive examples in the population is not known and a set of negatives cannot identified with complete confidence.

We have developed a general method for assessing the significance of the difference between RA values obtained in comparative trials. RA is estimated by summing the estimate of performance on each test-set instance. The method uses a) identically distributed random variables representing the outcome for each instance; b) a sample mean which approaches the population mean in the limit and c) a relatively small sample variance.

Acknowledgements

This research was conducted as part of a project entitled ‘Using Machine Learning to Discover Diagnostic Sequence Motifs’ supported by a grant from SmithKline Beecham. SmithKline Beecham identified the problem addressed by the case-study, prepared the data-set, and provided expertise on NPPs and general Bioinformatics methods. During part of his time spent writing this article C.H.B. was supported by the EPSRC grant (GR/M56067) entitled ‘Closed Loop Machine Learning’. A.S. holds a Nuffield Trust Research Fellowship at Green College, Oxford.

References

1. P. Baldi and S. Brunak. *Bioinformatics : the machine learning approach*. Massachusetts Institute of Technology Press, Cambridge, Massachusetts, 1998. 306

2. R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, 1998. 306
3. D.B. Emmert, P.J. Stoeck, G. Stoesser, and G.N. Camerson. The European Bioinformatics Institute (EBI) databases. *Nucleic Acids Research*, 22(17):3445–3449, 1994. 301
4. R.S. Michalski and J. Wnek. Guest editors' introduction. *Machine Learning: Second Special Issue on Multistrategy Learning*, 27:205–208, June 1997. 301
5. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995. 307
6. S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7):647–657, 1992. 307
7. H. Nielsen, J. Engelbrecht, S. Brunak, and G. vonHeijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997. 308
8. J.R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993. 307
9. D.B. Searls. Linguistic approaches to biological sequences [review]. *Computer Applications in the Biosciences*, 13:333–344, August 1997. 306

Mining TCP/IP Traffic for Network Intrusion Detection by Using a Distributed Genetic Algorithm

Filippo Neri

DSTA - University of Piemonte Orientale
Corso Borsalino 54, 15100 Alessandria (AL), Italy
`neri@di.unito.it`
`neri@mf.n.unipmn.it`

Abstract. The detection of intrusions over computer networks (i.e., network access by non-authorized users) can be cast to the task of detecting anomalous patterns of network traffic. In this case, models of normal traffic have to be determined and compared against the current network traffic. Data mining systems based on Genetic Algorithms can contribute powerful search techniques for the acquisition of patterns of the network traffic from the large amount of data made available by audit tools. We compare models of network traffic acquired by a system based on a distributed genetic algorithm with the ones acquired by a system based on greedy heuristics. Also we discuss representation change of the network data and its impact over the performances of the traffic models. Network data made available from the Information Exploration Shootout project and the 1998 DARPA Intrusion Detection Evaluation have been chosen as experimental testbed.

1 Introduction

The raise in the number of computer break-ins, virtually occurring at any site, determines a strong request for exploiting computer security techniques to protect the site assets. A variety of approaches to intrusion detection do exist [2]. Some of them exploit signatures of known attacks for detecting when an intrusion occurs. They are thus based on a model of virtually all the possible misuses of the resource. The completeness request is actually a major limit of this approach [8].

Another approach to intrusion detection tries to characterize the normal usage of the resources under monitoring. An intrusion is then suspected when a significant shift from the resource's normal usage is detected. This approach seems to be more promising because of its potential ability to detect unknown intrusions [7,3]. However, it also involves major challenges because of the need to acquire a model of the normal use general enough to allow authorized users to work without raising alarms, but specific enough to recognized unauthorized usages [9,4,11].

Our approach follows the last philosophy for detecting intrusion and we describe here how it is possible to learn a model of normal use of a network from logs of the network activity. A distributed genetic algorithm REGAL [5,14] is exploited for mining the network logs searching for interesting traffic patterns.

We are well aware that many aspects of deploying in practice learning system to acquire useful traffic patterns are still open including: selecting or building informative data representations, improving recognition performances (i.e., reducing both the rate of false alarms and of undetected intrusions), representing the traffic models for real world deployment (real-time classification of packets), and dealing with the shift in the patterns of normal use of the resources [10].

We concentrate here on the first two issues and we report our findings concerning the impact of different learning methods and of alternative data representation, with respect to the ones used in previous works, on the detection performances. As learning methods, we exploited two rule based systems: a heuristic one, RIPPER [1], and an evolutive one (based on genetic algorithms), REGAL [5,14]. The first system has been selected because of its previous use [11]; it will thus act as benchmark. The second system has been selected because we believe that its intrinsically stochastic behavior should allow the acquisition of alternative robust and simpler models [14].

In the following, a description of the used learning systems (Section 2) and of the experiments performed in the Information Exploration Shootout (IES) and DARPA contexts (Section 3 and Section 4) are reported. Finally, the conclusions are drawn.

2 The Learning Tools

In this section, a brief description of the two learning systems that have been exploited will be provided. Extended description of both systems can be found in the literature.

REGAL [5,14] is a learning system, based on a distributed genetic algorithm (GA). It takes as input a set of data (training instances) and outputs a set of symbolic classification rules characterizing the input data. As usual, learning is achieved by searching a space of candidate classification rules; in this case the searching method consists in a distributed genetic algorithm.

The language L used to represent classification rules is a Horn clause language in which terms can be variables or disjunctions of constants, and negation occurs in a restricted form [13]. An example of an atomic expression containing a disjunctive term is $color(x, [yellow, green])$, which is semantically equivalent to $color(x, yellow)$ or $color(x, green)$. Such formulas are represented as bitstrings that are actually the population individuals processed by the GA. Classical genetic operators, operating on binary strings, with the addition of task oriented *specializing* and *generalizing* crossovers are exploited, in an adaptive way, inside the system (for details see [5]).

REGAL is a distributed genetic algorithm that effectively combines the Theory of Niches and Species of Biological Evolution together with parallel pro-

cessing. The system architecture is made by a set of extended Simple Genetic Algorithms (SGA) [6], which cooperates to sieve a description space, and by a Supervisor process that coordinates the SGAs efforts by assigning to each of them a different region of the candidate rule space to be searched. In practice this is achieved by dynamically devising subsets of the dataset to be characterized by each SGA. Such a form of cooperation is obtained by exploiting a coevolutionary approach [15,5].

The system RIPPER [1] also takes as input a set of data and outputs an ordered sequence of classification rules. As usual, learning is achieved by searching a space of candidate classification rules; in this case the searching method consists in the iterative application of a greedy heuristic measure, similar to the Information Gain [16], to build conjunctive classification rules. At each iteration, those training instances correctly classified by the found rules are removed and the algorithm concentrate on learning a classification rule for the remaining one. The system output is an ordered list of classification rules (possibly associated to many classes); they have to be applied in that same order to classify a new instance. An interesting features of this learning method is that it exploits on-line rule pruning while incrementally building a new classification rule to avoid overfitting.

3 Intrusion Detection in the Information Exploration Shootout Contest

An evaluation of REGAL over an intrusion detection task, by exploiting data from the Information Exploration Shootout Project (IES), is reported in this section. The IES made available network logs produced by 'tcpdump' for evaluating data mining tool over large set of data. These logs were collected at the gateway between an enterprise LAN and the outside-network (Internet). In the IES context, detecting intrusions means to recognize the possible occurrence of unauthorized ('bad') data packets interleaved with the authorized ('good') ones over the network under monitoring. The IES's project makes available four network logs: one is guarantee not to contain any intrusion attempts, whereas the other ones do include both normal traffic and intrusions attempts. In the IES context, no classification for each data packets is requested, instead an overall classification of a bunch of the network traffic, as containing or not attacks, is desired.

An approach to intrusion detection, based on anomaly detection, has been selected. We proceed as follows. IES data can be partitioned, on the base of their IP addresses, into packets exiting the reference installation (Outgoing), entering the installation (Incoming) and broadcasted from host to host inside the installation (Interlan). Three models of the packet traffic, one for each direction, have been built from the intrusion-free dataset. Then, these models have been applied to the three datasets containing intrusions. We expect to observe a significant variation in the classification rate between intrusion-free logs and logs containing intrusions because of the *anormal* characteristics of the traffic produced by the

Table 1. Experimental results of applying RIPPER to IES datasets using the raw data representation

Dataset	interlan	incoming	outgoing
normal	0.04	0.04	0.04
intrusion1	0.23	0.07	0.04
intrusion2	0.09	0.07	0.05
intrusion3	0.08	0.14	0.04

intrusive behavior. If this would actually occur, we could assert that the learned traffic models correctly capture the essential characteristics of the intrusion-free traffic. Experiments have been performed both with RIPPER and REGAL.

When RIPPER is applied to the IES data, the classification rate appearing in Table 1 becomes evident [11]. This results have been obtained by applying RIPPER to the data as available from the tcpdumped files (see Appendix A). No preprocessing over the data, such as feature construction, has been applied. The experimental findings shows that the acquired models do not exhibit very different classification rate when applied to logs containing intrusions with respect to intrusion-free logs. These findings may suggest that the exploited data representation is too detailed with respect to the capability of the learning system. In turn, this causes the learned models to miss the information characterizing intrusion-free traffic.

Table 2. Experimental results of applying RIPPER to IES datasets using a compressed data representation

Dataset	interlan	incoming	outgoing
normal	0.02	0.05	0.04
intrusion1	0.11	0.11	0.21
intrusion2	0.03	0.13	0.12
intrusion3	0.11	0.21	0.12

Following this observation, we develop a more compact representation for the packets that consists in mapping a subset of feature’s values into a single value, thus reducing the cardinality of possible features values (see Appendix B). Exploiting this representation, RIPPER’s performances become the ones reported in Table 2 and REGAL’s performances exploiting the same compact data representation appear in Table 3. The observed figures show a more stable classification behavior of the models across different traffic conditions. Also a more distinct classification performance between the intrusion-free log and the logs including intrusions is evident. A compression-based representation is then a valuable way of increasing classification performances without introducing complex feature that may involves additional processing overhead. An evaluation of

```

IF      srcprt(x,[[0,20],[40,100],[150,200],[>500]]) and
      dstprt(x,[>1024]) and flag(x,[FP,pt]) and
      seq1(x,[[100,150],[200,300],[500,5000],[>10000]]) and
      seq2(x,[[50,100],[200,300],[500,20000]]) and
      ack(x,[[0,3000],[5000,10000]]) and
      win(x,[[0,2000],[>3000]]) and
      buf(x,[<=512])
THEN IncomingPacket(x)
Coverage: (Interlan, Incoming, Outgoing) = (0, 7349, 0)

```

Fig. 1. Example of a rule characterizing part of the incoming traffic. The rule describes 7349 incoming packets without confusing them with any outgoing or interlan packet

the effect caused by the addition of complex features to the raw network data representation has been performed in [11].

For the sake of clarity, an example of rule characterizing intrusion-free Incoming packets, learned by REGAL, appears in Figure 1. The Incoming packets are characterized in term of the values of the features from their TCP/IP header. This rule successfully covers 7349 Incoming packets without being fooled by any Interlan or Outgoing ones. A description of the predicates appearing in the rule is provided in Appendix A.

4 Intrusion Detection in the 1998 DARPA Intrusion Detection Evaluation Programme

We also performed an additional evaluation of our approach over network logs from 1998 DARPA Intrusion Detection Evaluation Programme [12] whose objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated

Table 3. Experimental results of applying REGAL to IES datasets using a compressed data representation

Dataset	interlan	incoming	outgoing
normal	0.02	0.04	0.04
intrusion1	0.12	0.15	0.11
intrusion2	0.06	0.11	0.12
intrusion3	0.12	0.15	0.11

in a military network environment, was provided. We exploited data available from the KDD'99 Intrusion Detection Contest¹.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to add from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Attacks fall into four main categories:

DOS: denial-of-service, e.g. syn flood;

R2L: unauthorized access from a remote machine, e.g. guessing password;

U2R: unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks;

Probe: surveillance and other probing, e.g., port scanning.

In practice two datafiles containing classified connections are available: one has to be used for acquiring a model of the traffic and the other one for testing its performances. The distinction is important because the test file contains attack types not occurring in the learning file. This is intended to make the task more realistic.

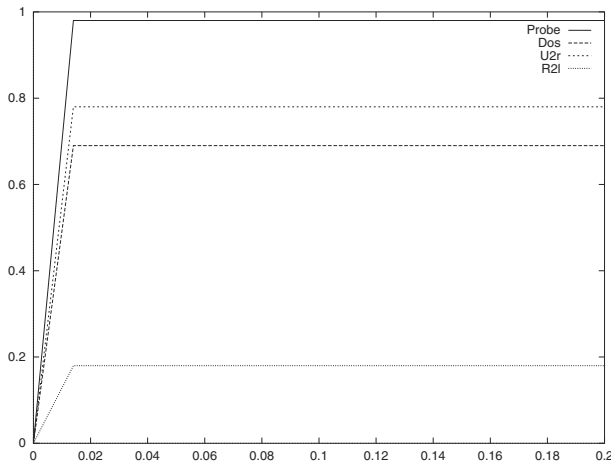


Fig. 2. Detection performances exhibited by RIPPER plus Meta-Learnig on the DARPA test data. An extended representation of the data and a complex learning approach (meta-level learning) have been exploited

¹ Information about KDD'99 Intrusion Detection Contest is available on-line at <http://www.epsilon.com/kdd98/task.html>.

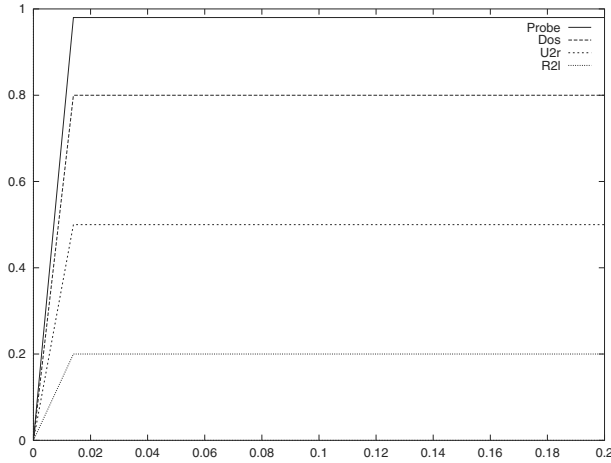


Fig. 3. Detection performances exhibit by REGAL on DARPA test data (no additional Meta-Learning has been used). A compressed data representation has been exploited

In figure 2 and figure 3, performances of RIPPER plus Meta-Learning (as used in [11]) and REGAL over DARPA's data are respectively shown. In the figures, the x axis represents the false alarm rate, i.e. the percentage of 'Normal' connections labeled as intrusions, whereas the y axis represents the detection rate, i.e. the percentage of intrusions that have been correctly recognized. The reported performances have been obtained on the connections occurring in the test file. The reported graphs show similar detection performances, between the models acquired by the systems, for Probe and Remote-To-Local (R2l) attacks types. Instead, REGAL's model performs slightly better on DOS type attacks but worst on User-To-Root (U2r) attacks.

Let consider, now, the modeling approaches exploited by the two systems. Lee and Stolfo [11] run RIPPER over an extended data representation of the tcp connection including, in addition to the basic tcp features, derived information such as: the number of connections to the same host in the past two seconds ('count'), the number of connections to the same service, as the current connection, in the past two seconds ('srv-count'). These features have been chosen on the basis of the authors expertise. A preprocessing of the raw network logs is required in order to exploits this features. Several classifiers (rule sets) for each attack type have been obtained. Eventually meta-learning, i.e. learning at the classifier level, has been applied to produce the reported performances.

REGAL, on the contrary, has been run after applying a compression mapping to the feature values, as described in Appendix B. Only the basic features of a TCP connection have been considered such as: 'duration', stating the length (number of seconds) of the connection, 'protocol-type', stating the type of the

protocol (e.g. tcp, udp, etc.), or 'src-bytes', stating the number of data bytes from source to destination. No additional meta-learning phase is necessary.

5 Conclusions

We investigated the potentiality of a distributed genetic learner and of a heuristic based learner in modeling network traffic to assist in detecting anomaly in data traffic. Two different applicative contexts to detect intrusions have been explored.

We analyzed the effect of exploiting a compressed representation for the network data packet values in modeling pattern of traffic. We are confident that a compression of the values of the packet's features may result in an abstract representation that, on one hand, could allow better recognition performances and, on the other one, could reduce the complexity of acquiring model of the traffic. We believe that discovering the right representation be an important prerequisite for the automatic modeling and the on-line deployment of intrusion detection system.

A Appendix. The Information Exploration Shootout Raw Data Representation

The IES data (available on line at <http://iris.cs.uml.edu>) have been collected by means of the TCPDUMP utility. Taking into account privacy concerns, the data portion of each packet has been dropped. For each packet in the datasets the following attributes are available:

time - converted to floating pt seconds .. $hr*3600+min*60+secs$.

addr and port - (just get rid of x.y.256.256.port) The first two fields of the src and dest address make up the fake address, so the converted address was made as: $x + y*256$.

flag - added a "U" for udp data (only has ulen) X - means packet was a DNS name server request or response. The ID# and rest of data is in the "op" field. (see tcpdump descrip.) XPE - means there were no ports... from "fragmented packets".

seq1 - the data sequence number of the packet.

seq2 - the data sequence number of the data expected in return.

buf - the number of bytes of receive buffer space available.

ack - the sequence number of the next data expected from the other direction on this connection.

win - the number of bytes of receive buffer space available from the other direction on this connection.

ulen - if a udp packet , the length.

op - optional info such as (df) ... do not fragment.

Particular attention has to be taken when dealing with fields like 'op' that contains a large amount of values.

Table 4. Compression mapping applied when dealing with IES network data

Original Value	New Value
$0 \leq \text{srcport} < 50$	srcport=0
$50 \leq \text{srcport} < 100$	srcport=0
<... skipped test ... >	<... skipped text ... >
$\text{srcport} > 20000$	srcport=10
<... skipped text ... >	<... skipped text ... >
op contains "DF"	op=1
op contains "NXDomain"	op=2
op contains ANY OTHER VALUE	op=3

B Appendix. The compressed Feature Representation of IES Data

Some features of the IES data may assume a large set of values either continuous or discrete. These large sets do impact over classification performances of the learned models because of the intrinsic difficulty of acquiring rule having a general scope. Then, a reduction of the range of potential values is desirable to increase both the generality of the learned model and to reduce the learning computational complexity.

An alternative approach to this problem consists in adding/building more complex features, combining the basic ones, to the original data representation. We do not follow this approach in this work, because we believe that the previous approach is simpler and should be the first to be analyzed.

As an instance of reducing the range of the feature values, considers that the feature 'srcport' (see Appendix A for a description) may virtually assume any integer number from 0 to 65536. Also, the feature 'op' may assume hundreds of discrete values. Taking into account basic knowledge about the domain, we manually developed the reduction mapping shown in Table 4. This mapping is not to be considered as the best one but as a proof that a simple reduction of the feature values may positively impact over the recognition capabilities.

References

1. W. Cohen. Fast effective rule induction. In *Proceedings of International Machine Learning Conference 1995*, Lake Tahoe, CA, 1995. Morgan Kaufmann. 314, 315
2. D. Denning. An intrusion detection model. *IEEE Transaction on Software Engineering*, SE-13(2):222–232, 1987. 313
3. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996. 313

4. A. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *USENIX Workshop on Intrusion Detection and Network Monitoring*. USENIX Association, 1999. 313
5. A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3 (4):375–416, 1995. 314, 315
6. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Ma, 1989. 315
7. S. A. Hofmeyr, A. Somayaji, and S. Forrest. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998. 313
8. S. Kumar and E. Spafford. A pattern matching model for misuse detection. In *National Computer Security Conference*, pages 11–21, Baltimore, 1994. 313
9. T. Lane and C. Brodley. An application of machine learning to anomaly detection. In *National Information Systems Security Conference*, Baltimore, 1997. 313
10. T. Lane and C. Brodley. Approaches to online learning and conceptual drift for user identification in computer security. Technical report, ECE and the COAST Laboratory , Purdue University, Coast TR 98-12, 1998. 314
11. W. Lee, S. Stolfo, and K. Mok. Mining in a data-flow environment: experience in network intrusion detection. In *Knowledge Discovery and Data Mining KDD'99*, pages 114–124. ACM Press, 1999. 313, 314, 316, 317, 319
12. R. Lippmann, R. Cunningham, D. Fried, I. Graf, K. Kendall, S. Webster, and M. Zissmann. Results of the DARPA 1998 offline intrusion detection evaluation. In *Recent Advances in Intrusion Detection 99, RAID'99*, W. Lafayette, IN, 1999. Purdue University. 317
13. R.S. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach*, volume I, pages 83–134. Morgan Kaufmann, Los Altos, CA, 1983. 314
14. F. Neri and L. Saitta. Exploring the power of genetic search in learning symbolic classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-18:1135–1142, 1996. 314
15. M. A. Potter, K. A. De Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Sixth International Conference on Genetic Algorithms*, pages 366–372, Pittsburgh, PA, 1995. Morgan Kaufmann. 315
16. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, California, 1993. 315

Learning Patterns of Behavior by Observing System Events

Marlon Núñez

Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga
Campus de Teatinos s/n, Málaga 29071, Spain
mnunez@lcc.uma.es

Abstract. The proposed algorithm (BPL) induces behavior patterns from events taking into account characteristics of observed systems and their environment. The main strategy of this method consists on building summaries of the behaviour of a system as events arrive, and take these summaries as training examples. BPL constructs summaries with new features from events, like duration of current event values, repetitions of an event in a period of time, amongst others. This algorithm has been tested in learning faulty behavior of networks with the purpose of continuously predicting alarms.

1 Introduction

The learning of behavior patterns is important for all intelligent entities and it is also useful for those researchers who want to know behavior patterns of a group of systems. This knowledge is useful in several fields. This paper is mainly focused in applying behavior knowledge for predicting events and explaining patterns. Applications can cover the control of systems, system imitation, customer behavior analysis, and alarm analysis, amongst other fields.

The techniques that could be considered nearer to this field are those related to the learning of patterns of sequences. Those technics are adapted to the analysis of unordered sets of examples. Basically, such data can be viewed as a sequence of events, where each event has an associated time of occurrence. When discovering episodes in a network alarm log, the aim is to find relationships between alarms. Such relationships can then be used in an on-line analysis of the incoming alarm stream, e.g., to better explain the problem that causes alarms, to suppress redundant alarms, and to predict severe faults.

Technical problems related to the recognition of episodes have been researched in several fields. A problem of discovering frequent episodes in a sequence of events was presented in [5]. Their patterns are arbitrary directed acyclic graphs, where each vertex corresponds to a single event (or item). An edge from event A to event B denotes that A occurred before B. They move a time window across the input sequence, and find all patterns that occur in some user-specified percentage of windows. An algorithm is designed for counting the number of occurrences of a pattern when moving a window across a single sequence. In a different approach [8] the problem is how to discover all sequential patterns with a user-specified minimum

support. Each sequence is a list of transactions ordered by transaction-time, and each transaction is a set of items. In Bioinformatics, the problem is to discover patterns common to a set of related protein or amino acid sequences [3].

2 BPL Algorithm

The general strategy of the BPL algorithm is to transform a problem that is seemingly not supervised, like a series of successive events in the time, into a supervised problem, where the training examples are mainly *behavior summaries*. Each summary will have several labels that are the temporal distances from the moment of the summary to the occurrence of each target event. Since these labels are numeric values, the Behavior Pattern Learner (BPL) algorithm generates regression trees, one for each target event that is wanted to be predicted.

The algorithm learns the behavior in terms of several groups of rules. Each group of rules is used for predicting a target event as well as explaining the prediction. The purpose of learning groups of rules is allowing the specialization of knowledge for each target event and improving the precision due to this specialization.

During the inference, several rules might fire simultaneously, predicting different target events, their expected intervals of time and their confidence. As the time gets closer to the expected occurrence of the target events, other rules might fire to predict the same target events for a shorter time interval; it is also possible that hypotheses change and previous hypothesis may be substituted by new hypothesis with different intervals of time, depending on the new events that have arrived.

Two important concepts used by BPL algorithm must be defined: events and BehaviorSummaries.

2.1 Events

An event indicates a change of value of an object attribute, as well as the time in which this change took place. An event is represented as *[Object. Attribute: Value, Time of occurrence]* where object is the identifier of an observed system. Attribute is the identifier of the variable that changes its value. Value is the new value of Attribute that changed at the specified time of occurrence. An examples of events is *[CAR122. Clutch: "pressed", 14:00]* which indicates that in the CAR122, the clutch was pressed at 14:00. Another example: *[PC1234. Alarm: CommunicationProblem, 12:00]* indicates that the PC1234 emitted the event *alarm = "Communication Problem"* at 12:00. If we are talking about an event in general without specifying the object, the notation *[Clutch: "pressed"]* is used.

2.2 BehaviorSummaries

The system learns from training examples called Behavior Summaries, made up of three kinds of preconditions and its consequences. The precondition types are Event Characteristics, SystemCharacteristics and Environment Characteristics. The possible

consequences of those characteristics are described in terms of the times of occurrence from that summary to target events.

2.2.1 EventCharacteristics

EventCharacteristics are new features calculated from events:

- Duration [*StillValidEvent*]: The duration of the validity of an event that occurred in the past but is still valid: $\text{Duration}[\text{StillValidEvent}] = \text{time}[\text{BehaviorSummary}] - \text{time}[\text{StillValidEvent}]$
- Latency [*PastEventNoLongerValid*]: There is a probability > 0 that *PastEventNoLongerValid* continues having a *latent* effect during a certain time interval. The time of latency is the time past since *PastEventNoLongerValid* occurred. But there should be a limit. This limit is a time window called LatencyWindow [Event]. After this user-defined window time the event is supposed not to have any effect.
- Repetitions [*PastEvent*]: the number of repetitions of PastEvents in a period of time equal to the LatencyWindow[PastEvent].

These are the most useful new event characteristics. But some others can be used, like PastDuration [*PastEventNoLongerValid*] or LatencyOfRepetition [*PastEvenNoLongerValid*].

User has to provide his knowledge about the LatencyWindow of every event, namely, the maximum time of influence of every value of a variable. E.g. the event [car. clutch:"pressed"], has a LatencyWindow, that could be, say, 20 seconds, at most, which means that whatever happened to the car behavior will not be influenced by a past event [car. clutch: "pressed"] after 20 seconds. On the other hand, an event [car. brakeAlarm: "low level"] will influence future events of the car for a running period of, say, one month [car. brakeAlarm: "low level"]. That is to say: 30x60x60x24x30 seconds.

2.2.2 SystemCharacteristics

Set of characteristics of the system that generated the events (The SystemCharacteristics_i are all the values that describe the system that generated the event in time $\text{time}[\text{BehaviorSummary}_i]$). There are special events, called *state* events, which might update values of system attributes.

2.2.3 EnvironmentCharacteristics

Set of characteristics of the environment of the system (all the values that describe the environment in a time $\text{time}[\text{BehaviorSummary}_i]$, that is to say, the instant of the BehaviorSummary_i).

2.2.4 Possible Consequences

The BehaviorSummary also registers the possible consequences of a situation. The Possible Consequences is a list of occurrence times from the BehaviorSummary to each target event: Then, $\text{PossibleConsequence}_i = (\text{time}[\text{targetEvent}_1], \text{time}[\text{targetEvent}_2], \dots, \text{time}[\text{Target Event}_k])$, where $\text{time}[\text{Target Event}_i]$, is the time of the next occurrence of the TargetEvent_i.

2.3 Description of the BPL Algorithm

BPL algorithm has two phases: BehaviorSummary creation and learning phase. During the BehaviorSummary creation, BPL receives events. Each event generates several BehaviorSummaries to allow analysis of its consequences. During learning phase, BehaviorSummaries are used to grow regression trees, one for each target event. Every tree constructs a group of behavior rules. Table 1 illustrates BPL algorithm. Basic BPL algorithm, is independent of the regression tree method used.

2.3.1 Construction of BehaviorSummaries from Events

As it was mentioned previously the system learns from behavior summaries labeled with continuous classes. When an event arrives, all these variables (duration, repetition, and latency, among others) are updated in a table called Instant BehaviorSummary (IBS) table. There will be an IBS table for each observed system. When there is an order for constructing a behavior summary with an older time than the time of arrived event, a summary is constructed using values of attributes at the order time and calculated attributes like duration, repetitions and latency of events.

2.3.2 Scheduling Summary Orders

BehaviorSummaries are not generated periodically. Each event generates a schedule of summary orders that force the learning program to monitor its consequences in terms of occurrence of each Target Event.

An event at time $time[event]$ generates several summary orders inserting them in chronological order in the SummaryOrdersList until a temporal position $time[event] + LatencyWindow[event]$. The Summary Orders are being scheduled ordered by time in the SummaryOrdersList. Once an event arrives, BPL algorithm analyses if constructing a Summary by consulting the top of the SummaryOrdersList. If the time of the incoming event is greater than that time of the summary order, then the summary is constructed. This way, BPL will not have to update IBS table continuously, but only when an event arrives.

2.3.3 Creation and Labeling of Behavior Summaries

When an event arrives, among other steps explained above, a SummaryOrder might be executed. Then BPL constructs a Behaviour summary, updating the IBS table calculating duration, repetition and latency according to the time of the BehaviorSummary. A BehaviorSummary will have as many numeric classes (also called labels) as target events are declared. Initially, BehaviorSummary labels are empty, pending on being filled in. To label a BehaviorSummary, BPL has to wait until a target event occurs. When one of the target events arrives it forces labeling the pending summaries. The label of each pending summary will be the time between that BehaviorSummary and the target event. If a target event did not arrive during a LatencyWindow (the largest one), the label is "it did not happen", namely, *d.n.h.* This value is set in the regression analysis as a specific negative value. In other words, there is a special label of BehaviorSummaries which means that a target event did not arrive or it did arrived at a time greater that the largest LatencyWindow of events. Attributes also have a special value: *n.a.* which stands for "not applicable". If the field

duration, for instance, has the value *n.a.*, it means that the event occurred in the past and it is no longer valid, therefore it does not make sense to have any value. This value is also set as a specific negative value.

Table 1. BPL algorithm

Input:	Events, list of Target Events _{1...n} , LatencyWindows, N (Minimum new Behavior Summaries), and minimum support.
Output:	Behavior rules to predict/explain Target Events _{1...n}

Continually insert incoming events in the eventLog

// Behavior Summary creation Phase

While there exist not analyzed events in the eventLog,

Read the oldest not analyzed event. Consider it as *AnalyzedEvent*

While time[AnalyzedEvent]>time[top (BehaviorSummaryOrder List)]

Create a BehaviorSummary based on the InstantBehaviorSummary table. Insert it in BehaviorSummaryLog.

Extract the order at the Top of the BehaviorSummaryOrdersList

If AnalyzedEvent is a target event

Label each BehaviorSummary_i whose label is pending on update, with time[AnalyzedEvent] - time[BS i]

Based on the AnalyzedEvent, update their event characteristics in the InstantBehaviorSummary (IBS) table (number of repetitions, Duration/Latency, etc.)

Based on the AnalyzedEvent, schedule a set of SummaryOrders inserting them into the SummaryOrdersList ordered by time until LatencyWindow_{AnalyzedEvent}

// Learning phase

If there exist N new BehaviorSummaries with labels of a target event_i

Grow a Regression Tree_i based on BehaviorSummaryLog

Construct Behavior Rules for target event_i and replace previously learnt rules.

2.3.4 Learning Time Intervals

When there are N new BehaviorSummaries already labeled with a target event, BPL uses all summaries to generate knowledge to predict the target event. It learns as many regression trees as continuous classes, namely, target events. Although BPL can use any regression tree algorithm [1], [4], it has been tested with the EGR method [7].

EGR selects the best variable, analyzing the mixture of the data in every possible split, calculating the mean μ , standard deviation σ , and weight π of the components of the mixture, $[(\mu_1, \sigma_1, \pi_1), (\mu_2, \sigma_2, \pi_2) \dots (\mu_n, \sigma_n, \pi_n)]$. EGR learns regression trees using background knowledge (taxonomies and cost) associated to attributes, in a similar way that EG2 [6] for induction of decision trees.

2.3.5 Construction of Behavior Rules

Regression trees do not show target events. They handle special negative numeric information, meaning *n.a.* and *d.n.h.*, this step pursues the following transformations:

- Every regression tree is transformed into a group of rules with a consequence in which the target event and the statistical support is expressed
- Negative numeric value in attributes, previously illustrated as *n.a.*, into symbolic values meaning "not applicable".
- Negative numeric value in classes, previously illustrated as *d.n.h.*, into symbolic values meaning "it will not happen".

2.4 An Example

Let us suppose that we have systems. The first one is of Type A and the second one is of type B. Two variables will be observed: Var1 and Var2. The values of Var1 are A, R and S. The values of the attribute Var2 are W and F. Figure 2 illustrates a time-line detailed by the symbols: (.), (~) and (/). The symbol (.) indicates a minute with no events; symbol (~) indicates a minute in which the last value of the attribute did not change; symbol (/) indicates an hour in which the last value of the attribute did not change. Vertical gray lines indicate BehaviorSummaries. Let's say that the target events are [Var1: A].

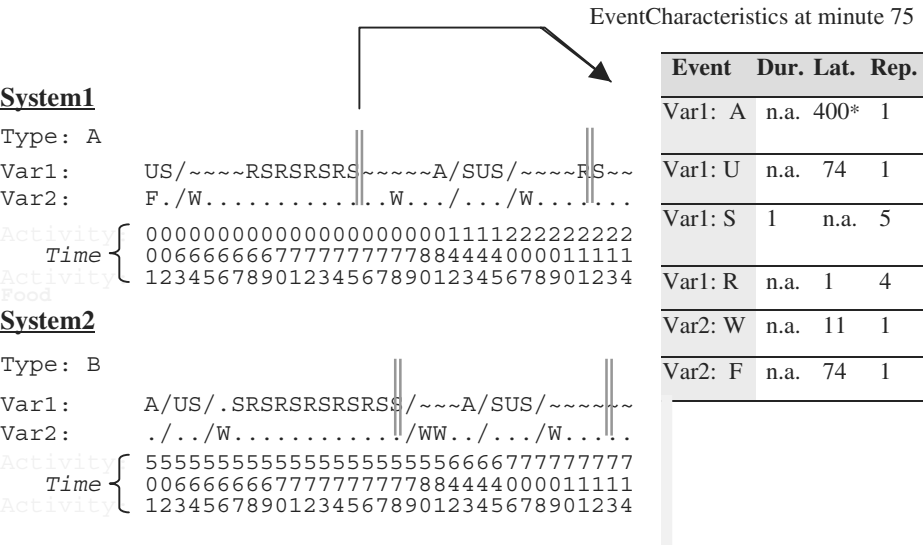


Fig. 1. Evolution of attribute values and the BehaviorSummary at minute 75

Events from minute 1 to minute 75 update the ISB table. Figure1 also shows the calculated EventCharacteristics for System1 at minute 75. The asterisk "*" means that last time system1. Var1 changed its value to A was 400 minutes ago, that is to say, outside the timeline. Let's see what happen when event [Var2: W] arrives at time 78, and it is considered as the *analyzed event*. Then time[analyzed Event] >

top[BehaviorSummary Order] since minute 78 > minute 75. Therefore the **While** instruction applies. Consequently a BehaviorSummary is created from ISB table. This new BehaviorSummary is inserted in a log, without label yet, because its consequences are not known yet. Since the event [Var2: W] is a target event, it labels the BehaviorSummary with the distance time from the behavior summary (minute 75) to the time of the target Event (minute 78), that is to say, 3. The BehaviorSummary is a record of 20 fields: two with the current values of attributes (Var1 and Var2) and the other 18 fields regarding EventCharacteristics that summarized that situation. This training example has one numeric class. The process explained above goes on continuously. When there are N new behavior summaries, then they become training instance of a regression tree method. Figure 2 shows parts of the two regression trees generated, one for each target event.

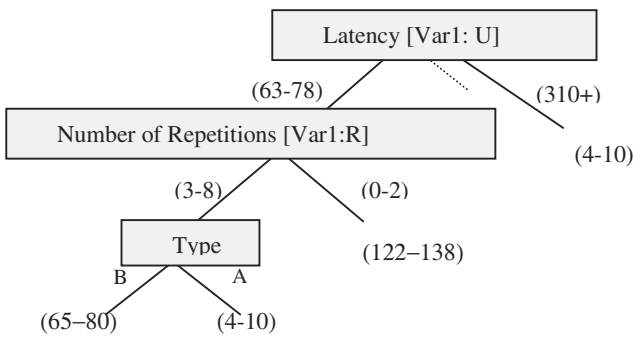


Fig. 2. Regression tree for predicting [Var1: A]

3 Experimentation

Basic BPL has been evaluated trying to find faulty-behavior patterns. An analyzed domain was the faulty behavior of a PC Network in terms of its alarms. The PC network is made up of a Server, three Workstations and a LAN. The system was feed with six event logs of operating system (WindowsNT Server and Workstation).

We analyzed 450 errors and warnings messages, corresponding to 10 months of Windows NT administration. We selected 5 target events mostly related to network problems. We took the *oldest* 70% of consecutive events for training and the most recent 30% for testing.

BPL generated 52 behavior rules for predicting 5 target events. It was surprising that 48% of behavior rules have "will not happen" value as a consequent. These rules describe the preconditions for predicting that a target event will not occur in the *near future* (max. of LatencyWindows). The amount of rules of this kind is large; after analyzing these rules, we realized that there were very useful for being more specific in the prediction. For example, if we say that target event X will probably occur in [25-30] minutes and target event Y will not occur in [320+] minutes, we are giving more information about the network.

Latency Windows affects directly the rules learnt. If LatencyWindows are too small, BPL can not recognize true consequences of Alarm "Netlogon" because this event does not generate SummaryOrders to a future time when these consequences really occur. On the other hand, too big LatencyWindows adds many patterns not related to the problem, which does not help to find true patterns.

4 Conclusions

An important characteristic of the BPL algorithm is that periodic analysis does not exist. Generation of regression trees is carried out every time that N new BehaviorSummary arrives, which allows not to be necessary to establish a certain learning rhythm. If there are target events with short effect and, simultaneously, other target events with long effect, BPL will generate each tree with different rhythm and different time intervals. On the other hand, an important finding, ignored by sequence pattern learning approaches, is the importance of the characteristics of systems and environment how these factors affect systems behavior.

Acknowledgments

Special thanks to Rafael Morales for giving comments and support to this research work. I would also thank Nohelia Barreiro, Raúl Fidalgo and Ignacio Ballesteros for developing, and testing some modules. I would also thank José L. Pérez who helped with some logistical support. This work has been partially supported by project FACA PB98-0937-C04-01 of the CICYT, Spain. FACA is a part of the FRESCO project.

References

1. Breiman, L., Friedman, J.H., Olsen, R.A., & Stone, C.J., (1984). *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, California.
2. Grossi, R and Luccio, F. (1989). Simple and efficient string matching with k mismatches. *Information Processing Letters*, 33:113 – 120, 1989.
3. Jonassen I., Collins J.F. and Higgs D. Finding flexible patterns in unaligned protein sequences. *Protein Science*, 4 (8): 1587 - 1595, 1995.
4. Karalic A. (1992) Employing Linear regression in regression trees leaves. In Technical Report IJS DP-6450, Jozef Stefan Institute, Ljubljana, Slovenia.
5. Manila, H., Toivonen H., and Verkamo Inkeri (1997) Technical Report C-1997-15, Department of Computer Science, University of Helsinki.
6. Núñez, M. (1991) The Use of Background Knowledge in Decision Tree Induction. *Machine Learning (Vol. 6, Nº3)*. Los Altos, CA. Morgan Kaufman.
7. Núñez, M. (2000). *Generalised Regression Trees*, accepted paper for the 14th Intl. Conference of Statistical Computing (Compstat), The Netherlands.
8. Srikant, R. and Agrawal R (1996). Mining Sequential Patterns: Generalization and Performance Improvements. In *Proceedings of the 5th International Conference EDBT-96*, Avignon, France.

Dimensionality Reduction through Sub-space Mapping for Nearest Neighbour Algorithms

Terry R. Payne¹ and Peter Edwards²

¹ The Robotics Institute, Carnegie Mellon

University, 5000 Forbes Avenue, Pittsburgh PA 15232, USA

² Department of Computing Science, King's College, University of
Aberdeen, Aberdeen, Scotland, AB24 3UE

Abstract. Many learning algorithms make an implicit assumption that all the attributes present in the data are relevant to a learning task. However, several studies have demonstrated that this assumption rarely holds; for many supervised learning algorithms, the inclusion of irrelevant or redundant attributes can result in a degradation in classification accuracy. While a variety of different methods for dimensionality reduction exist, many of these are only appropriate for datasets which contain a small number of attributes (e.g. < 20). This paper presents an alternative approach to dimensionality reduction, and demonstrates how it can be combined with a Nearest Neighbour learning algorithm. We present an empirical evaluation of this approach, and contrast its performance with two related techniques; a *Monte-Carlo* wrapper and an *Information Gain*-based filter approach.

1 Introduction

The dimensionality of a supervised learning task can be characterised in many ways. A dataset contains a number of situations or instances, each of which contain several attributes and a class value. The attributes may be considered to be *predictor* (relevant) attributes, as they may be used to induce a classification hypothesis (sometimes represented as a set of rules or a decision tree) which is later used to predict the class of a new instance. However, other attributes may be considered as *irrelevant* attributes, as they contribute nothing to the classification task, and may even degrade the accuracy of the resulting classifications. The time taken to induce a concept description from a training set, and to predict the class of a new instance, is dependent on both the learning algorithm used and the number of attributes present (i.e. the number of dimensions used to describe the data).

Determining which of the attributes are relevant to the learning task (i.e. identifying attributes which predict the class value) is a central problem in machine learning. In the past, domain experts selected the attributes believed to be relevant to the learning task. However, in the absence of such background knowledge, automatic techniques are required to identify such attributes, as the presence of irrelevant attributes can reduce the performance of various learning

techniques. Nearest neighbour algorithms are especially prone to the inclusion of such attributes within datasets, as many utilise distance metrics that calculate an average similarity measure across all of the attributes [1]. In addition to this, the sample complexity (i.e. the number of instances required to learn a concept) grows exponentially with the number of irrelevant attributes [11], indicating that simple nearest neighbour algorithms may not scale up well if irrelevant attributes are present. For these reasons, various weighting techniques have been investigated in an attempt to reduce the contribution of irrelevant attributes within nearest neighbour algorithms [14].

A *redundant-attribute* set occurs when two or more relevant attributes exist, such that each makes an equal contribution towards learning some concept [10]. In general, only a single member of this redundant-attribute set is required when learning the concept. The inclusion of more than one member will not only increase the time taken to induce the concept description, but may place emphasis on the part of the concept description the attributes in the set represent, and thus reduce the influence of other relevant attributes [12]. The remaining attributes in this set are sometimes described as *redundant*.

In this paper, we present an alternative approach that can be used by machine learning algorithms to reduce the dimensionality of datasets. The instances in a dataset are represented as vectors within an instance space. An approximation of this space is then found, and the vectors are projected into this lower dimensional space. This is achieved by using the geometric technique, *Correspondence Analysis* [9], to identify and approximate the lower dimensional space (or *sub-space*). This sub-space can then be used by a nearest neighbour learning algorithm to perform class predictions for new instances. The two learning algorithms, *CA* and *CACP* utilise this approach to dimensionality reduction, and are described below.

2 Dimensionality Reduction for Machine Learning and Information Retrieval Systems

The dimensionality reduction techniques used by machine learning algorithms can be grouped into two broad categories: those that are instances of the *filter* model, where the selection technique is independent of the learning algorithm used to learn the concept hypothesis; and those that are instances of the *wrapper* model, where the learning algorithm is integral to the selection mechanism [10]. Both models perform a search within a space of attribute subsets to determine the optimal (or sub-optimal) subset for the classification task. The size of the search space is exponential; if there are n attributes in the original dataset, then there are a total of 2^n possible states in the search space. This exponential rise means that exhaustive, optimal searches are infeasible for all but simple problems involving few attributes. Therefore, most systems perform greedy or stochastic searches. Several studies have also shown that the wrapper model can identify better attribute sets, when compared with the filter model [10]. However, induction is performed at every search state visited. The number of instances, i ,

in the training set and the control mechanism used to evaluate each state will also influence the length of time taken to determine the final attribute subset.

Dimensionality reduction techniques have also been utilised by a variety of Information Retrieval (IR) systems [18] to reduce the number of terms used to index documents. These techniques have also been applied to the problem of reducing the number of terms presented to learning algorithms for text categorisation problems [7]. Whilst some studies have omitted this stage, the number of unique terms (typically in the region of tens or hundreds of thousands) is prohibitively high for most machine learning algorithms. Many text categorisation systems employ *filter* based methods. *Latent Semantic Indexing* (LSI) [7] is an alternative approach for reducing the number of dimensions used to represent documents in many IR systems. LSI utilises an orthogonal decomposition technique to determine a smaller numeric representation for each document. A corpus is represented as a *term* $\times$ *document* matrix, where each row corresponds to a document, and each column to one of the terms appearing within the corpus. Thus, each document (i.e. row vector) is expressed as a point within some geometric space. An orthogonal decomposition technique is then applied to this matrix, resulting in a set of decomposed matrices that describe this space and the points within it. The space can then be approximated (by approximating the decomposed matrices) resulting in a lower dimensional representation of the points [15].

Various studies have demonstrated that LSI improved the performance of both IR and text categorisation systems. For example, Deerwester et. al. [7] achieved a reduction from 5000-7000 terms to 100 dimensions. Similar techniques have also been successfully applied to the problem of reducing the dimensionality of protein sequence data for presentation to neural networks [19]. The size of the input vectors presented to a backward propagation neural network was reduced from 9696 to 100, resulting in an overall improvement in the predictive accuracy of the neural network. These studies have demonstrated that LSI and the principles behind this method work for specific problems, but LSI's applicability to a broader range of classification tasks has not yet been investigated. For this reason, we have investigated a similar technique, based on *Correspondence Analysis* [9], and have developed two learning algorithms, *CA* and *CACP*, which combine variations of this technique with a Euclidean nearest neighbour learning algorithm. These algorithms have been applied to a variety of classification problems found in the UCI Machine Learning Database Repository [4], and to artificial data (described in Section 5).

3 Subspace Approximation through Correspondence Analysis

Correspondence analysis is a mathematical tool that is used to graphically present multi-dimensional data within low (e.g. two or three) dimensional data plots [14,15]. This is achieved by identifying an approximation of the Euclidean space that contains the instances (which are represented as vectors). This ap-

proximation is used to project the vectors from a J -dimensional instance space into a K -dimensional sub-space, where J is the number of attributes of the dataset, and consequently the number of components of the vectors, and K (where $K \leq J$) is the rank of the approximated space.

The approximation is achieved by first determining an orthonormal basis for the instance space, and then removing those dimensions that have low singular values. *Singular Value Decomposition* (SVD) [16,9] is normally used to perform the orthogonal decomposition, although other decomposition approaches, such as the ULV decomposition [3], can be used to replace SVD for this task. The SVD of a matrix $\mathbf{X}$ of I rows (i.e. instances) and J columns (i.e. attributes) can be expressed as:

$$\mathbf{X} = \mathbf{L} \mathbf{D} \mathbf{R}^T$$

where $\mathbf{L}^T \mathbf{L} = \mathbf{R}^T \mathbf{R} = \mathbf{I}$ (the identity matrix). The orthonormal vectors of $\mathbf{R}$, called the right singular vectors, form an orthonormal basis for the rows of $\mathbf{X}$. The diagonal matrix $\mathbf{D}$ contains the singular values of $\mathbf{X}$, where the elements of $\mathbf{D}$: $d_1 \geq d_2 \geq \dots \geq d_N > 0$, and $N \leq \min(I, J)$. A third matrix, $\mathbf{L}$, is also expressed, which forms an orthonormal basis for the columns of $\mathbf{X}$.

The sub-space approximation framework used by *CA* and *CACP* consists of two main routines: one that generates a mapping function between the original space and the transformed and approximated sub-space (Figure 1); and a routine that uses the mapping function to project instances from the original space into the new space [14,15]. Data sets are presented to these routines as matrices, where each row of the matrix corresponds to an instance, and each column corresponds to one of the attributes of the dataset. The mapping function¹ consists of the basis $\mathbf{R}_{(K)}$ of the approximated sub-space, and a centroid, $\bar{\mathbf{y}}$. Instances, represented as vectors in the matrix $\mathbf{Y}$, are projected into the new space by translating them with respect to the centroid, $\bar{\mathbf{y}}$, and multiplying the translated vectors with the basis, $\mathbf{R}_{(K)}$. Thus, to determine a K -rank approximation of the dataset $\mathbf{Y}$:

1. Find the centroid vector $\bar{\mathbf{y}}$ for the training dataset $\mathbf{Y}$.
2. Translate the training dataset by the centroid vector into the matrix $\mathbf{X} = \mathbf{Y} - 1\bar{\mathbf{y}}^T$.
3. Determine the basis $\mathbf{R}$ and the diagonal singular matrix $\mathbf{D}$ of $\mathbf{X}$ using singular value decomposition.
4. Select the K columns of $\mathbf{R}$ (or K rows of $\mathbf{R}^T$) that correspond with the largest K singular values in the diagonal matrix $\mathbf{D}$.
5. Project the instances represented by the matrix $\mathbf{X}$ into the space characterised by $\mathbf{R}_{(K)}$, by multiplying $\mathbf{X}$ with $\mathbf{R}_{(K)}$.

Two algorithms have been developed based on the sub-space mapping approach described above. The first, *CA*, uses the function *generate_mapping* which ignores class information when generating the basis $\mathbf{R}$ from the training data. The second algorithm, *CACP*, exploits the class labels when determining the

¹ The details of these functions are described in greater detail in [14].

mapping function. The *generate_cpmapping* routine generates a single *prototype* point for each class, by finding the centroid of all the instances belonging to that class. Once all the prototype points have been found, they are used to generate the new basis, $\mathbf{R}$.

<pre> 1 proc generate_mapping($\mathbf{Y}$, rank) $\equiv$ 2 $\mathbf{y} = \text{get\_centroid\_vector}(\mathbf{Y})$; 3 $\mathbf{X} = \text{translate\_data}(\mathbf{Y}, \mathbf{y})$; 4 5 $[\mathbf{L}, \mathbf{D}, \mathbf{R}] = \text{SVD}(\mathbf{X})$; 6 7 $\mathbf{R}_{(K)} = \text{low\_rank}(\mathbf{D}, \mathbf{R}, \text{rank})$; 8 map[basis] = $\mathbf{R}_{(K)}$; 9 map[centroid] = $\mathbf{y}$; 10 return(map). </pre>	<pre> 1 proc generate_cpmapping($\mathbf{Y}$, rank) $\equiv$ 2 $\mathbf{y} = \text{get\_centroid\_vector}(\mathbf{Y})$; 3 $\mathbf{X} = \text{translate\_data}(\mathbf{Y}, \mathbf{y})$; 4 $\mathbf{P} = \text{get\_class\_prototypes}(\mathbf{X})$; 5 6 $[\mathbf{L}, \mathbf{D}, \mathbf{R}] = \text{SVD}(\mathbf{P})$; 7 8 $\mathbf{R}_{(K)} = \text{low\_rank}(\mathbf{D}, \mathbf{R}, \text{rank})$; 9 map[basis] = $\mathbf{R}_{(K)}$; 10 map[centroid] = $\mathbf{y}$; 11 return(map). </pre>

Fig. 1. The sub-space mapping algorithms, *generate_mapping* and *generate_cpmapping*, are used by *CA* and *CACP* respectively to map dataset $\mathbf{Y}$ to a sub-space of rank *rank*

4 Experimental Design

Many machine learning systems incorporate, or utilise some form of dimensionality reduction to generate an optimal (or sub-optimal) subset of dimensions prior to induction. The sub-space approximation techniques described above project instances (represented as data points within some instance space) into a lower dimensional sub-space. To compare the benefits (in terms of predictive accuracy) of this approach with other attribute selection techniques, a suitable learning paradigm is required. Instance-based learning algorithms, which are sometimes referred to as Nearest Neighbour (NN) algorithms [6] are ideal, as the accuracy of these techniques degrades in the presence of irrelevant or redundant data [14]. They store and represent some or all the training instances as data points within a hyperdimensional instance space. The instance space is usually described by N dimensions, where each dimension corresponds to a single attribute of the dataset. New (unseen) instances are classified by determining their location within this instance space, and by identifying their nearest neighbour using some *distance* function. The class value of the nearest instance is then used to predict the class of the unseen instance.

To compare the effects of using correspondence analysis for dimensionality reduction with more traditional approaches to attribute selection, a wrapper based attribute selection method was implemented. The search method used was a stochastic search known as the *Monte Carlo* method [13]. This method was chosen as the number of search states visited can be controlled, and, unlike hill climbing approaches, it is not susceptible to local maxima [14]. It is also possible to show that as the number of states visited increases, so does the

probability of finding an optimal solution [13]. This method searches for the best attribute subset by selecting a random subset and evaluating it. The evaluation was performed using a leave-one-out cross validation with the nearest neighbour Euclidean distance learning algorithm on the training dataset.

A filter-based attribute selection method was also tested. The learning algorithm, *FNN* was implemented, which utilises the C4.5 decision tree learning algorithm [17] to identify relevant attribute subsets and remove the remaining attributes from the dataset. The modified dataset is then presented to a Euclidean nearest neighbour learning algorithm. C4.5 uses a *divide and conquer* approach to inducing decision trees, by recursively determining the attribute that best splits the data into homogeneously classified clusters of instances. As a consequence, many decision trees utilise a subset of the available attributes, which reduces the impact of irrelevant attributes on the target concept². This behaviour has been exploited as an attribute selection mechanism in its own right, with the resulting attributes being tested with other learning algorithms [5].

5 Experimentation and Results

A 20-fold cross validation strategy was used to evaluate the performance of the learning algorithms on eleven numerical datasets (Table 1) from the UCI Machine Learning Database Repository [4]. Several of these datasets each contained an attribute corresponding to a unique identification value. These attributes were removed from the datasets to prevent them affecting the classification accuracy. For example, the *glass* dataset contains an ordered numeric identifier, which is highly correlated with the class (using Spearman’s Rank Correlation, the coefficient is 0.958). To determine the lowest number of dimensions that achieve the highest accuracy, the *CA* and *CACP* algorithms varied the number of dimensions to approximate the sub-space for each dataset between 1 and n , where n was the total number of attributes available for the dataset. The results presented in the tables below refer to those tests that achieved the highest classification accuracy.

Table 1. UCI datasets used in this study

balance	Balance Scale Weight & Distance	bupa	BUPA liver disorders
ionosp	JHU Ionosphere DB	glass	Glass Identification DB
pima	Pima Indians Diabetes DB	iris	Iris Plants DB
sonar	Sonar, Mines vs. Rocks	wine	Wine Recognition Data
wdbc	Wisconsin Diagnostic Breast Cancer	wiscon	Wisconsin Breast Cancer DB
wdbc	Wisconsin Prognostic Breast Cancer		

The results of the 20-fold cross validated tests for the five algorithms are given in Table 2. The results in the second column (NN) represent a baseline result, i.e.

² The selection metrics utilised by decision tree learning algorithms will not necessarily select the optimal set of attributes [2].

the result of the nearest neighbour algorithm when no dimensionality reduction technique is used. The wrapper method, *MC*, succeeded in reducing the number of attributes for ten of the eleven datasets. The number of attributes found for these datasets was typically half that of the original number of attributes. There was a significant increase in classification accuracy for the *iris* dataset (at the 5% confidence level) and *ionosp* dataset (at the 10% confidence level). However, there was a significant decrease in classification accuracy for the *pima* and *wiscon* datasets. No significant difference in classification accuracy was found between *NN* and *MC* for the remaining seven datasets. These results suggest that this wrapper algorithm can successfully reduce the number of attributes in most cases, with little or no loss in classification accuracy, and that in some cases the classification accuracy can increase.

Table 2. Classification accuracies for the UCI datasets for the learning algorithms tested. Results followed by † were significantly different at the 5% confidence level to the baseline (i.e. *NN*) result, whereas those followed by ‡ were significantly different at the 10% confidence level (using a one-tailed t-test in both cases). The number of dimensions selected for each dataset are given in parentheses

	NN	MC	FNN	CA	CACP
bupa	61.98 (6)	↓ 60.38 (4)	61.98 (6)	61.98 (6)	61.98 (6)
ionosp	87.17 (34)	↑ 90.64‡ (14)	↑ 92.60† (9.6)	↑ 90.90† (22)	↑ 91.19† (11)
pima	70.99 (8)	↓ 67.96† (4)	70.99 (8)	70.99 (8)	70.99 (8)
sonar	85.96 (60)	↓ 83.68 (28)	↓ 82.32 (14)	↑ 86.96 (23)	↑ 86.00 (60)
wiscon	95.90 (9)	↓ 95.03‡ (5)	95.90 (6)	↑ 97.36† (6)	↑ 96.19 (3)
wdbc	95.40 (30)	↑ 96.11 (14)	↑ 95.43 (8)	↑ 96.65‡ (5)	↑ 96.29 (16)
wpbc	69.06 (33)	↑ 71.17 (15)	↑ 70.50 (14)	↑ 71.61† (16)	↑ 73.06 (15)
balance	78.10 (4)	78.10 (4)	78.10 (4)	↑ 78.12 (4)	↑ 88.95† (1)
glass	68.09 (9)	↑ 71.00 (5)	68.09 (9)	68.09 (8)	↑ 70.00 (8)
iris	96.16 (4)	↑ 98.13† (2)	↑ 98.13† (2)	96.16 (4)	↑ 96.70 (3)
wine	94.86 (13)	↓ 94.79 (7)	↑ 96.04 (4)	↑ 97.08‡ (6)	↑ 97.64† (6)

The filter method, *FNN*, succeeded in improving the classification accuracy with respect to that achieved by *NN* for five of the eleven datasets. The *iris* dataset is known to contain two relevant attributes (see Figure 2) and two irrelevant attributes [8]. The C4.5 decision trees utilised only the two relevant attributes, and thus *FNN* succeeded in successfully increasing the classification accuracy to 98.13%, whilst halving the number of dimensions used. All four relevant attributes in the *balance* dataset were successfully identified and utilised. Similarly, all the attributes found in the *bupa* and *pima* datasets appeared in the C4.5 decision trees, and as a result, there was no difference in classification accuracy or dimensionality for these datasets. Although there was a drop in classification accuracy for two of the remaining datasets, these results were not significant. The rejection of attributes had no effect on the results for the *wiscon* and *glass* datasets. This suggests that not all the attributes are required

to represent the target hypothesis, and that the rejected attributes may be either irrelevant or redundant.

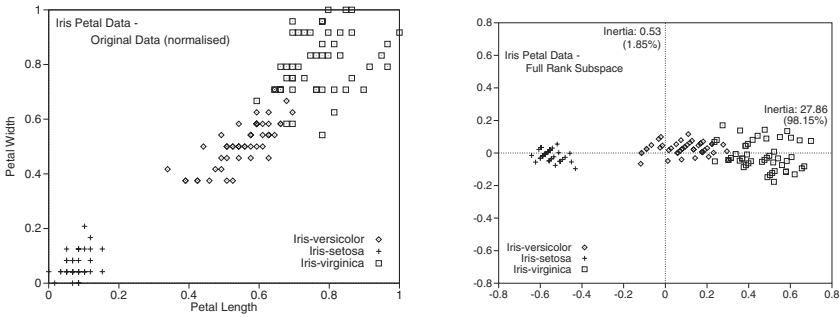


Fig. 2. Mapping the two most relevant attributes of the *iris* dataset into a two dimension sub-space

Both *CA* and *CACP* reduce the number of dimensions required to represent the dataset for six of the eleven datasets. The effects of the two algorithms differed for the *balance*, *iris* and *sonar* datasets: *CA* failed to reduce the dimensionality of *balance* or *iris*; whereas *CACP* failed to reduce the dimensionality of *sonar*. As with *FNN*, neither dataset succeeded in reducing the dimensionality of the *bupa* or *pima* datasets. The sub-space mappings used by *CA* and *CACP* resulted in an increase in classification accuracy for most of the datasets, in addition to reducing the dimensionality. Both methods achieved higher accuracies than either the filter or wrapper methods for six datasets, but in most cases utilised more dimensions.

The result achieved by *CA* for the *balance* dataset suggests that when all the dimensions are present (i.e. no approximation is generated), the sub-space mapping may still affect the classification accuracy of the learning algorithm. This can be illustrated by examining the instance space for the *iris* dataset when only the petal attributes are used, and comparing it with a full rank (i.e. two dimension) sub-space generated by *CA* (Figure 2). In this case, the mapping function performs a rotation and a linear translation. The varying translation of each dimension has the effect of distorting the sub-space with respect to the original space, which is analogous to assigning relevance weights to each dimension.

All four methods (*MC*, *FNN*, *CA* and *CACP*) succeeded in reducing the number of attributes required for the majority of the datasets used in this study. The reductions in dimensionality for each dataset (given as a percentage of the original number of dimensions) are listed in Table 3. *MC* reduced the number of attributes by an average of 44.4%, and *FNN* by an average of 39.2%. In contrast, *CA* and *CACP* only reduced the dimensionality of the datasets by an average of 30.4%, and 36.4% respectively.

Table 3. The number of attributes used by each algorithm and the corresponding reduction in dimensionality (given as a percentage of the original number of dimensions)

	NN		MC		FNN		CA		CACP	
	attrs		attrs	% red.	attrs	% red.	attrs	% red.	attrs	% red.
bupa	6		4	33.3%	6	—	6	—	6	—
ionosp	34		14	58.8%	10	70.6%	22	39.3%	11	67.7%
pima	8		4	50.0%	8	—	8	—	8	—
sonar	60		28	53.3%	14	76.7%	23	61.7%	60	—
wiscon	9		5	44.4%	6	33.3%	6	33.3%	3	66.7%
wdbc	30		14	53.3%	8	73.3%	5	83.3%	16	46.7%
wdbc	33		15	54.6%	14	57.6%	16	51.5%	15	54.6%
balance	4		4	—	4	—	4	—	1	75.0%
glass	9		5	44.4%	9	—	8	11.1%	8	11.1%
iris	4		2	50.0%	2	50.0%	4	—	3	25.0%
wine	13		7	46.2%	4	69.2%	6	53.9%	6	53.9%
Average Reduction			10 datasets		7 datasets		7 datasets		8 datasets	
			44.4%		39.2%		30.4%		36.4%	

The results for the *iris* dataset suggest that the performance of *CA* and *CACP* may degrade in the presence of irrelevant attributes. To investigate this hypothesis, two further datasets were created, consisting of 100 instances each. The datasets each consist of two numeric attributes and a boolean class label. The first dataset comprises of two linearly separable partitions. As *CACP* identifies and utilises class centroids, the second dataset contains four linearly inseparable partitions, two per class. Fifty additional irrelevant attributes were constructed, each containing a single random value for each instance. Various experiments were performed to investigate the behaviour of *CA* and *CACP* in the presence of irrelevant attributes. For each experiment, the two datasets containing the relevant attributes were combined with a random sample of irrelevant attributes, where the random sample increased in size from 0 to 50. Each dataset was then tested with *NN*, *CA* and *CACP*. This was repeated fifteen times for different combinations of irrelevant attributes.

Figure 3 illustrates the results obtained from experiments on the linearly separable dataset. The classification accuracy of all three algorithms falls exponentially, as the number of irrelevant attributes increase. The classification accuracies of both *NN* and *CA* are similar for datasets containing small numbers of irrelevant attributes. However, once the number of irrelevant attributes exceeds 14, the difference in classification accuracy between the two algorithms becomes small but significant (a one-tailed t-test shows significance at the 5% level), with *CA* achieving a slightly higher accuracy than *NN*. The number of dimensions used by *CA* varies as the number of irrelevant attributes in the dataset increases. There is no reduction in dimensionality for datasets with few irrele-

vant attributes. As the number of irrelevant attributes exceeds 8, the number of dimensions selected by *CA* increases slowly from 8 to 29.

The error rate of *CACP* is much lower than that achieved by either *CA* or *NN*. *CACP* achieved a mean accuracy of 74.74% with 49 additional attributes, whereas *CA* and *NN* achieved mean accuracies of 57.47% and 55.93% respectively. The presence of additional irrelevant attributes had little effect on the number of dimensions selected by *CACP* (three to five dimensions in most cases).

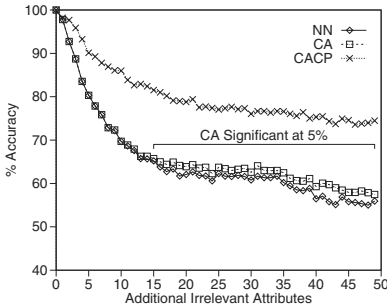


Fig. 3. The effects of additional irrelevant attributes for a linearly separable dataset on three learning algorithms

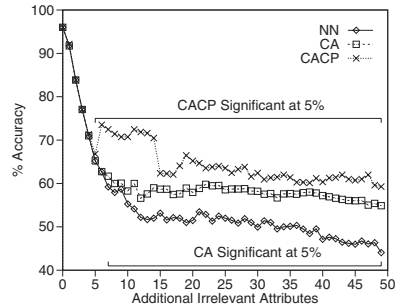


Fig. 4. The effects of additional irrelevant attributes for a linearly inseparable dataset on three learning algorithms

The results for the three algorithms on the linearly inseparable datasets are shown in Figure 4. Although *CACP* achieved superior results for these datasets, the overall performance was much lower than with linearly separable data. However, this drop in accuracy for *CACP* may be due to the proximity of the centroids generated for each class. The initial drop in accuracy exhibited by *NN* is not surprising, as there is an additional boundary separating the points of the two classes, and a small number of points lie along this new boundary. However, the results after the addition of only a few attributes (e.g. 11 attributes) are little better than that achieved by pure chance, indicating that any contribution that the relevant attributes have to any classification hypothesis has been obscured by the effects of the irrelevant attributes. The results show an unusual increase in accuracy for *CACP* for datasets containing between 5 and 14 additional attributes. As yet, no explanation has been found for this behaviour.

The above experiments were repeated to investigate the behaviour of both *CA* and *CACP* in the presence of redundant attributes. In this case, 48 additional attributes were constructed. The values of the additional attributes were calculated in one of several ways: values were copied from one of the dimensions of the original datasets; or values were calculated by inverting one of the dimensions using the function $f(x) = 1 - x$. In addition, some of the attribute values were modified to introduce some variability to the similar dimensions. The func-

tion $f(x) = x \times (1 \pm \text{rnd}(\delta))$ was used, where $\text{rnd}(\delta)$ generates a small random number between 0 and δ ; for this study we used $\delta = 0.05$.

All three algorithms achieved approximately 100% accuracy for the linearly separable dataset and 96.00% for the linearly inseparable dataset. A rank of two was always selected for *CA*, whereas the mean rank varied between one and four for *CACP*.

6 Conclusions

A number of attribute selection techniques that reduce the dimensionality of a dataset have been investigated in recent years. These techniques not only reduce the number of dimensions required to learn a hypothesis, but can result in an increase in classification accuracy. Various filter techniques have been proposed, but studies have shown that by including the learning algorithm in the selection process, better attribute subsets can be found. However, this wrapper approach does not scale up well to problems of more than a few attributes, due to the exponential increase in the size of the search space.

A technique known as Latent Semantic Indexing [7] has been used to reduce the dimensionality of large text-based corpora for some Information Retrieval systems. We have studied the underlying principles upon which LSI is based, and have developed two machine learning algorithms, *CA* and *CACP*, that combine these principles with a nearest neighbour learning algorithm. Both algorithms were found to reduce the number of dimensions required for the majority of datasets studied. In addition, the resulting classification accuracy increased for all but one of these reduced datasets. The techniques used by *CA* and *CACP* identified a new basis for a space that contained the instances in the training set, and then generated a lower dimension approximation to this space. The data points are represented by an attribute-by-instance matrix. Once this matrix has been decomposed, the rank of the matrix can be determined by the resulting diagonal matrix. This rank represents the number of linearly independent, orthogonal dimensions within a sub-space. Therefore, the addition of any duplicate attributes, or any linear combination of attributes will not result in an increase in rank, and so will be eliminated by the decomposition. If two or more attributes contain very similar but not identical values, then there will be additional orthogonal dimensions to express the slight deviations between them. Because the inertia of such dimensions will be small, a lower rank sub-space that excludes these dimensions will closely approximate the original sub-space.

CA and *CACP* appear to be very successful in removing redundant dimensions from the dataset. However, unlike many of the existing attribute selection techniques, they have little impact in reducing the effects of irrelevant attributes. The performance of the class projected variant *CACP* degrades at a slower rate than either *CA* or a simple nearest neighbour in the presence of irrelevant attributes. An investigation is required to determine the behaviour of this approach when used in conjunction with other attribute selection methods, such as weighted methods that identify and eliminate irrelevant attributes, but retain

redundant ones. Further investigations are also required to compare this approach with constructive induction techniques, and more traditional statistical approaches such as Principal Components Analysis.

Acknowledgements

T.Payne acknowledges financial support provided by the UK Engineering & Physical Sciences Research Council (EPSRC).

References

1. D. W. Aha. Tolerating Noisy, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms. *International Journal of Man-Machine Studies*, 36:267–287, 1992. 332
2. H. Almuallim and T. G. Dietterich. Learning With Many Irrelevant Features. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, pages 547–552. MIT Press, 1991. 336
3. M. W. Berry and R. D. Fierro. Low-Rank Orthogonal Decompositions for Information Retrieval Applications. *Numerical Linear Algebra with Applications*, 1(1):1–27, 1996. 334
4. C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. 333, 336
5. C. Cardie. Using Decision Trees to Improve Case-Based Learning. In *Proceedings of the 10th International Conference on Machine Learning*, pages 25–32. San Francisco, CA:Morgan Kaufmann, 1993. 336
6. B. V. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, California:IEEE Computer Society Press, 1991. 335
7. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. 333, 341
8. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973. 337
9. M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. London, UK:Academic Press, 1984. 332, 333, 334
10. G. John, R. Kohavi, and K. Pfleger. Irrelevant Features and the Subset Selection Problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129. San Francisco, CA:Morgan Kaufmann, 1994. 332
11. P. Langley and W. Iba. Average-case Analysis of a Nearest Neighbor Algorithm. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 889–894. San Mateo, CA:Morgan Kaufmann, 1993. 332
12. P. Langley and S. Sage. Induction of Selective Bayesian Classifiers. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Seattle, WA:Morgan Kaufmann, 1994. 332
13. H. Liu and R. Setiono. A Probabilistic Approach to Feature Selection - A Filter Solution. In *Proceedings of the 13th International Conference on Machine Learning*, pages 319–327. San Francisco, CA:Morgan Kaufmann, 1996. 335, 336
14. T. R. Payne. *Dimensionality Reduction and Representation for Nearest Neighbour Learning*. PhD thesis, The University of Aberdeen, Scotland, 1999. 332, 333, 334, 335

15. T. R. Payne and P. Edwards. Dimensionality Reduction through Correspondence Analysis. Technical Report AUCS/TR9910, Department of Computing Science, University of Aberdeen, Scotland., 1999. 333, 334
16. W. H. Press. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. 334
17. J. R. Quinlan. *C4.5 Programs for Machine Learning*. San Mateo, CA:Morgan Kaufmann, 1993. 336
18. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill, 1983. 333
19. C. Wu, M. Berry, S. Shivakumar, and J. McLarty. Neural Networks for Full-Scale Protein Sequence Classification: Sequence Encoding with Singular Value Decomposition. *Machine Learning*, 21:177–193, 1995. 333

Nonparametric Regularization of Decision Trees

Tobias Scheffer

Otto von Guericke University, FIN/IWS
Universitätsplatz 2, 39106 Magdeburg, Germany
`scheffer@iws.cs.uni-magdeburg.de`

Abstract. We discuss the problem of choosing the complexity of a decision tree (measured in the number of leaf nodes) that gives us highest generalization performance. We first discuss an analysis of the generalization error of decision trees that gives us a new perspective on the regularization parameter that is inherent to any regularization (*e.g.*, pruning) algorithm. There is an optimal setting of this parameter for every learning problem; a setting that does well for one problem will inevitably do poorly for others. We will see that the optimal setting can in fact be estimated from the sample, *without* “trying out” various settings on hold-out data. This leads us to a nonparametric decision tree regularization algorithm that can, in principle, work well for all learning problems.

1 Introduction

Decision tree algorithms (*e.g.*, [14,3]) have to solve two distinct problems: they need to identify the size of the tree that leads to optimal generalization performance and, subject to these size constraints, they have to minimize the empirical error rate. The problem of choosing the appropriate tree size is in essence a problem of estimating the misclassification probability of the best decision tree of a given size.

A quick clarification of some notational details is useful for further discussion. Let H_i be the class of all decision trees with exactly i leaf nodes over some fixed set of possible tests. $h \in H_i$ is then a decision tree and maps instances x to class labels y . A learning problem is given by an (unknown) density $p(x, y)$. The generalization error rate of h with respect to this problem (which we want to minimize) is then $\epsilon(h) = \int \sum_y \ell(h(x), y) p(x, y) dx$, where $\ell(\cdot, \cdot)$ is the zero-one loss function. Given a sample S consisting of m independent examples, drawn according to $p(x, y)$, the empirical (or sample) error rate of h is $e(h) = \frac{1}{m} \sum_{(x, y) \in S} \ell(h(x), y)$. It is important to distinguish between generalization error ϵ (which we really *want* to minimize) and empirical error e (which we *are able* to measure and minimize using the sample) throughout this paper.

Many decision tree algorithms try to minimize the generalization error by minimizing a regularization function $f(e(h), c(H_i))$ that depends on the empirical error $e(h)$ and some complexity measure $c(H_i)$ of the hypothesis class H_i which the hypothesis tree h came from. In other words, the complexity (or size) of the decision tree is getting penalized. Technically, this is often realized by

employing some pruning rule that trades off a lower empirical error rate against the number of branches required to achieve this gain in empirical accuracy. Such complexity regularization techniques will in fact lead to a low generalization error rate *if and only if* $f(e(h), c(H_i))$ is a “reasonable” estimate of $\epsilon(h)$, the generalization error rate. This raises the question whether *there is* a regularization function $f(e, c(H_i))$ that maps an empirical error rate and some measure of the complexity of a class of decision trees H_i to a “reasonable” estimate of $\epsilon(h)$.

We can use (PAC-style) Chernoff bounds to bound the greatest possible difference between true and empirical error rate of any hypothesis in H_i and then conclude that, with high probability, the generalization error rate of h is no more than $e(h) + \text{bound}(m, |H_i|)$, where $|H_i|$ is the size (alternatively, the VC-dimension) of H_i . However, the actual error rate may lie *anywhere* between 0 and the worst-case bound, depending on characteristics of the given learning problem. We can easily construct two learning problems (one with an error rate that increases steeply when $|H_i|$ grows and one with a slowly increasing error curve) such that any regularization function $f(e, c(H_i))$ fails (*i.e.*, incurs an additional error of $\lambda > 0$ that does not vanish when the sample size grows) for at least one of them [9]. This means that the empirical error rate $e(h)$ and the complexity of H_i do not suffice to determine the actual error rate; some information is missing. Obviously, we can determine a near optimal setting for the regularization parameter for each single problem by trying out many values and assessing the resulting decision tree on holdout data. Alternatively, we can use cross validation to select the optimal number of leaves in the first place, like, for instance, the CART algorithm does [3]. The primary disadvantage of n -fold cross validation [19] lies in its unsatisfactory efficiency caused by the necessity of invoking the learner n times for each considered number of leaves.

We will pursue a different approach. We will take a careful look at the generalization error rate of decision trees and study just what information regularization functions are missing – *i.e.*, what other information than $e(h)$ and the complexity of H_i do we need to obtain a reliable estimate of $\epsilon(h)$ for all possible problems, without assessing hypotheses on holdout data. We will identify this missing information and discuss how it can be acquired efficiently in many cases. In Section 2, we simplify the expected error analysis of [18] slightly and apply it to the problem of choosing the optimal decision tree complexity. The original analysis is restricted to exhaustive learners while decision tree algorithms are usually greedy. Our main theoretical result (Section 3) is an extension of the analysis to greedy learning algorithms. In Section 4 we discuss a nonparametric regularization algorithm which we study empirically in Section 5.

2 Error Rate of Exhaustive Decision Tree Learners

In this section, we assume that, given a number i of leaf nodes, the learning algorithm determines the hypothesis h_i^L with least *empirical* error that has exactly i leaf nodes. When there are several hypotheses with the same low empirical error, we assume the learner to break ties by drawing at random under uniform

distribution. Let us assume that the sample size m is fixed and given in advance, whereas the sample S itself is a random variable, governed by the distribution $(p(x, y))^m$. We will now study $E(\epsilon(h_i^L)|H_i, m)$, the expected *generalization* error of the returned hypothesis h_i^L given the sample size m and the number of leaves i . In order to determine the expected true error (expected over all samples) of h_i^L (the decision tree with i leaf nodes that incurs the least empirical error rate), we factorize the hypothesis h that the learner returns (Equation 1). Since we assume the learner to break ties between hypotheses with equally small empirical error at random, all hypotheses with equal true error rates ϵ have an exactly equal prior probability of becoming h_i^L . We re-arrange Equation 1 such that all hypotheses h_ϵ with true error ϵ are grouped together. $\pi(\epsilon|H_i)$ is the density of decision trees with error rate ϵ among all the decision trees with i leaf nodes (with respect to the given learning problem). Intuitively, if we would draw a decision tree with i leaf nodes at random under uniform distribution from all decision trees H_i , $\pi(\epsilon|H_i)$ would be the chance of the resulting decision tree incurring an error rate of ϵ for the given problem. This takes us to Equation 2.

$$E(\epsilon(h_i^L)|H_i, m) = \int_h \epsilon(h)P(h_i^L = h|H_i, m)dh \quad (1)$$

$$= \int_\epsilon \epsilon P(h_i^L = h_\epsilon|\epsilon, H_i, m)\pi(\epsilon|H_i)d\epsilon \quad (2)$$

Let $H_i^* = \operatorname{argmin}_{h \in H_i} \{e(h)\}$ be the set of hypotheses in H_i which incur the least empirical error rate with respect to some sample S . Note that H_i^* is a random variable because only the sample size m is fixed whereas the sample S itself (on which H_i^* depends) is a random variable. In order to determine the chance that h_ϵ (an arbitrary hypothesis with true error rate ϵ) is selected as h_i^L , we first factorize the chance that h_ϵ lies in H_i^* , the empirical error minimizing hypotheses of H_i (Equation 3). A hypothesis that does not lie in H_i^* has a zero probability of becoming h_i^L (Equation 4). In Equation 5, we factorize the cardinality of $|H_i^*|$. When this set is of size n , then each hypothesis in H_i^* has a chance of $\frac{1}{n}$ of becoming h_i^L (the learner breaks ties at random) (Equation 6). In Equation 7, we factorize the least empirical error e and, in Equation 8, we simply split up the conjunction (like $p(a, b) = p(a)p(b|a)$).

$$\begin{aligned} P(h_L = h_\epsilon|\epsilon, H_i, m) \\ = P(h_L = h_\epsilon|H_i, m, h_\epsilon \in H_i^*)P(h_\epsilon \in H_i^*) \end{aligned} \quad (3)$$

$$\begin{aligned} + P(h_L = h_\epsilon|H_i, m, h_\epsilon \notin H_i^*)(1 - P(h_\epsilon \in H_i^*)) \\ = P(h_L = h_\epsilon|H_i, m, h_\epsilon \in H_i^*)P(h_\epsilon \in H_i^*) \end{aligned} \quad (4)$$

$$= \sum_n P(h_L = h_\epsilon|H_i, m, h_\epsilon \in H_i^*, |H_i^*| = n)P(h_\epsilon \in H_i^*, |H_i^*| = n) \quad (5)$$

$$= \sum_n \frac{1}{n} P(h_\epsilon \in H_i^*, |H_i^*| = n) \quad (6)$$

$$= \sum_e \sum_n \frac{1}{n} P(h_\epsilon \in H_i^*, |H_i^*| = n | e(h_\epsilon) = e)P(e(h_\epsilon) = e|\epsilon, m) \quad (7)$$

$$= \sum_e \sum_n \frac{1}{n} P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) P(|H_i^*| = n | h_\epsilon \in H_i^*, e(h_\epsilon) = e) P(e(h_\epsilon) = e | \epsilon, m) \quad (8)$$

By inserting Equation 8 into Equation 2 we get

$$\begin{aligned} E(\epsilon(h_L) | H_i, m) \\ = \int_\epsilon \epsilon \left(\sum_e \sum_n \frac{1}{n} P(|H_i^*| = n | h_\epsilon \in H_i^*, e(h_\epsilon) = e) P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) \right. \\ \left. P(e(h_\epsilon) = e | \epsilon, m) \pi(\epsilon | H_i) \right) d\epsilon \end{aligned} \quad (9)$$

Assuming that the chance of the set of empirical error minimizing hypotheses H_i^* being of size n when h_ϵ is known to lie in this set does not depend on *which* hypothesis is known to lie in this set (formally, $P(|H_i^*| = n | h_1 \in H_i^*) = P(|H_i^*| = n | h_2 \in H_i^*)$ for all h_1, h_2) we can claim that $const = P(|H_i^*| = n | h_\epsilon \in H_i^*, e(h_\epsilon) = e)$ is constant for all h_ϵ . Equation 10 specifies the expectation of $\epsilon(h_L)$. The density $p(\epsilon(h_L))$ has to integrate to 1. $const$ is therefore a normalization constant which is determined uniquely.

$$const = \left(\int_\epsilon \sum_e P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) P(e(h_\epsilon) = e | \epsilon, m) \pi(\epsilon | H_i) d\epsilon \right)^{-1} \quad (11)$$

Combining Equations 10 and 11 we obtain

$$\begin{aligned} E(\epsilon(h_L) | H_i, m) \\ = \frac{\int_\epsilon \epsilon \left(\sum_e P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) P(e(h_\epsilon) = e | \epsilon, m) \pi(\epsilon | H_i) \right) d\epsilon}{\int_\epsilon \sum_e P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) P(e(h_\epsilon) = e | \epsilon, m) \pi(\epsilon | H_i) d\epsilon} \end{aligned} \quad (12)$$

Let us now tackle the last unknown term, $P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m)$. A hypothesis h_ϵ (with true error rate ϵ) lies in H_i^* when no hypothesis in H_i achieves a lower empirical error rate. There are $|H_i|$ many hypotheses; their true error rates are fixed but completely arbitrary – *i.e.*, they are neither independent nor governed by some identical distribution. These $|H_i|$ error rates constitute the density $\pi(\epsilon | H_i)$ which measures how often each error rate ϵ occurs in H_i (we have already seen this density in Equation 2). Each of these hypotheses incurs an empirical error rate that is by itself governed by the binomial distribution $B[m, \epsilon]$. (Each example can be classified correctly or erroneously; the chance of the latter happening is ϵ ; this leads to a binomial distribution). Let us assume that the empirical error rates of two or more hypotheses are independent *given the corresponding true error rates*. Formally, $P(\bigwedge_{h_j \in H_i} e(h_j) | \epsilon(h_j)) = \prod_{h_j \in H_i} P(e(h_j) | \epsilon(h_j))$. Now we can quantify the chance that no hypothesis incurs an error of less than e which makes our hypothesis h with $e(h) = e$ a member of H_i^* . For all but extremely small H_i (formally, $p^{|H_i|} \approx p^{|H_i|-1}$) we can write this chance as

$$P(h_\epsilon \in H_i^* | e(h_\epsilon) = e, m) = \prod_{\epsilon'} P(e(h) \geq e | \epsilon', m)^{|H_i| \pi(\epsilon' | H_i)}. \quad (13)$$

Finally, let us determine $|H_i|$, the number of decision trees with i leaf nodes. With c classes and a total of n possible tests available, there are exactly $|H_i| = \tau(i) \times c^i$ decision trees with i leaf nodes ($\tau(i)$ is the number of “trunks” and there are c^i labelings of the leaf nodes), where $\tau(1) = 1$ and $\tau(i) = \sum_{j=1}^{i-1} n \times \tau(j) \times \tau(i-j)$. Intuitively, at each test node there are n possible tests and j of the remaining i leaf nodes can be placed in the left subtree while the remaining $j-i$ leaf nodes go into the right subtree (for all possible j between 1 and $i-1$).

What have we achieved so far? Equations 12 and 13 quantify the expected generalization error of h_i^L for a given problem in terms of three quantities: the number of decision trees $|H_i|$ (can easily be computed), the sample size m (which is known), and the density of error rates in H_i , $\pi(\epsilon|H_i)$. Note that, for Equations 12 to give us the expected error $\epsilon(h_i^L)$, it is not necessary to actually run the learner and determine $e(h_i^L)$. Let us also emphasize that we are not talking about *bounds* on the error rate for a class of possible problems. Subject to the mentioned independence assumptions, Equations 12 and 13 quantify *the* expected generalization error of an empirical error minimizing hypothesis *for a particular, given learning problem*. When only the sample size m and $|H_i|$ are given, it is impossible to determine where in the interval specified by the Chernoff bound the actual error rate lies which motivates the negative result of Kearns *et al.* [9] on the performance of complexity regularization algorithms. Additionally given the density $\pi(\epsilon|H_i)$, however, we can determine the *actual* density that governs the generalization error, and thereby also the expected generalization error. We have therefore identified the information that complexity penalization algorithms are missing as being $\pi(\epsilon|H_i)$. If there was a feasible way to estimate $\pi(\epsilon|H_i)$ we could construct a regularization algorithm that uses this additional information and circumvents the negative result of Kearns *et al.*. But before we discuss how π can be estimated, let us look at the generalization error of greedy decision tree learners.

3 Greedy Decision Tree Algorithms

The solution presented so far quantifies the generalization error of the decision tree with i leaf nodes that incurs the least empirical error with respect to a sample of size m . Hence, the analysis applies to exhaustive learners that are able to always find the empirical error minimizing hypothesis. However, when the problem requires the decision tree to have many nodes, exhausting the space of all decision trees with that number of nodes may not be feasible and a greedy algorithm (that cannot be guaranteed to find the decision tree with least empirical error) may have to be employed. We will now discuss the expected generalization error of a hypothesis with an empirical error rate of e , (found, for instance, by a greedy learner) which may be distinct from the hypothesis with the globally smallest empirical error rate. The following solution depends additionally on the empirical error rate e of the hypothesis returned by the learner. This means that we have to run the greedy learner and determine the resulting training set error which was not necessary in the exhaustive analysis.

Let H_i^e be the subset of H_i with empirical error of $e(h) = e$. Let h_i^e be a hypothesis drawn from H_i^e at random under uniform distribution – *i.e.*, h_i^e is an arbitrary hypothesis with empirical error rate of e . We start off by factorizing the hypothesis which the learner chooses as h_i^e (Equation 14). Similarly to Equation 2, in Equation 15 we factorize the error rate ϵ , forming (for each ϵ) “subgroups” of $\pi(\epsilon|H_i)$ hypotheses with equal error rate ϵ . In Equation 16 we factorize the empirical error rate of h_i^e and then say that all empirical error rates have probability zero, except for the value e which has a probability of 1. In Equation 17, we factorize the cardinality of H_i^e and, in Equation 18, we claim that the chance of a hypothesis h_e in H_i^e being selected as h_i^e is $\frac{1}{n}$ when $|H_i^e| = n$ (remember that we assumed h_i^e to be drawn at random from H_i^e).

The probability of H_i^e being of size n when we know already that one hypothesis (h_i^e) is in this set is equal to the chance of H_i^e being of size $n - 1$ (Equation 19), and the empirical error rate of h_e is governed by the binomial distribution with mean value ϵ . Now note that the sum over n in Equation 20 does not depend on ϵ any more – *i.e.*, it is a constant. Since $p(\epsilon(h_i^e)|H_i, m, e)$ has to integrate to 1 can simply normalize the expectation in Equation 21 like we did in Equation 11. Now only $\pi(\epsilon|H_i)$ remains which means that we are done.

$$\begin{aligned} E(\epsilon(h_i^e)|H_i, m, e) \\ = \int \epsilon(h) P(h_i^e = h|H_i, m, e) dh \end{aligned} \quad (14)$$

$$= \int \epsilon P(h_i^e = h_e|\epsilon, H_i, m, e) \pi(\epsilon|H_i) d\epsilon \quad (15)$$

$$= \int \epsilon P(h_i^e = h_e|\epsilon, H_i, m, e) P(e(h_e) = e) \pi(\epsilon|H_i) d\epsilon \quad (16)$$

$$\begin{aligned} = \int \epsilon \sum_n P(h_i^e = h_e | |H_i^e| = n, \epsilon, H_i, m, e) \\ P(e(h_e) = e, |H_i^e| = n) \pi(\epsilon|H_i) d\epsilon \end{aligned} \quad (17)$$

$$= \int \epsilon \sum_n \frac{1}{n} P(e(h_e) = e) P(|H_i^e| = n | e(h_e) = e) \pi(\epsilon|H_i) d\epsilon \quad (18)$$

$$= \int \epsilon \sum_n \frac{1}{n} P(e(h_e) = e) P(|H_i^e| = n - 1) \pi(\epsilon|H_i) d\epsilon \quad (19)$$

$$= \int \epsilon B[\epsilon, m](e) \left(\sum_n \frac{1}{n} P(|H_i^e| = n - 1) \right) \pi(\epsilon|H_i) d\epsilon \quad (20)$$

$$= \frac{\int \epsilon B[\epsilon, m](e) \pi(\epsilon|H_i) d\epsilon}{\int B[\epsilon, m](e) \pi(\epsilon|H_i) d\epsilon} \quad (21)$$

We have now found a solution that quantifies $E(\epsilon(h_i^e)|H_i, m, e)$, the expected generalization error of a decision tree with i leaf nodes and empirical error rate e for a given learning problem $p(x, y)$. In contrast to the result of Section 2, h_i^e is not assumed to be the result of an exhaustive learner, it can be the outcome of a greedy learner. This time, the solution depends on the empirical error e (*i.e.*,

we need to run the learner to determine the training set error), the density of error rates in the set of decision trees with i leaf nodes, $\pi(\epsilon|H_i)$, and the sample size m , but it does not depend on $|H_i|$. Again, Equation 21 specifies the actual error rate for the given learning problem rather than a worst-case bound that holds for all possible learning problems (which PAC theory does). The additional information of $\pi(\epsilon|H_i)$ makes this possible.

4 Nonparametric Decision Tree Regularization

Decision tree pruning algorithms minimize a regularization function $f(\epsilon(h_i^L), c(H_i))$ (depending on the empirical error rate and some complexity measure of H_i) which has to be a good estimate of $\epsilon(h_i^L)$ if we want to minimize the right quantity. However, the influence of the complexity on the error has to be weighted and this weight has to be chosen for each problem. If, however, $\pi(\epsilon|H_i)$ was known, then we could construct a decision tree learner that minimizes Equation 12 (for exhaustive learning) or Equation 21 (for greedy learning), respectively. We then have a regularization function that, in principle, should work well for all possible learning problems without having a parameter.

$\pi(\epsilon|H_i)$ cannot be measured directly since it depends on $p(x, y)$ which is unknown. However, there is an empirical counterpart $\pi(e|H_i)$ (the density of empirical error rates of hypotheses in H_i with respect to the sample S) which we can record when H_i is known and a sample S is available. We can obtain $\pi(e|H_i)$ by repeatedly drawing hypotheses from H_i under uniform distribution, or by conducting a Markov random walk in the hypothesis space with the uniform distribution as stationary distribution [7, 12]. While the general problem of estimating densities is very hard, the situation is not quite as bad in our special case. Like $\pi(\epsilon|H_i)$, $\pi(e|H_i)$ is one-dimensional, but is furthermore discrete since there are only $m + 1$ possible empirical error rates when m is the sample size. How many hypotheses of H_i do we have to look at in order to obtain a reliable estimate of π ? We want to estimate m probabilities; suppose that we want none of these estimates to be off by more than some ε with high probability $(1 - \delta)$. In this case, we need to draw $\frac{1}{2\varepsilon^2} \log \frac{m}{\delta} = O(\log m)$ hypotheses which is not particularly much. Although drawing $O(\log m)$ hypotheses will typically suffice for an accurate estimate of $\epsilon(h_i^L)$, there are cases in which a misestimation of π by some small ε can lead to an inaccurate estimate of $\epsilon(h_i^L)$. In this theoretical worst-case, estimating π sufficiently accurately can be as difficult as running a learning algorithm, see [15] for a more detailed discussion. We can now describe a decision tree algorithm that uses the expected error analysis to regularize the decision tree complexity.

Algorithms QDT and Greedy-QDT.

1. **For** $i = 1 \dots \text{maxleaves}$.

- (a) Draw $O(\log m)$ decision trees with i leaf nodes and record their empirical error rates, thus measuring $\pi(e|H_i)$ which will serve as an estimate of $\pi(\epsilon|H_i)$.

- (b) For exhaustive **QDT**: Evaluate Equation 12 to determine the estimated expected error of $\epsilon(h_i^L)$.
- (c) For **Greedy-QDT**: Minimize the empirical error greedily using exactly i leaf nodes, the resulting hypothesis is h_i^L . Determine the empirical error $e(h_i^L)$ on the training set. Evaluate Equation 21 to obtain an estimate of the expected generalization error of h_i^L .
- 2. Let i^* be the number i which minimizes the estimated expected generalization error of h_i^L determined in step 1b or 1c, respectively.
- 3. For exhaustive **QDT**: Exhaust the space of decision trees with i^* leaf nodes (this takes $O(n^{i^*})$). The resulting tree is $h_{i^*}^L$.
- 4. For **Greedy-QDT**: $h_{i^*}^L$ has already been determined.
- 5. **Return** $h_{i^*}^L$.

For a given number of leaf nodes, we use the following algorithm to minimize the empirical error rate. When we set the threshold to i , the algorithm is almost exhaustive while with a threshold of 1 it is completely greedy. We use the information gain heuristic; note, however, that our complexity regularization method can be “plugged” into almost any greedy or exhaustive decision tree learner.

Algorithm EmpiricalErrorMinimization

- 1. **Input**: Number i of nodes, **Output**: decision tree with least empirical error.
- 2. **If** $i = 1$ return leaf node with class label that minimizes the empirical error rate.
- 3. **For** all attributes a ,
 - (a) Find optimal split for the given attribute a ,
 - (b) **If** $i > \text{threshold}$ commit to the split. **Otherwise** backtrack to find the globally optimal split.
 - (c) **If** $i > \text{threshold}$ **Then Let** $\text{left} : \text{right} = H_{\text{left}} \times \#$ of instances in the left branch : $H_{\text{right}} \times \#$ of instances in right ranch (split the number of remaining nodes according to the remaining entropy weighted by the number of instances in the left and right branch).
 - (d) **Otherwise** backtrack to find optimal values for left and right .
 - i. Determine left and right subtree by invoking **EmpiricalErrorMinimization** recursively with left and right as desired number of leaf nodes, and with the corresponding subset of the sample.

Some technical details are left for the full paper, due to lack of space. An algorithm that records the error rates of $n \times |Y|^i$ decision trees in $O(n \times i \times m)$ (required for step 1a) is described in [15].

5 Empirical Studies

In order to select an appropriate number of leaf nodes, the QDT algorithm has to be able to predict the error rate in dependence of the number of leaf nodes used. Therefore, in the first part of the empirical studies, we will study how the

predicted error rate (depending on the number of leaves) relate to the error rates measured by n -fold cross validation.

Is the error rate predicted accurately? We drew 8 learning problems that have little (or no) missing values at random from the UCI repository [2]. For each problem and every number of leaf nodes i , we estimate the density of error rates $\pi(\epsilon|H_i)$ in $O(i \times m)$. For the exhaustive version (results in Figure 1), we evaluate Equation 12 to obtain the predicted generalization error. For the greedy version (results in Figure 2), we run the greedy learner, measure the empirical error e and evaluate Equation 21 to get the predicted generalization error. We then run a 10-fold cross validation loop (for each number i). In each fold, we run the exhaustive/greedy learner and estimate the generalization error using the holdout set (the exhaustive learner is not completely exhaustive; due to the high computational costs only subtrees of up to four leaves are searched exhaustively). Figure 1 compares the predicted to the measured generalization error rates for the exhaustive learner and Figure 2 for the greedy learner. For most measurements, the predicted value lies within the standard deviation of the measured value which indicates that the predictions are accurate. Note that, even if all predictions were totally accurate, 14% of all predictions would lie outside their standard deviation. Only for the Cleveland and *E. Coli* problem we can see significant deviations; but there is no case in which relying on the prediction would result in selecting a number of leaves that is significantly suboptimal. In some cases, the greedy analysis appears to give just slightly more accurate predictions. This might be due to the fact that the greedy analysis gets to know the resulting empirical error rate as additional information. In many cases, the exhaustive learner achieves a slightly (not significantly) lower generalization error.

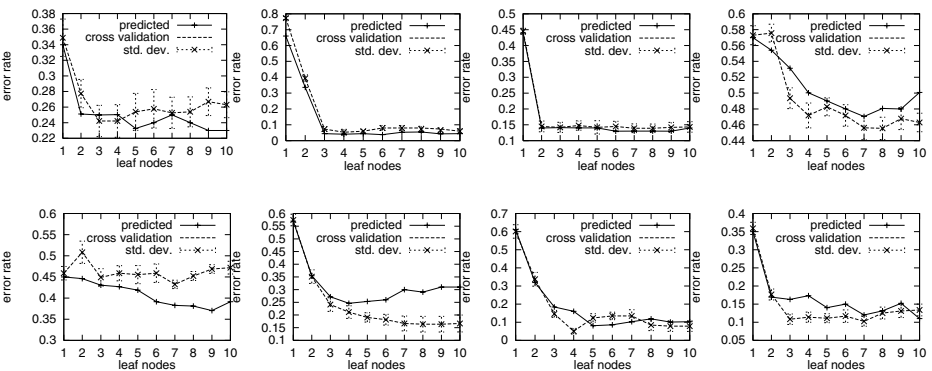


Fig. 1. Predicted (expected error analysis) and measured (10-fold cross validation) generalization error rates of decision trees restricted to i leaf nodes. (a) diabetes, (b) iris, (c) crx, (d) cmc, (e) cleveland, (f) ecoli, (g) wine, (h) ionosphere

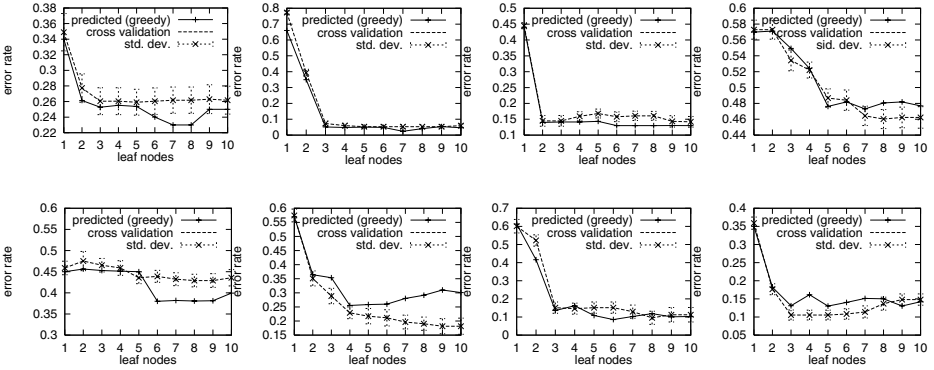


Fig. 2. Predicted and measured (10-fold cross validation) generalization error rates of a greedy decision tree learner, restricted to i leaf nodes. (a) diabetes, (b) iris, (c) crx, (d) cmc, (e) cleveland, (f) ecoli, (g) wine, (h) ionosphere

Does QDT better or worse than cross validation based learning? We will now study how our regularization procedure compares to cross validation based pruning (*e.g.*, [3]). We wrap the learner into an outer layer of 10-fold cross validation. In this “wrapper”, for every number of leaf nodes i we first evaluate Equation 12 to estimate which number of leaves would be optimal. We then run an inner loop of n -fold cross validation to find out which number of leaf nodes the cross validation based learner would select. We assess both recommended numbers of leaf nodes in the outer cross validation wrapper. We try this for various n . Figure 3 shows the results. Surprisingly, using 10-fold cross validation is in no case significantly better than using one fold (training and test with a split ratio of 70%). In some cases (*e.g.*, wine) this might still be the case but the differences are not significant. The differences between QDT and 10-fold cross validation based selection of the number of leaf nodes are not significant – in other words, our analysis determines the optimal number of leaf nodes just as good as n -fold cross validation (for the studied problems).

6 Discussion and Related Work

Kearns and Mansour [8] proposed a nonparametric Chernoff-based pruning rule. Their rule removes all subtrees *unless* it can prove that the subtree really enhances the generalization performance. Therefore, the algorithm has a bias towards over-pruning the tree. Bayesian or MDL-based pruning strategies (*e.g.*, [11]) can also be seen as not being parametric. But they require additional information in terms of the prior probability of target densities ($p(p(x, y))$, in our formalism). In a way, this prior contains even more information than $\pi(\epsilon|H_i)$ because it tells something about all learning problems whereas $\pi(\epsilon|H_i)$ is a property of a certain given learning problem. Our experiments may slightly strengthen

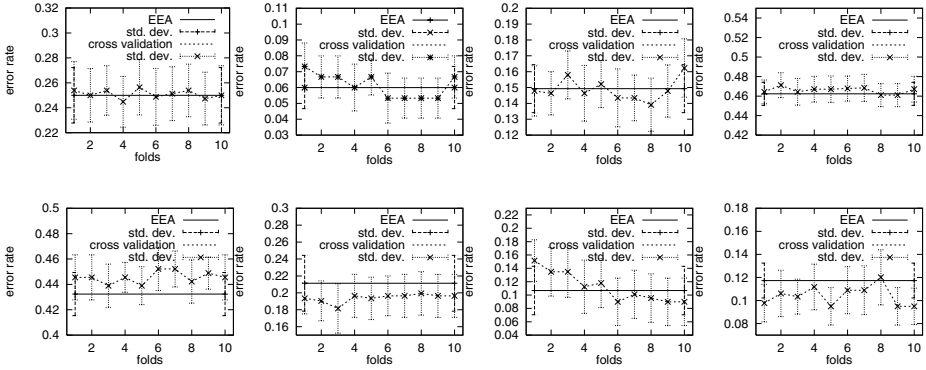


Fig. 3. Error rate (estimated by 10-fold cross validation) when either expected error analysis or n -fold cross validation is used to determine the number of leaf nodes. (a) diabetes, (b) iris, (c) crx, (d) cmc, (e) cleveland, (f) ecoli, (g) wine, (h) ionosphere

the belief (but do not prove) that exhaustive learning improves generalization slightly over greedy learning. (Relatively) fast exhaustive decision tree learners that are restricted to balanced trees have been presented in [1,4]. Unfortunately, exhaustive learning is not feasible when the largest considered tree possesses many leaf nodes. However, our experiments show that a near optimal number of leaf nodes can be determined by means of our analysis as accurately as by 10-fold cross validation in $O(n \times i_{max} \times m)$ (while exhaustive tree learning is exponential in i_{max}). Our analysis predicts the generalization error rate of an exhaustive learner even when it would be far too expensive to actually run the learner. This opens the opportunity to determine the optimal number of leaf nodes very efficiently using our analysis and to invoke an exhaustive learner in case the optimal number of leaves is small. The speed-up that our regularization procedure achieves compared to n -fold cross validation is exponential when the underlying learner is exhaustive, and is roughly a factor of n for greedy learners.

The analysis of the generalization error of the empirical error minimizing decision tree with i leaf nodes opens some new insights. Regularization algorithms that penalize the complexity of decision trees cannot directly minimize the generalization error $\epsilon(h_i^L)$ because empirical error and complexity do not suffice to infer the generalization error – therefore they possess a parameter that has to be adjusted for all problems (e.g., [13,20,14]). However, we have seen that the missing information is contained in $\pi(\epsilon|H_i)$, a density that can often be estimated for a given i in $O(\log m)$ when a sample is given. Our analysis is an *actual-case* analysis (for a given learner and a given learning problem), rather than a (PAC-style) *worst-case* analysis (for the worst possible problem). Compared to earlier *actual case* analyses [17,16,5] our analysis is based on weaker assumptions. Compared to [18], our analysis is considerably simpler and, most

importantly, covers greedy learners. An actual case analysis for Naive Bayesian classifiers that is guided by a similar idea has been presented by Langley and Sage [10], an actual case analysis for linear neural networks is given in [6].

Acknowledgment

Thanks to Ross Quinlan for an inspiring discussion at last year's machine learning conference. Thanks to Stefan Wrobel for helpful comments.

References

1. Peter Auer, Robert C. Holte, and Wolfgang Maass. Theory and applications of agnostic PAC-learning with small decision trees. In *12th ICML*, pages 21–29, 1995. 354
2. C. Blake and C. Merz. UCI repository of machine learning data sets. <http://www.ics.uci.edu/~mllearn/MLRepository.html>, 1999. 352
3. L. Breiman, J. Friedman, R. Ohlsen, and C. Stone. *Classification and Regression Trees*. Pacific Grove, 1984. 344, 345, 353
4. D. Dobkin, D. Gunopoulos, and S. Kasif. Computing optimal shallow decision trees. In *Proc. International Workshop on Mathematics in Artificial Intelligence*, 1996. 354
5. P. Domingos. Process-oriented estimation of generalization error. In *IJCAI-99*, 1999. 354
6. K. Fukumizu. Generalization error of linear neural networks in unidentifiable cases. In *Algorithmic Learning Theory*, 1999. 355
7. W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995. 350
8. M. Kearns and Y. Mansour. A fast, bottom-up decision tree pruning algorithm with near optimal generalization. In *ICML-98*, pages 269–277, 1998. 353
9. M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning Journal*, 27:7–50, 1997. 345, 348
10. P. Langley and S. Sage. Tractable average case analysis of naive bayes classifiers. In *ICML-99*, pages 220–228, 1999. 355
11. M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 216–221, August 1995. 353
12. N. Metropolis, A. Rosenbluth, M. Teller, and E. Teller. Equations of state calculations by fast computing machine. *J. Chem. Phys.*, 21:1087–1091, 1953. 350
13. J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989. 354
14. J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publisher, 1993. 344, 354
15. T. Scheffer. *Error Estimation and Model Selection*. Infix, 1999. 350, 351
16. T. Scheffer and T. Joachims. Estimating the expected error of empirical minimizers for model selection. Technical Report TR 98-9, Technische Universitaet Berlin, 1998. 354

17. T. Scheffer and T. Joachims. Estimating the expected error of empirical minimizers for model selection (abstract). AAAI-98, 1998. 354
18. T. Scheffer and T. Joachims. Expected error analysis for model selection. In *ICML-99*, 1999. 345, 354
19. G. Toussaint. Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory IT20*, 4:472–479, 1974. 345
20. S. Weiss and N. Indurkha. Small sample decision tree pruning. In *Proc. ICML-94*, pages 335–342, 1994. 354

An Efficient and Effective Procedure for Updating a Competence Model for Case-Based Reasoners

Barry Smyth and Elizabeth McKenna

Department of Computer Science, University College Dublin,
Belfield, Dublin 4, Ireland
{Barry.Smyth,Elizabeth.McKenna}@ucd.ie

Abstract. Case-based reasoning systems solve new problems by reusing previous problem solving experience stored as cases in a case-base. In recent years the maintenance problem has become an increasingly important research issue for the case-based reasoning community. In short, the goal is to develop strategies for effectively maintaining the efficiency and competence of case-based reasoning systems as they evolve. Our research has focused on the development of a model of competence for case-based reasoning systems, a model that measures the contributions of individual cases to overall system competence, and which forms the computational basis for a variety of maintenance strategies. However, while this model offers many potential advantages its upkeep adds an additional cost to the CBR cycle. In this paper we evaluate a new method for more efficiently updating the model at run-time.

1 Introduction

Case-based reasoning (CBR) systems solve new problems by retrieving and adapting the solutions of similar problems stored as cases in a case-base [3,4,10]. The CBR method has been successfully adapted for a wide variety of tasks (classification, diagnosis, prediction, planning, and design) across a wide variety of domains (for example, fraud detection, property valuation, route planning, and software design).

An important issue facing the CBR community involves the ongoing maintenance of CBR systems [5,7,8,11,12,13,14,15]. The maintenance problem focuses on the issue of how best to manage the organisation and contents of a case-base in order to optimise future reasoning performance with respect to an agreed set of performance objectives. In practice, this involves the development of a range of policies for controlling the growth of case-bases and the organisation and indexing of the cases themselves. For example, a variety of case addition and deletion policies have been proposed to manage case-base growth [9,11,13,15]. Ultimately these procedures encode judgements about the importance of cases with respect to the efficiency and competence of a given case-based reasoner, and then use these judgements to prioritise cases for addition or deletion.

Zhu and Yang [15] describe a procedure for case addition that uses a probabilistic estimate of competence (see [13] for a related approach). Smyth & Keane [11] have looked at techniques for deleting cases (mainly as a way of coping with the

deleterious performance effects of the so-called *utility problem* [2,6,9]). They pre-categorise the competence contributions of individual cases in order to prioritise cases for deletion with respect to competence, thereby guaranteeing the preservation of system competence during deletion. In contrast, coming from a speed-up learning background, Minton [6] proposes a technique for estimating the efficiency of knowledge items (cases) as a guide for an efficiency-based deletion criterion. Again, this is a coping strategy for dealing with the utility problem, but it is found to be inappropriate for CBR because it does not respect competence during deletion.

These approaches all focus on one particular aspect of the maintenance problem, for example case addition or deletion, and the result is a collection of solutions that are limited to particular instances of the more general maintenance problem. Furthermore, these solutions operate on the basis of key assumptions and heuristics about the nature of competence and efficiency in case-based reasoners, but they fail to reveal the full picture. As the core motivation for our research, we argue that any effective and generic solution to the maintenance problem will only come about by developing a more complete understanding of the true nature of efficiency and competence for case-based reasoning systems. In short, we propose the development of an explicit theory or model of competence to act as the foundation for maintenance solutions. We argue that such models will provide us with access to more accurate and effective measures of the “worth” or competence of a case in order to better inform future case deletion or addition strategies.

One such competence model is presented and evaluated in [12] where we show that it provides accurate predictions of true competence under a variety of operational conditions. In recent work we have also shown how this competence model can be used as the computational basis for a host of innovative approaches to case addition and deletion, case retrieval, authoring support and case-base visualisation [12,13,14]. However, while this model has proved to be effective at estimating case competence, as the case-base grows its update adds an additional cost to the CBR cycle. In this paper we explain how to significantly reduce this cost by presenting a novel procedure for more efficiently updating the competence model at run-time as new cases are learned (Section 3). In Section 4 we fully evaluate this new update procedure and in Section 5 we explain how further cost savings are possible by integrating the update procedure with an existing competence-guided retrieval technique; in fact, we argue that under certain conditions the competence update essentially comes for free as a side-effect of retrieval. We begin with an outline description of our existing competence model.

2 A Model of Case Competence

Our competence model can be best understood in terms of four distinct stages that combine local competence estimates to produce a global prediction of case-base competence. The following sections outline each of these stages and the interested reader is referred to [8,11,12,13,14] for further details.

2.1 Local Competence Estimates

The local competence contributions of individual cases are characterised by two sets. The *coverage set* of a *case* is the set of *target problems* that this case can successfully solve, while the *reachability set* of a *target problem* is the set of *cases* that can solve this target problem. It is impossible to enumerate all possible future target problems (T), but by using the case-base (C) itself as a representative sample of the target problem space we can efficiently estimate these sets as shown in Def. 1 and 2.

Def. 1 CoverageSet(c) = {c' ∈ C: Solves(c, c')}

Def. 2 ReachabilitySet(c) = {c' ∈ C: Solves(c', c)}

2.2 Shared Coverage & Competence Groups

Coverage and reachability sets are local estimates of competence only and to estimate the true competence contributions of cases it is necessary to model the interactions between cases in terms of how their coverage and reachability sets overlap.

Def. 3 RelatedSet(c) = CoverageSet(c) ∪ ReachabilitySet(c)

Def. 4 For c1, c2 ∈ C, SharedCoverage(c1, c2)
iff [RelatedSet(c1) ∩ RelatedSet(c2)] ≠ {}

Def. 5: For G = {c1, ..., cn} ⊆ C, CompetenceGroup(G)
iff ∀ ci ∈ G, ∃ cj ∈ G - {ci}: SharedCoverage(ci, cj) ∧
∀ cj ∈ C - G, ¬∃ cl ∈ G: SharedCoverage(cj, cl)

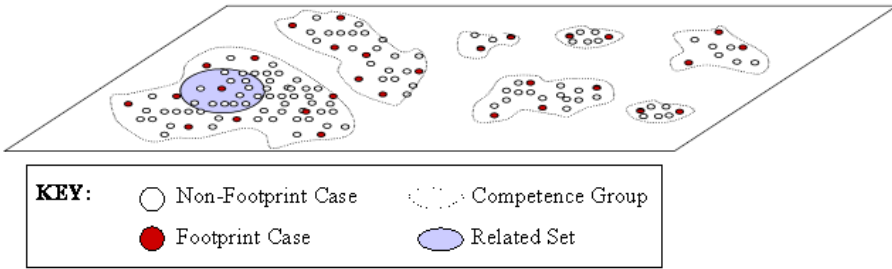


Fig. 1. A sample case-base showing competence groups, footprint cases, and related sets

First we define the related set of a case to be the union of its coverage and reachability sets (see Def. 3). When the related sets of two cases overlap we say that they exhibit *shared coverage* (see Def. 4) and cases can be grouped together into so-called *competence groups* which are maximal sets of cases exhibiting shared coverage (see Def. 5). In fact, every case-base can be organised into a unique set of competence groups which, by definition, do not interact from a competence viewpoint – that is, while each case within a given competence group must share coverage with at least one other case in that group, no case from one group can share coverage with any case from another group (see Figure 1). Moreover, this vital property of competence groups means that each group makes an independent contribution to overall

competence. In fact we argue that it is the competence groups, not the individual cases, that are the *fundamental units of competence* in case-based reasoners.

2.3 Footprint Cases

While every competence group makes a unique (and independent) contribution to competence, not every case in a competence group makes a positive competence contribution; for example, Smyth & Keane [11] have shown that so-called *auxiliary* cases make no competence contributions. The *footprint cases* of a competence group are those cases that do make a positive competence contribution and the *footprint set* is that minimal set of group footprint cases that collectively provides the same coverage as the entire group. The footprint set is important because it is only these cases that we need to consider when estimating the competence properties of a given group, the non-footprint cases are irrelevant from a competence viewpoint (although they may be relevant from an efficiency viewpoint) – see also Figure 1.

```

G, Competence Group
COV-FP (G)
R-Set  $\leftarrow$  cases in G; FP  $\leftarrow$  {}
While R-Set is not empty
    C  $\leftarrow$  case in R-Set with largest coverage set size
    FP  $\leftarrow$  FP  $\cup$  C
    R-Set  $\leftarrow$  R-Set - CoverageSet (C)
    Update coverage sets of cases in R-Set
EndWhile
Return (FP)

```

Algorithm 1. Computing the footprint set of a competence group

Algorithm 1 is one of many algorithms that we have explored to compute footprint sets (see also [12,13,14]). The algorithm adds cases with the largest coverage sets to the growing footprint set (FP). Each time a case is added, all of the cases that it covers are removed from the remaining case set (R-Set).

2.4 Relative Coverage & Global Competence

Each footprint case is chosen because it makes a positive contribution to group competence, but this does not mean that each footprint case makes the same competence contribution. In fact, one of the insights of the work of Smyth & Keane [11] is to demonstrate that cases in a CBR system will tend to vary greatly in their competence contributions. In order to compute the actual competence contribution of a footprint case we need to estimate its competence relative to other group cases, and this is the role of the *relative coverage* estimate (see Def. 6). It is based on the idea that if a case c' is covered by n other cases then each of the n cases will receive a contribution of $1/n$ from c' to their relative coverage measures.

Def 6:

$$\text{RelativeCoverage}(c) = \sum_{c' \in \text{CoverageSet}(c)} \frac{1}{|\text{ReachabilitySet}(c')|}$$

The competence contribution of a competence group then is simply the sum of the relative coverage values of the group's footprint cases. In turn, since competence groups make independent competence contributions, the competence of the case-base as a whole is simply the sum of the group competence estimates.

This completes the outline description of our competence model. With this model we can estimate and predict the competence of different case-bases. Moreover, it is now possible to assess the competence contributions of individual cases, a vital component in any case addition/deletion maintenance strategy.

3 An Efficient Procedure for Updating the Competence Model

Each time a case is learned by a case-based reasoner the competence model must be updated and this contributes an additional cost to the learning process. In this section we explain how to reduce this cost by describing an efficient model-update procedure.

3.1 The Standard Update Procedure

Before describing this new update procedure it is useful to begin with a description of what might be termed the standard update procedure (shown in Algorithm 2). This algorithm consists of three basic steps. First, the local competence characteristics, the coverage and reachability sets (and therefore the related sets), of the new case must be computed. This involves comparing every case in the case-base to the newly learned case in order to determine whether these cases can solve, or be solved by, this new case; this process is $O(n)$ in the size of the case-base.

Second, the competence group membership of the new case must be computed and there are a number of conditions that must be checked during this computation. If the case cannot be solved by, or cannot itself solve, any other case in the case-base – such a case is termed a pivotal case by Smyth & Keane [8,11] and will have a related set that contains only itself – then it will belong to a new singleton competence group. However, usually the new case will be solved by, or can solve, at least one other case and will therefore have a related set containing a number of other cases. If all of these cases belong to the same competence group then the new case will also belong to this group. However, if these cases belong to different groups then the new case is a so-called spanning case (see [8,11]) and results in the merging of these competence groups into one new competence group. This new group is made up of the union of the cases from the merged groups plus the new case.

The third step involves updating the footprint set of the competence group containing the new case. Strictly speaking this should involve recomputing the footprint set for the affected group but a more efficient procedure is available. If the new case is a pivotal case and resulted in the creation of a new singleton competence group then this new group will have a footprint set that contains just this pivotal case.

If the new case was found to be part of one existing competence group then it can only impact on the footprint set of this group if it is not currently covered by the cases in this footprint set; that is, if the intersection between the footprint set and the new case's reachability set is the empty set. If this is true then the new case is added to the group footprint set. Finally, if the new case is a spanning case, and has resulted in the creation of a new group by merging existing groups, then the footprint set of this new group is the union of the footprint sets of the merged groups (for simplicity this is carried out as part of the group membership update in Algorithm 2) and if the new case is not covered by any cases in this footprint set then it is added.

```

C - learned case, CB - case-base
Model-Update(C, CB)

  1. Compute Local Competences
  For each case  $x \in CB$ 
    If Solves(C, x) then
      Add x to CoverageSet(C)
      Add C to ReachabilitySet(x)
    EndIf
    If Solves(x, C) then
      Add x to ReachabilitySet(C)
      Add C to CoverageSet(x)
    EndIf
  EndFor

  2. Update Group Membership
  If RelatedSet(C) - {C} = {} create new group G containing C
  Else
    If all cases in RelatedSet(C) belong to same group, G
      Add C to G
    Else create a new group, G, by merging the groups
      that the cases in RelatedSet(C) belong to.
      FootprintSet(G) = union of footprint sets of
        Merged groups.
  EndIf

  3. Update Group Footprint
  If RelatedSet(C) - {C} = {} then FootprintSet(G) = {C}
  Else if FootprintSet(G)  $\cap$  ReachabilitySet(C) = {} then
    Add C to FootprintSet(G)
  EndIf

```

Algorithm 2. The Standard Competence Model Update Procedure

3.2 A More Efficient Update Procedure

The majority of the cost of the standard update procedure is associated with the calculation of the local competence characteristics of the new case; $O(n)$ in the size of the case-base. By comparison the group membership and footprint update costs are very low. Therefore, any efficiency gains in the local competence calculation will have a significant impact on the overall cost of updating the model. In this section we describe a more efficient competence model update procedure (see Algorithm 3) that

benefits from significant reductions in the cost of computing the related set for the newly learned case. The basic idea is to compute the related set for the new case by examining only a small fraction of the cases in the case-base. This makes sense because in reality only a small fraction of the cases in the case-base will find their way into the coverage and reachability sets of a new case – most cases in the case-base cannot solve, or be solved by, a newly learned case.

```

C - learned case, CB - case-base, k
Compute Local Competences(C, CB, k)
  Ordered-FS  $\leftarrow$  FootprintSet(CB) sorted in descending
    order of similarity to C
  Footprint-Cases  $\leftarrow$  first k cases in Ordered-FS
  Update-Set  $\leftarrow$  Union of related sets of Footprint-Cases
  For each case x  $\in$  Update-Set
    If Solves(C, x) then
      Add x to CoverageSet(C)
      Add C to ReachabilitySet(x)
    EndIf
    If Solves(x, C) then
      Add x to ReachabilitySet(C)
      Add C to CoverageSet(x)
    Endif
  EndFor

```

Algorithm 3. A more efficient procedure for computing the local competence characteristics of a newly learned case

Of course, if the related set is to be computed with respect to a small subset of the case-base then it is critical that this subset (which we will call the *update set*) is carefully selected so that it contains only those cases that are likely to be members of the new case's related set. If any related cases are missing from the update set then they will never be added to the new case's related set, resulting in model errors.

We propose a two-step method for computing the update set for a new case, which can be adjusted in favour of update efficiency or accuracy as required. The first step of this method is to compare the new case to each of the cases in the footprint set of the entire case-base; that is, the union of each of the competence group footprint sets. This footprint set is sorted in ascending order of similarity between its cases and the newly learned case to produce an *ordered footprint set* (Ordered-FS in Algorithm 3). Step two then computes the update set as the union of the related sets of the first k footprint cases in the ordered footprint set. The coverage and reachability sets for the new case can now be determined in the usual way but with reference to the update set only, rather than the entire case-base.

This procedure has the potential to significantly reduce the cost of computing the new case's related set since the update set will contain a small fraction of the cases in the case-base. The cost of computing the update set, which involves comparing the new case to each case in the case-base footprint set, also remains low since the size of the footprint set is generally a fraction of the entire case-base. The accuracy of the update set can be tailored by adjusting k, the number of footprint cases that need to be examined. Low values of k improve efficiency by producing small update sets, but

these sets may be missing relevant cases. Higher values of k increase the likelihood that the update set will contain all of the relevant cases, but efficiency is reduced.

4 Experimental Analysis

In the previous sections we have outlined a proven model of competence for case-based reasoners, and introduced a new procedure for updating this model to accommodate the run-time learning of new cases. We claim that this new update procedure will offer significant cost savings without loss of model accuracy, and in this section we back up these claims with empirical evidence.

4.1 Experimental Setup

The test data for our analysis comes from two freely available case-bases. The Travel case-base (available from the case-base archive on the AI-CBR web-site at <http://www.ai-cbr.org>) contains 1400 cases from the travel domain. Each case describes a package holiday in terms of features such as location, style, accommodation, number of people, price etc. The Property case-base (available from the UCI Machine Learning Repository, [1]) contains cases from the residential property domain, each describing an individual residential property in terms of features such as, location, style, facilities, crime rates, price, etc.

We built CBR systems for these case-bases. For each system the solvability criterion was based on a similarity threshold; a target problem was successfully solved if the similarity between it and the retrieved case exceeded the threshold.

4.2 Update Efficiency

In this experiment we compare the cost of competence model updates using the standard and new update procedures. Moreover, we focus on the local competence update costs since this is where the two procedures differ and, in the standard procedure, this accounts for the lion's share of the update costs.

As explained in Section 3, the cost of computing the local competence characteristics by using the standard update procedure is $O(n)$ in the size of the case-base; that is, using our experimental CBR systems, each time a new case is added to the current case-base, it must be compared to every case in this case-base to determine which cases can solve it and which cases it can solve. In contrast, the corresponding cost in the new update procedure is based on p comparisons to compute the update set (that is, each of the p footprint cases must be compared to the new case) plus q comparisons, since each of the q cases in the update set must be compared to the new case to assess solvability. In this experiment we measure these costs (n and $p+q$) during learning for the Travel and Property systems.

Method: The Travel and Property systems are initialized to contain 400 and 200 cases respectively and the competence model is built for each case-base from scratch. Next we emulate run-time case learning by adding the remaining cases to each case-

base and updating the competence models accordingly. In total, four competence models are maintained for each system, one based on the standard update procedure and three based on the new update procedure with values of k of 1, 3, and 5. We compute the update cost (in terms of the number of case comparisons during the local competence computation) for each case update and calculate the speed-up ratio $(n/p+q)$, for each value of k . To remove any ordering effects from the results we repeat this process 100 times each time using a different random ordering of the Travel and Property case-bases – this guarantees different initial case-bases and ensures that the order in which cases are learned varies with each run. This produces 100 different speed-up ratios for each update from which a mean value is computed.

Results: The results are shown in Figure 2(a&b) as graphs of mean speed-up versus case-base size for the Travel and Property domains respectively. Each graph consists of three plots and each plot corresponds to the speed-up profile produced for a given value of k (1, 3 or 5, as indicated).

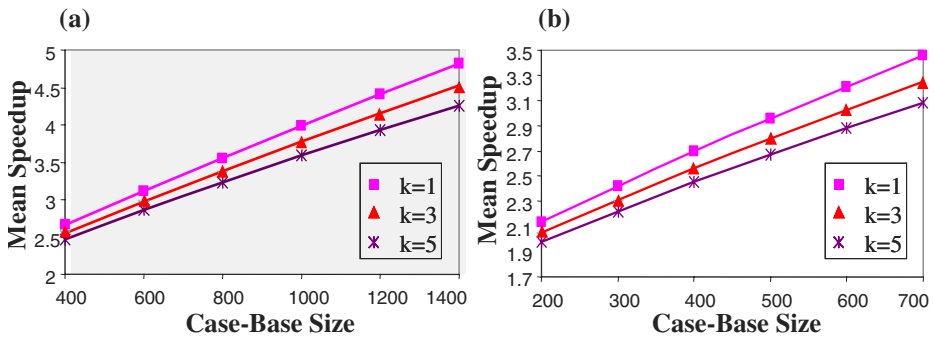


Fig. 2. The speed-up of the new model update procedure compared to the standard update procedure for the (a) Travel and (b) Property domains

Discussion: The results indicate that there are cost savings associated with the new update procedure. For example, in the Travel domain, Figure 2(a), we attain a speed-up value of nearly 5 for the model update associated with a 1400 case case-base (at the $k=1$ setting). This speed-up value indicates that the new procedure examines only 20% of the cases that the standard procedure must examine. We also see that the potential speed-up is increasing with case-base size. This means that the cost saving associated with the new update procedure is increasing, relative to the standard cost, as the case-base grows. Finally, as expected, as the value of k increases we witness a small decrease in speed-up; for example we note a 2.5% decrease in speed-up for each single increment of k . These results are found to be consistent in both domains and in summary we can conclude that the new update procedure offers significant and reliable cost savings over the standard procedure.

4.3 Model Accuracy

The efficiency benefits of the update procedure are meaningless if model accuracy is compromised; key cases could be missing from the coverage or reachability sets computed by the new procedure. We look at this issue here.

Def 7.
$$\text{Accuracy}(\text{Set1}, \text{Set2}) = 100 \bullet \frac{|\text{Set1} \cap \text{Set2}|}{|\text{Set1} \cup \text{Set2}|}$$

Method: We follow the same method used in the previous experiment. However, after each update, instead of noting the update cost, we use the formula in Def. 7 to measure the percentage accuracy of the coverage and reachability sets produced by the new update procedure with respect to the equivalent sets produced by the standard update procedure. For each update this gives a coverage set accuracy value and a reachability set accuracy value and these are combined to produce a mean accuracy value for that update. This results in 100 mean accuracy values for each update to a case-base of size n over the 100 different case orderings. The 100 values are combined to produce an overall mean accuracy for each case-base size.

Results: The results are shown in Figure 3(a&b) as graphs of overall mean accuracy versus case-base size for the Travel and Property domains respectively. Each graph consists of three plots and each plot corresponds to the accuracy profile of the new update procedure for a given value of k as indicated.

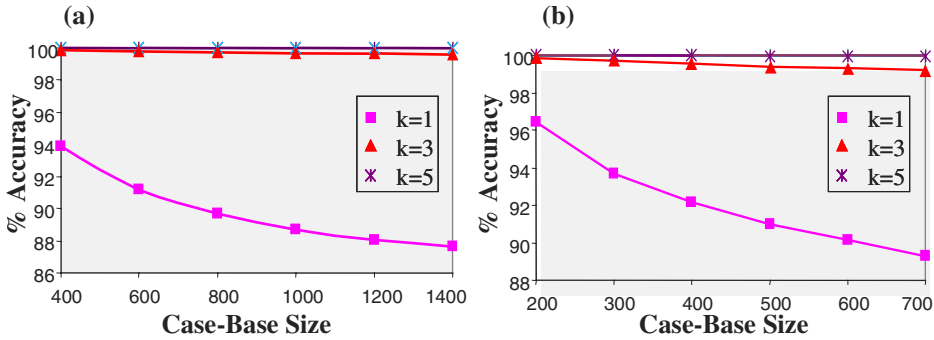


Fig. 3. The percentage accuracy of the competence models produced by the new and standard model update procedures for the (a) Travel and (b) Property domains

Discussion: The results are very positive. As expected, the accuracy of the model improves with increasing values of k . In both domains the new update procedure is seen to deliver 100% accuracy at $k=5$. This level of accuracy drops for lower values of k . For example, for a 1400 case Travel case-base, at $k=3$ the accuracy is 99.6%, which means that only 1 case in 250 is missing from the local competence sets produced by the new update procedure. In both domains, there are significant accuracy jumps between the values of $k=1$ and $k=3$. Accuracy is also seen to slowly degrade as the case-base grows in size; this is especially noticeable for low values of k (see $k=1$ plots in Figure 3(a&b)). This is expected as update errors have a cumulative effect, and any errors that creep in early on can result in the introduction of additional errors in the future. This cumulative effect is small, and reduces for increasing values of k . In the Travel domain, the accuracy drops from 99.9% (at the 400 case level) to 99.6% (at the 1400 case level) for the $k=3$ setting in the new update procedure, a fall of only 0.25% in accuracy; that is, less than 4 new errors in 1000 case updates. A similar pattern of results is found in the Property domain. Thus, the new update procedure provides significant efficiency benefits without compromising

model accuracy. In fact, the new procedure manages to attain perfect accuracy levels at reasonable values of k , for which significant efficiency gains are still available.

5 Discussion

The story so far shows that our competence model update procedure can significantly reduce the cost of maintaining the competence model without compromising its accuracy or effectiveness. There is one final twist in the tale that deserves discussion.

In previous work we have described how the competence model can be used to drive a novel case retrieval technique (called *footprint-based retrieval*) that can deliver superior retrieval results by examining only a fraction of the cases in the case-base [14]. The interesting feature of this retrieval technique is that the competence model is used to select a subset of cases (called the *retrieval set*) for examination during retrieval. Moreover, the method used for selecting this subset of cases, and the subsequent comparisons between the target problem and this subset, is very similar to the method used by our new model update procedure to compute the local competence characteristics of a newly learned case. In other words, the same basic computations are carried out in footprint-based retrieval and model update – the target case in retrieval corresponds to the learned case in model update, the selection of the retrieval set corresponds to the selection of the update set, and the retrieval comparisons correspond to the local competence comparisons.

Therefore, in a CBR system using footprint-based retrieval, the cost of updating the competence model can come almost for free, since the task of computing the coverage and reachability sets for the newly learned case is carried out as a normal part of the footprint-based retrieval that led to this new case. In other words, starting with a target problem, the footprint-based retrieval procedure selects the most similar case in the case-base, this case is adapted and (assuming learning is appropriate) the target problem description plus the newly adapted solution are packaged as a new case and added to the case-base. Finally, if the similarity between the target and a case can be used to estimate the solvability of the target with respect to the case, then the computations performed during retrieval to calculate the similarity between the target and its related cases in the case-base, can be reused to determine the coverage and reachability sets of the new case during the competence model update.

6 Conclusions

We have outlined an effective competence model for CBR systems and introduced a new procedure for updating this model at run-time. We have also provided empirical evidence to support claims that this new update procedure offers significant efficiency benefits over the standard update method without compromising model accuracy. We believe that explicit models of performance (such as competence and efficiency models) are of fundamental importance to the CBR community, as we believe that such models may hold the key to a range of important CBR problems. For instance, we have already shown how our competence model can be used to develop novel case

addition, retrieval, case-base visualisation, and authoring support solutions [12,13,14]. Our new update procedure means that these solutions are available at a reduced cost.

References

1. Blake, C., Keogh, E. & Merz, C.J: UCI Repository of machine learning databases [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science (1998)
2. Francis, A.G. and Ram, A.: A Comparative Utility Analysis of Case-Based Reasoning and Control Rule Problem Solving. In: Proceedings of the 8th European Conference on Machine Learning (1995)
3. Kolodner, J. Case-Based Reasoning. Morgan-Kaufmann, San Mateo, California (1993)
4. Leake, DB. Case-Based Reasoning: Experiences, Lessons, and Future Directions. MIT Press (1996)
5. Leake, DB, Wilson DC: Categorizing Case-Base Maintenance: Dimensions and Directions. In: B. Smyth & P. Cunningham (Eds.) Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, LNAI 1488. Springer-Verlag (1998). 196-207
6. Minton, S.: Qualitative results concerning the utility of explanation based learning. Artificial Intelligence, **42**(2-3) (1991) 393-391
7. Racine, K. & Yang, Q: Maintaining Unstructured Case Bases. In: DB Leake & E. Plaza (eds.) Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence, LNAI 1266. Springer-Verlag (1997) 553-564
8. Smyth, B.: Case-Base Maintenance. In: AP del Pobil, J. Mira, M. Ali (eds.) Tasks and Methods in Applied Artificial Intelligence, LNAI 1416. Springer-Verlag (1998). 507-516
9. Smyth, B. & Cunningham, P.: The Utility Problem Analysed: A Case-Based Reasoning Perspective. In: I. Smith, & B. Faltings (eds.): Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, Vol. 1168. Springer-Verlag (1996) 392-399
10. Smyth, B. & Keane, M.T.: Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. Artificial Intelligence, **102** . (1998) 249-293
11. Smyth, B. & Keane, M.T.: Remembering to Forget: A Competence Preserving Deletion Policy for Case-Based Reasoning Systems. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. Morgan-Kaufmann. (1995) 377-382
12. Smyth, B. & McKenna, E. Modelling the Competence of Case-Bases. In: B. Smyth & P. Cunningham (eds.) Advances in Case-Based Reasoning. Lecture Notes in Artificial Intelligence, LNAI 1488. Springer-Verlag, (1998). 208-220
13. Smyth, B. & McKenna, E. Building Compact Competent Case-Bases. In: K-D Althoff, R. Bergmann, L-K Branting (eds.) Case-Based Reasoning Research & Development. Lecture Notes in Artificial Intelligence, LNAI 1650. Springer-Verlag, (1999) 329-342
14. Smyth, B. & McKenna, E. Footprint-Based Retrieval. . In: K-D Althoff, R. Bergmann, L-K Branting (eds.) Case-Based Reasoning Research & Development. Lecture Notes in Artificial Intelligence, LNAI 1650. Springer-Verlag (1999) 343-357
15. Zhu, J. & Yang, Q: Remembering to Add: Competence-preserving Case Addition Policies for Case Base Maintenance. In: Proceedings of International Joint Conference in Artificial Intelligence (IJCAI-99). Morgan-Kaufmann. 234-241

Layered Learning

Peter Stone¹ and Manuela Veloso²

¹ AT&T Labs — Research

180 Park Ave., Florham Park, NJ 07932, USA

² Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract. This paper presents *layered learning*, a hierarchical machine learning paradigm. Layered learning applies to tasks for which learning a direct mapping from inputs to outputs is intractable with existing learning algorithms. Given a hierarchical task decomposition into *subtasks*, layered learning seamlessly integrates separate learning at each subtask layer. The learning of each subtask directly facilitates the learning of the next higher subtask layer by determining at least one of three of its components: (i) the set of training examples; (ii) the input representation; and/or (iii) the output representation. We introduce layered learning in its domain-independent general form. We then present a full implementation in a complex domain, namely simulated robotic soccer.

1 Introduction

Machine learning (ML) algorithms select a hypothesis from a hypothesis space based on a set of training examples such that the chosen hypothesis is predicted to characterize unseen examples as accurately as possible. Each hypothesis maps a set of input features to a set of output features. Inputs are constructed from information in the domain and outputs are possible classifications or actions.

Our research focuses on learning tasks for which learning a direct mapping from inputs to outputs is intractable given existing learning algorithms. The approach we take is to break the problem down into several hierarchical learning layers such that each layer facilitates the learning of the next. By determining the set of training examples, the input representation, or the output representation, previously learned functions can enable the creation of increasingly complex learned functions. We call this approach to machine learning “layered learning.”

Layered learning assumes that the appropriate aspects of the task to be learned are determined as a function of the specific domain. It does not include an automated hierarchical decomposition of the task. Each layer is learned by applying an ML algorithm that is appropriate for the specific subtask characteristics. In this paper, we apply layered learning to a complex multiagent learning task, namely simulated robotic soccer.

We have previously presented the individual learned tasks in this domain [18,20] as well as a preliminary version of the concept of layered learning [18]. This paper contributes the concrete domain-independent specification of layered

learning as presented in Sections 2 and 3. Section 4 reviews our machine learning research in the simulated robotic soccer domain, couching it in the terms of our layered learning specification. This layered learning example is fully implemented and tested as described in Section 5. In Sections 6 and 7, we relate layered learning to previous research and discuss directions for future work.

2 Layered Learning

Table 1 summarizes the principles of our layered learning paradigm which are described in detail in this section.

Table 1. The key principles of layered learning

-
1. A mapping directly from inputs to outputs is not tractably learnable.
 2. A bottom-up, hierarchical task decomposition is given.
 3. Machine learning exploits data to train and/or adapt. Learning occurs separately at each level.
 4. The output of learning in one layer feeds into the next layer.
-

2.1 Principle 1

Layered learning is designed for domains that are too complex for learning a mapping directly from the input to the output representation. Instead, the layered learning approach consists of breaking a problem down into several task layers. At each layer, a concept needs to be acquired. A machine learning algorithm abstracts and solves the local concept-learning task.

2.2 Principle 2

Layered learning uses a bottom-up incremental approach to hierarchical task decomposition. Starting with low-level subtasks, the process of creating new ML subtasks continues until reaching the high-level task that deal with the full domain complexity. The appropriate learning granularity and subtasks to be learned are determined as a function of the specific domain. The task decomposition in layered learning is not automated. Instead, the layers are defined by the ML opportunities in the domain.

2.3 Principle 3

Machine learning is used as a central part of layered learning to exploit data in order to *train* and/or *adapt* the overall system. ML is useful for training functions that are difficult to fine-tune manually. It is useful for adaptation when the task details are not completely known in advance or when they may change

dynamically. In the former case, learning can be done off-line and frozen for future use. In the latter, on-line learning is necessary: since the learner needs to adapt to unexpected situations, it must be able to alter its behavior even while executing its task. Like the task decomposition itself, the choice of machine learning method depends on the subtask.

2.4 Principle 4

The key defining characteristic of layered learning is that each learned layer directly affects the learning at the next layer. A learned subtask can affect the subsequent layer by:

- constructing the set of training examples;
- providing the features used for learning; and/or
- pruning the output set.

All three cases are illustrated in our implementation described in Section 4.

3 Formalism

Consider the learning task of identifying a hypothesis h from among a class of hypotheses H which map a set of state feature variables S to a set of outputs O such that, based on a set of training examples, h is most likely (of the hypotheses in H) to represent unseen examples.

When using the layered learning paradigm, the complete learning task is decomposed into hierarchical subtask layers $\{L_1, L_2, \dots, L_n\}$ with each layer defined as

$$L_i = (\mathbf{F}_i, O_i, T_i, M_i, h_i)$$

where:

$\mathbf{F}_i$ is the input vector of state features relevant for learning subtask L_i . $\mathbf{F}_i = \langle F_i^1, F_i^2, \dots \rangle$. $\forall j, F_1^j \in S$.

O_i is the set of outputs from among which to choose for subtask L_i . $O_n = O$.

T_i is the set of training examples used for learning subtask L_i . Each element of T_i consists of a correspondence between an input feature vector $\mathbf{f} \in \mathbf{F}_i$ and $o \in O_i$.

M_i is the ML algorithm used at layer L_i to select a hypothesis mapping $\mathbf{F}_i \mapsto O_i$ based on T_i .

h_i is the result of running M_i on T_i . h_i is a function from $\mathbf{F}_i$ to O_i .

As set out in Principle 2 of layered learning, the definitions of the layers L_i are given a priori. Principle 4 is addressed via the following stipulation. $\forall i < n$, h_i directly affects L_{i+1} in at least one of three ways:

- h_i is used to construct one or more features F_{i+1}^k .
- h_i is used to construct elements of T_{i+1} ; and/or
- h_i is used to prune the output set O_{i+1} .

It is noted above in the definition of $\mathbf{F}_i$ that $\forall j, F_1^j \in S$. Since F_{i+1} can consist of new features constructed using h_i , the more general version of the above special case is that $\forall i, j, F_i^j \in S \cup_{k=1}^{i-1} O_k$.

4 Implementation

In this section, we illustrate layered learning via a full-fledged implementation in the RoboCup Soccer Server [14]. Here, the high-level goal is for a team of independently controlled agents to achieve complex collaborative and adversarial behavior. The subtasks are increasingly complex individual and multiagent behaviors.

The purpose of this section is to illustrate layered learning via a fully-implemented system. full details of the domain and each individual learned subtask have been previously reported. However they have not been represented in terms of the formalism presented in Section 3.

4.1 Simulated Robotic Soccer

The RoboCup soccer server [14] has been used as the basis for successful international competitions and research challenges [8]. As presented in detail in [17], it is a fully *distributed*, *multiagent* domain with both *teammates* and *adversaries*. There is *hidden state*, meaning that each agent has only a partial world view at any given moment. The agents also have *noisy sensors and actuators*, meaning that they do not perceive the world exactly as it is, nor can they affect the world exactly as intended. In addition, the perception and action cycles are *asynchronous*, prohibiting the traditional AI paradigm of using perceptual input to trigger actions. *Communication* opportunities are limited; the agents must make their decisions in *real-time*; and the actions taken by other agents, both teammates and adversaries, and their resulting state transitions are unknown. We refer to this last quality of unknown state transitions as *opaque transitions*. These italicized domain characteristics combine to make simulated robotic soccer a realistic and challenging domain.

4.2 Layered Learning in Robotic Soccer

Consider the task of a robotic soccer agent retrieving a moving ball and deciding what to do with it. It could dribble the ball, pass to a teammate, or shoot towards the goal. While this task does not encompass the entire robotic soccer task (agents must also decide what to do when they don't have the ball), it comprises an important part of the complete task.

We decompose this task into three learning components or subtasks: ball interception, pass evaluation, and pass selection. Given this hierarchical decomposition, layered learning allows us to create effective team-oriented agent behaviors.

Table 2 illustrates our set of learned behavior levels within the simulated robotic soccer domain. We identify a useful low-level skill that must be learned before moving on to higher-level strategies. Then we build upon it to create higher-level multiagent and team behaviors. Full details regarding the training and testing of each learned behavior are reported in [17].

Table 2. Examples of different behavior levels in robotic soccer

Layer	Behavior type	Example
L_1	individual	ball interception
L_2	multiagent	pass evaluation
L_3	team	pass selection

L_1 : Ball Interception — an individual skill. First, the agents learn a low-level individual skill that allows them to control the ball effectively. While executed individually, the ability to intercept a moving ball is required due to the presence of other agents: it is needed to block or intercept opponent shots or passes as well as to receive passes from teammates. As such, it is a prerequisite for most ball-manipulation behaviors. We chose to have our agents learn this behavior because it was easier to collect training data than to fine-tune the behavior by hand.¹

L_1 is defined as follows.

$F_1 = \{BallDist_t, BallAng_t, BallDist_{t-1}\}$: The agent learns what action to take based on the ball's current distance and angle from the defender, and the ball's distance a fixed time (250 msec.) in the past.

$O_1 = \{TurnAng\}$: The agent chooses an angle to turn such that it will be likely to intercept the ball.

T_1 : The training procedure for ball interception involves a stationary forward repeatedly shooting the ball towards a defender in front of a goal. The defender collects training examples by acting randomly and noticing when it successfully stops the ball. Test examples are classified as saves (successful interceptions), goals (unsuccessful attempts), and misses (shots that went wide of the goal).

$M_1 =$ a neural network: Ball interception is trained with a fully-connected neural network with 4 sigmoid hidden units and a learning rate of 10^{-6} . The weights connecting the input and hidden layers use a linearly decreasing weight decay starting at .1%. We use a linear output unit with no weight decay. The neural network was trained for 3000 epochs.

$h_1 =$ a trained interception behavior: Table 3 shows the effect of the number of training examples on learned save percentage. With about 750 training examples, the defender is able to stop 91% of shots on goal (saves + goals; misses are omitted), a comparable save rate to that achieved when using an analytic ball interception behavior [18].

L_2 : Pass Evaluation — a multiagent behavior. Second, the agents use their learned ball-interception skill as part of the behavior for training a multiagent

¹ The learning was done in an early implementation of the soccer server (Version 2) in which agents did not receive any velocity information when seeing the ball.

Table 3. The defender’s performance when using neural networks trained with different numbers of training examples

Training Examples	Saves		
	Saves(%)	Goals(%)	$\frac{\text{Saves}}{\text{Goals} + \text{Saves}}(\%)$
100	57	33	63
200	73	18	80
300	81	13	86
400	81	13	86
500	84	10	89
750	86	9	91
1000	83	10	89
4773	84	9	90

behavior. When an agent has the ball and has the option to pass to a particular teammate, it is useful to have an idea of whether or not the pass will actually succeed if executed: will the teammate successfully receive the ball? Such an evaluation depends on not only the teammate’s and opponents’ positions, but also their abilities to receive or intercept the pass. Consequently, when creating training examples for the pass-evaluation function, we equip the intended pass recipient as well as all opponents with the previously learned ball-interception behavior, h_1 . Again, we chose to have our agents learn the pass-evaluation capability because it is easier to collect training data than to construct it by hand.

L_2 is defined as follows.

F_2 = a set of 174 continuous and ordinal features: There are many features that could possibly affect pass evaluation. We encode a large set of attributes representing the relative positions of teammates and opponents on the field as well as statistical counts reflecting their relative positioning [18]. These features are not carefully chosen. On the contrary, many possibly irrelevant features are included, leaving the ML algorithm to select the proper ones. A full list of the attributes can be found in [18].

$O_2 = [-1, 1]$: A potential pass to a particular receiver is classified as a success with a confidence factor $\in (0, 1]$, a failure with a confidence factor $\in [-1, 0)$, or a miss ($= 0$).

T_2 : The training procedure for pass evaluation involves a passer kicking the ball towards randomly-placed teammates interspersed with randomly-placed opponents. The training scenario is illustrated within a screen shot of the soccer server in Figure 1. The dashed line indicates the region in which the teammates and opponents are randomly placed. *The intended pass recipient and the opponents all use the learned ball-interception behavior, h_1 .* Trials are classified as successes (a teammate intercepts the ball), failures (an opponent intercepts the ball), and misses (no player intercepts the ball). When passing to a random teammate, 51% of passes are successful.

$M_2 = C4.5$: To learn pass evaluation, we use the C4.5 decision tree training algorithm [15] with all of the default parameters. Decision trees are cho-

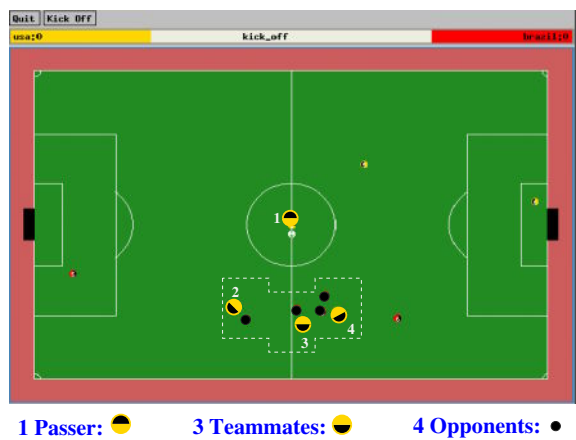


Fig. 1. The training scenario for pass evaluation. The dashed line indicates the region in which the teammates and opponents are randomly placed prior to each trial

sen over neural networks because of their ability to ignore irrelevant input features.

h_2 = a trained pass-evaluating decision tree: During testing, the trained decision tree returns a predicted classification as well as a confidence factor, resulting in a value between -1 and 1 . Table 4 tabulates our results indicating that the trained decision tree enables the passer to choose successfully from among its potential receivers. Overall results are given as well as a breakdown by the passer’s confidence prior to the pass. In this experiment, the passer is forced to pass even if it predicts failures for all 3 teammates. In that case, it passes to the teammate with the lowest likelihood of failure. 65% of all passes and 79% of passes predicted to succeed with high confidence are successful.

Table 4. The results of 5000 trials during which the passer uses the DT to choose the receiver. Results are given in percentages of the number of cases falling within each confidence interval (shown in parentheses)

Result (Number)	Overall (5000)	Success Confidence:		
		.8-.9 (1050)	.7-.8 (3485)	.6-.7 (185)
SUCCESS (%)	65	79	63	58
FAILURE (%)	26	15	29	31
MISS (%)	8	5	8	10

L₃: Pass Selection — a collaborative and adversarial team behavior.

Third, the agents use their learned pass-evaluation capability h_2 to create the input space and output set for learning pass selection.² When an agent has the ball, it must decide to which teammate it should pass the ball.³ Such a decision depends on a huge amount of information including the agent’s current location on the field, the current locations of all the teammates and opponents, the teammates’ abilities to receive a pass, the opponents’ abilities to intercept passes, teammates’ subsequent decision-making capabilities, and the opponents’ strategies. The merit of a particular decision can only be measured by the long-term performance of the team as a whole. Therefore, we drastically reduce the input space with the help of the previously learned decision tree, h_2 : rather than considering the positions of all of the players on the field, only the pass evaluations for the possible passes to each teammate are considered.

L_3 is defined as follows.

$F_3 = \{PlayerPosition, O_2, O_2, O_2, \dots\}$: The input representation consists of one coarse geographical component and one action-dependent feature [20] for each possible pass. *The action-dependent features are precisely the result of h_2 executed for each possible receiver.*

$O_3 = \{shoot\} \cup \{Teammates\}$: The result of a pass selection decision is either a shot on goal or a pass to a particular teammate.

T_3 : Training examples are gathered on-line by individual team members during real games. Each individual agent learns in a separate partition of F_3 according to its position on the field. Agents learn based on the observed long-term effects of their actions [17]. *For each particular action decision, the eligible members of O_3 are pruned based on h_2 : only passes predicted to succeed are considered.*

$M_3 = \text{TPOT-RL}$: For training pass selection, we use TPOT-RL [20], an on-line, multiagent, reinforcement learning method motivated by Q-learning that is applicable in team-partitioned, opaque-transition domains such as simulated robotic soccer. We use the default parameters as reported in [20].

$h_3 = \text{a distributed pass-selection policy}$: We test the pass-selection learning by directly comparing two teams with identical behaviors other than their pass-selection policies. Agents on both teams begin by passing randomly, but agents on one team adjust their behavior based on experience using TPOT-RL. The other agents continue passing randomly. Figure 2 demonstrates the effectiveness of the learned passing policies. Additional tests against goal-directed opponents are reported in [20].

5 Discussion

In this section, we analyze the key benefits and limitations of layered learning. We also present empirical results of our overall layered learning implementation.

² The pass most likely to succeed is not in general the best pass strategically.

³ It could also choose to shoot. For the purposes of this behavior, the agents are not given the option to dribble.

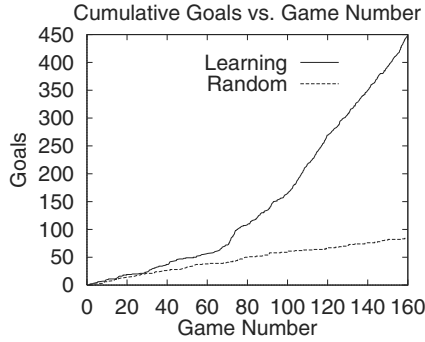


Fig. 2. Total goals scored by a learning team playing against a randomly passing team. The independent variable is the number of 10-minute games that have elapsed

5.1 Analysis

The three learned layers described in Section 4 illustrate the four principles of the layered learning paradigm from Section 2:

1. The decomposition of the task into smaller subtasks enables the learning of a more complex behavior than is possible when learning straight from the agents' sensory inputs.

Indeed, there have been two attempts at monolithic learning of agent behaviors in the soccer server. First, Luke et al. [11] set out to create a completely learned team of agents using genetic programming [9]. However, the ambition was eventually scaled back and low-level player skills were created by hand as the basis for learning. The resulting learned team won two of its four games at the international RoboCup-97 robotic soccer simulator competition, losing in the second round. The following year, at RoboCup-98, another genetic programming attempt at learning the entire team behavior was made [1]. This time, the agents were indeed allowed to learn directly from their sensory input representation. While making some impressive progress given the challenging nature of the approach, this entry was unable to advance past the first round in the tournament.

2. The hierarchical task decomposition is constructed in a bottom-up, domain-dependent fashion. The fact that the task decomposition needs to be provided to layered learning a priori our paradigm's main limitations, and it is this characteristic that leads us to describe layered learning as a "paradigm" or a "method" as opposed to an "algorithm." Automatically selecting abstractions for learning is still a challenging open problem.

However, layered learning could be combined with any algorithm for *generating* abstraction levels to create an abstraction selection routine. In particular, let A be an algorithm for generating task decompositions within a domain. Suppose that A does not have an objective metric for comparing different

decompositions. Applying layered learning on the task decomposition and quantifying the resulting performance can be used as a measure of the utility of A 's output.

3. Learning methods are chosen or created to suit the subtask. They exploit available data to train difficult behaviors (ball interception and pass evaluation) or to adapt to changing/unforeseen circumstances (pass selection). Again, this need to select the ML algorithm by hand is a limitation of layered learning. Automatically mapping from tasks to ML algorithms is another challenging open problem in the field. However, the flexibility to use any algorithm to match the needs of the subtask is an important characteristic of layered learning. For example, we exploited the ability of neural networks to learn continuous output values in L_1 of our robotic soccer implementation, used C4.5 to ignore irrelevant input features in L_2 , and created a multiagent learning algorithm capable of learning from limited training data in L_3 .
4. Learning in one layer feeds into the next layer either by providing a portion of the behavior used for training (ball interception – pass evaluation) or by creating the input representation and pruning the action space (pass evaluation – pass selection).

This last characteristic is a key principle of layered learning. It specifies how each successive subtask can leverage off of the learning of previous subtasks.

5.2 Results

The layered learning approach has contributed to our success at the first three international RoboCup robotic soccer competitions.⁴ Although competitions are not controlled testing scenarios and they do not provide means for isolating the positive and negative aspects of an approach, they do allow for evaluation of an overall implementation. We present our results at these competitions as supporting evidence, rather than proof, that layered learning is effective. Note that all of the individual learned layers described in Section 4 were validated in controlled experiments.

At the first robotic soccer world cup competition, RoboCup-97 [7], our team made it to the semi-finals in a field of 29 teams. At RoboCup-98 [2], our team won in a field of 34 teams. And at RoboCup-99 [22], our team repeated as champion in a field of 37 teams. Full details of the competitions are available at www.robocup.org.

6 Related Work

The original hierarchical learning constructs were devised to improve the generalization of a single learning task by running multiple learning processes. Both boosting [16] and stacked generalization [23] improve function generalization by

⁴ Robust low-level skills and a sophisticated team member agent architecture [19] also contributed significantly. We thank Patrick Riley for his implementation of the low-level skills [21].

combining the results of several generalizers or several runs of the same generalizer. These approaches contrast with layered learning in that the layers in layered learning each deal with *different* tasks. Boosting or stacked generalization could potentially be used within any given layer, but not across different layers.

More in line with the type of hierarchical learning discussed in this paper are hierarchical reinforcement learning algorithms. Because of the well-known “curse of dimensionality” in reinforcement learning RL researchers have been very interested in hierarchical learning approaches. As surveyed in [6], most hierarchical RL approaches use *gated* behaviors:

There is a collection of behaviors that map environment states into low-level actions and a gating function that decides, based on the state of the environment, which behavior’s actions should be switched through and actually executed. [6]

In some cases the behaviors are learned [13], in some cases the gating function is learned [12], and in some cases both are learned [10]. In this last example, the behaviors are learned and fixed prior to learning the gating function. On the other hand, feudal Q-learning [3] and the MAXQ algorithm [4] learn at all levels of the hierarchy simultaneously. A constant among these approaches is that the behaviors and the gating function are all control tasks with similar inputs and actions (sometimes abstracted). In the RL layer of our layered learning implementation, the input representation itself is learned. In addition, none of the above methods has been implemented in a large-scale, complex domain.

In all of the above RL approaches, like in layered learning, the task decomposition is constructed manually. However, there has been at least one attempt at the challenging task of learning the task decomposition. Nested Q-learning [5] generates its own hierarchical control structure and then learns low-level skills at the same time as it learns to select among them. Thus far, like other hierarchical RL approaches, it has only been tested on very small problems (on the order of 100 states in this case).

7 Conclusion and Future Work

This paper has presented the layered learning paradigm and illustrated it with a fully-implemented example in the robotic soccer domain. Our layered learning implementation, along with robust low-level skills and a sophisticated team member agent architectures which incorporates a flexible teamwork structure [19], has contributed to the success of our complete team of simulated robotic soccer competitions.

An important direction for future work is to apply layered learning in a new domain. As an example apparently orthogonal to robotic soccer, consider natural language understanding as another application of layered learning. Natural language understanding can have a clear hierarchical task decomposition. For example, learned word sense disambiguation could facilitate learned sentence parsing, which in turn could facilitate semantic encoding of sentences or

Table 5. Natural language understanding: a proposed layered learning application

Layer	Learning Task
L_1	Word sense disambiguation
L_2	Sentence syntax
L_3	Sentence semantics

paragraphs (see Table 5). While it is currently not possible in general to learn sentence semantics straight from a string of words, a hierarchical decomposition of the task coupled with the layered learning paradigm may render the learning task tractable. Indeed, layered learning is potentially applicable to any complex learning problem for which a hierarchical decomposition exists.

Layered learning is potentially applicable to this and other tasks that are too complex for monolithic learning. Its power is derived from the concept of directly combining different ML algorithms within a hierarchically decomposed task representation.

References

1. David Andre and Astro Teller. Evolving team Darwin United. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999. 377
2. Minoru Asada and Hiroaki Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Artificial Intelligence 1604. Springer Verlag, Berlin, 1999. 378
3. Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, San Mateo, CA, 1993. 379
4. Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998. 379
5. Bruce L. Digney. Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In *Proceedings of the 4th International Conference of Simulation of Adaptive Behavior*, pages 363–372. MIT Press, 1996. 379
6. Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996. 379
7. Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*. Springer Verlag, Berlin, 1998. 378
8. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29, San Francisco, CA, 1997. Morgan Kaufmann. 372
9. John R. Koza. *Genetic Programming*. MIT Press, 1992. 377

10. Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1993. 379
11. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 398–411, Berlin, 1998. Springer Verlag. 377
12. Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 796–802. Morgan Kaufmann, 1990. 379
13. Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 328–332, 1991. 379
14. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998. 372
15. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993. 374
16. Robert E. Shapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. 378
17. Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1998. Available as technical report CMU-CS-98-187. 372, 376
18. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998. 369, 373, 374
19. Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999. 378, 379
20. Peter Stone and Manuela Veloso. Team partitioned, opaque transition reinforcement learning. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 206–212. ACM Press, May 1999. 369, 376
21. Peter Stone, Manuela Veloso, and Patrick Riley. The CMUnited-98 champion simulator team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999. 378
22. Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors. *RoboCup-99: Robot Soccer World Cup III*. Springer Verlag, Berlin, 2000. To appear. 378
23. David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992. 378

Problem Decomposition for Behavioural Cloning

Dorian Šuc and Ivan Bratko

Faculty of Computer and Information Sciences, University of Ljubljana, Slovenia

Tržaška 25, 1000 Ljubljana

{dorian.suc, ivan.bratko}@fri.uni-lj.si

Abstract. In behavioural cloning of the human operator's skill, a controller is usually induced directly as a classifier from system's states into actions. Experience shows that this often results in brittle controllers. In this paper we explore a decomposition of the cloning problem into two learning problems: the learning of operator's control trajectories and the learning of the system's dynamics separately. We analyse advantages of such *indirect controllers*. We give characterization of the learner's error that is plausible explanation of why this decomposition approach has empirically proved to be usually superior to *direct* cloning.

1 Introduction

Controllers for dynamic systems can be designed by machine learning using different kinds of information available to the learning system. The idea of *behavioural cloning* (a term introduced by Donald Michie (1993)) is to make use of the operator's skill in the development of an automatic controller. Early work in behavioural cloning was done by Donaldson [8]) and Chambers and Michie [7]. A skilled operator's control traces are used as examples for machine learning to reconstruct the underlying control strategy that the operator executes subconsciously. The goal of behavioural cloning is not only to induce a successful controller, but also to achieve better understanding of the human operator's subconscious skill [13]. Behavioural cloning was successfully used in problem domains as pole balancing, production line scheduling, piloting [11] and operating cranes. These experiments are reviewed in [6].

The usual approach is to induce a control rule as a function $Action = Action(State)$ where $State$ is the state vector of the dynamic system, and $Action$ is the control action to be performed in $State$. The induced function is typically represented by a decision or regression tree. Although successful clones have been induced in the form of trees or rule sets, the following problems have generally been observed with this approach:

- Typically, induced clones are brittle with respect to small changes in the control task.
- The clone induction process typically has low yield: the proportion of successful controllers among all the induced clones is low, typically well below 50%.

An approach aiming at rectifying these defeciences, proposed in [14], exploits some results from control theory. This approach considers the system’s dynamics and automatic identification of discrete subgoals. It improves both the clones’ robustness with respect to changes in the control task, and the yield of the cloning process. However, this approach still has difficulties in domains with significant nonlinearities or where the operator’s control plan is not simply expressed in terms of a few discrete subgoals.

In [17] we proposed another approach to behavioural cloning, suitable also for strongly nonlinear domains. The trajectory the operator is trying to follow is generalized separately from the system’s dynamics and can be viewed as a *continuous subgoal*. In particular, we do not learn the trajectory in time, but rather the constraints among the state variables in the execution trace. These constraints determine the corresponding desired trajectory to the goal, also for system states other than those explicitly mentioned in the operator’s execution trace. Actions that maintain the desired trajectory are computed using knowledge of the system’s dynamics, learned by nonlinear function approximators. So the initial learning problem is decomposed into two learning problems: first, the generalisation of the operator’s control *trajectory* to areas outside the example trace, and second, the learning of the system’s dynamics.

Experiments performed in the crane domain in [17] and in the Acrobot domain [15] demonstrated that this decomposition approach enormously improves the yield of the cloning process and provides a good insight in the operator’s subcognitive skill. Qualitative strategy, generalized from the operator’s trajectory, is comprehensible and offers a possibility to optimize the operator’s control strategy.

In this paper we investigate why the proposed decomposition of the controller induction task into two learning tasks (learn generalized trajectory T and system’s dynamics D) works better than the original problem definition (learn $Action = Action(State)$ directly). In other words, why *indirect* controllers are more robust than *direct* controllers.

2 Problem Decomposition and Indirect Controllers

Let us now present the details of the problem decomposition for behavioural cloning investigated in this paper. The construction of a controller by induction from the operator’s traces consists of three stages (see Fig. 1):

1. Learn constraints T on operator’s trajectories, which can be formally stated as a mapping T :

$$T : States \times States \mapsto \{true, false\} \quad (1)$$

The trajectory constraints are usually represented as a function t , expressing a chosen state variable (*dependent* variable) as a function of the system’s state. A suitable chosen dependent variable is one that is most directly affected by the available actions. For example, for the pole-cart sys-

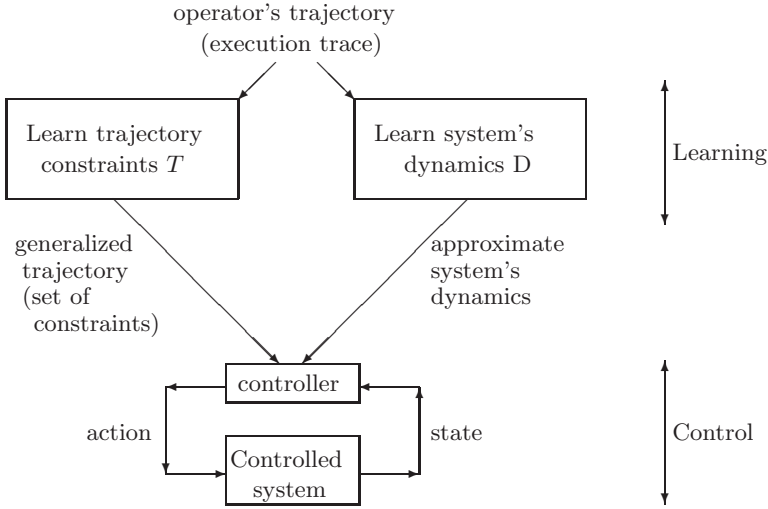


Fig. 1. Indirect controller can be induced from operator's control trace by generalizing the operator's trajectory and learning the system dynamic's model. Both together are used to control the system

tem: $\dot{x} = t(x, \theta, \dot{\theta})$. Constraints T define what will be called a *generalized operator's trajectory*.

2. Learn appropriate model D of the system's dynamics. D can for example be a function that maps system's states and actions into acceleration, eg.: $\ddot{x} = f(x, \dot{x}, Action)$.
3. Once a trajectory model T and dynamics model D have been induced, they are used to construct a controller which works as follows:
 - (a) Given a current system state x_k at time point k , $\Delta = \text{diff}(x_k, T)$ denotes the deviation between the state and the generalized operator's trajectory. Usually the deviation is defined simply as the difference between the value of chosen distinguished variable defined by T and its current value.
 - (b) Use system's dynamics model D to determine an action A which reduces the deviation Δ in time. For discrete time k , this can be written as:

$$A_k = \arg \min_{A \in \mathcal{A}} \text{diff}(x_{k+1}, T) \quad (2)$$

where $x_{k+1} = D(x_k, A_k)$ and $\mathcal{A}$ is the set of possible actions. We call this an *indirect controller* as opposed to a *direct controller* that computes action directly as $A_k = A(x_k)$ (see Fig. 2).

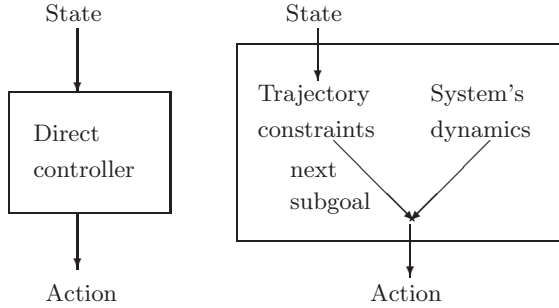


Fig. 2. Direct and indirect controller: on the left is an direct controller, and on the right is indirect controller. Direct controller uses a direct mapping from states to actions, whereas the indirect controller uses the trajectory constraints and the model of the system dynamics to take the action towards the next subgoal

3 Direct and Indirect Controllers

In the following analysis we assume deterministic system's dynamics f and discrete time, i.e. $x_{k+1} = f(x_k, u_k)$, where index k indicates successive sample times $t_0, t_0 + dt, \dots, t_0 + k dt$, x denotes the state vector and u the control action vector.

Controller observes current state x and takes corresponding action u , i.e, it is a function mapping states to control actions: $x \mapsto u$. This controller function can take different forms. In the case of a linear controller it is a product of the error in state $(x - x_g)$ and a matrix K , while in the case of classical approach to behavioral cloning, it is a regression or decision tree. Classic control theory approach to controller design usually involves modeling of the system's dynamics, whereas the usual approach to behavioral cloning does not use the system's dynamics model and learns a direct mapping from states to actions.

To make distinction between the controllers that do and do not use a system's dynamics model and to compare the learnability of such controllers we define two kinds of controllers. *Direct controller* corresponds to a direct mapping from the observed state to the control action. *Indirect controller* uses the system's dynamics model and some intermediate concepts to achieve the goal-directness of the controller and therefore corresponds to an indirect mapping from states to actions. Approaches like BOXES [10], ASE/ACE, [3] and behavioral cloning (reviewed in [6]) learn direct controllers. A controller using the generalized trajectory [16,17] is an indirect controller. The classical linear controller ([5], for example) can also be viewed as an indirect controller. These controllers use the system's dynamics model and intermediate concepts in the form of goal x_g (linear controller) or generalized trajectory. One could argue that a tree is an indirect mapping too, because it maps state x to a generalized state represented by a leaf in the tree and then maps this generalized state to action u . If we are lucky, the

generalized state is an understandable intermediate concept, perhaps a region in a state space where the operator takes similar actions. But the point is that it does not consider the system's dynamics and is not goal (or subgoal) directed.

In the case of a generalized trajectory, the controller is a composite of two functions: the trajectory $f^T : x \mapsto x^T$ and the inverse dynamics $f^D : x \times x^T \mapsto u$. The trajectory function maps the current state x_k to desired state x_{k+1}^T (note that typically x_{k+1}^T is not completely quantified; some of the state components can be omitted or just qualitatively described). The inverse dynamics function maps the current state x_k and the desired next state x_{k+1}^T to action u_k which aims to achieve x_{k+1}^T from x_k .

4 Robustness of Direct and Indirect Controllers against the Learning Error

Here we study and compare the performance of direct and indirect controllers learned from an operator's execution trace $((x_k^{op}, u_k^{op}), k = 0 \dots N)$. Direct controller generalizes the actions, while indirect controller generalizes the trajectory $((x_k^{op}), k = 0 \dots N)$. We will show that a direct controller is prone to the departure from the trajectory and that this departure is likely to happen, due to the learning error of the generalized action.

The operator takes the action u_k^{op} in the current state x_k^{op} that achieves the desired next state x_{k+1}^{op} . The learned controller mimics the operator and takes action u_k that achieves the desired next state with some error ϵ_{k+1} : $x_{k+1} = f(x_k, u_k) = x_{k+1}^{op} + \epsilon_{k+1}$.

Intuitively a direct controller is not likely to be successful, if the operator's action is applied in a state far from the operator's trajectory. This is because a property of many nonlinear dynamics systems is that an action applied far from the operator's trajectory tends to cause the system to move in a different direction than when the same action is applied on the trajectory. Since the generalized action u_k^D is learned with some error, and these errors are often propagated to the next state, the trajectory is likely to diverge from the operator's trajectory.

As opposed to the direct controller, an indirect controller's action u_k^I uses the learned dynamics at the current state $x_k = x_k^T + \epsilon_k$ when aiming to approach the next state on the trajectory x_{k+1}^T . Even if the current state x_k is far from the trajectory, the controller explicitly aims at decreasing the error at the next step. Since the system's dynamics model can be updated online, indirect controller can approach the trajectory even from states not seen in the operator's trace.

4.1 Experiment

Here we compare the robustness of direct and indirect controllers with respect to the learning error on an example of a simple dynamic system control. We consider a deterministic, discrete time dynamic system with the following dynamics:

$$\begin{aligned} x_{k+1} &= x_k + \dot{x}_k \\ \dot{x}_{k+1} &= x_{k+1} + u_k x_{k+1}^2 + u_k \end{aligned} \tag{3}$$

A similar system was used in [2] as an example of a nonlinear system that is easily controlled with the optimal linear controller (LQR) in the region near zero, but hard to control elsewhere in the state space.

The task is to control the system to reach and maintain the goal position $x_g = 0.5$ from the start state ($x_0 = \dot{x}_0 = 0$). The criterion of the controller's success is the controller's error c . Here we define c as the sum of absolute errors in x for the first $N = 30$ steps: $c = \sum_{k=0}^{N-1} |x_k - x_g|$. If $|x_k|$ exceeds 100 the error is set to its maximum $c = 100$. Since $x_0 = 0$ and $x_1 = 0$, $c \geq 1$.

We suppose that the operator uses a very simple control rule that does the task with $c = 1.004$:

$$u_k^{op} = \begin{cases} 0.5, & \text{if } x_k \leq 0 \text{ and } \dot{x}_k = 0 \\ -0.4, & \text{if } x_k > 0 \text{ and } |x_g - x_k| \leq 0.001 \\ -0.4 - 0.001 \operatorname{sign}(x_g - x_k), & \text{otherwise} \end{cases} \quad (4)$$

making the system to move approximately along the trajectory $((x_0, \dot{x}_0) \dots (x_{k+1}^{op}, \dot{x}_{k+1}^{op}) \dots)$, where:

$$\begin{aligned} x_{k+1}^{op} &= x_k + \dot{x}_k \\ \dot{x}_{k+1}^{op} &= \begin{cases} (x_g - x_k) - \dot{x}_k, & \text{if } x_k = 0 \\ 0.09(x_g - x_k), & \text{otherwise} \end{cases} \end{aligned} \quad (5)$$

A generalized trajectory is specified by constraints between the current and the next state. Since the control u_k directly affects only $\dot{x}_{k+1}$ (and not x_{k+1}), an appropriate generalized trajectory can be learned as a function mapping from $(x_k, \dot{x}_k)$ to the desired $\dot{x}_{k+1}^T$ and the indirect controller action u_k^I can be computed from the learned system's dynamics. In our experiments with indirect controllers we used a very simple kind of local learning [1,2] to learn (during the control) a local model of the system's dynamics near the current state $(x_k, \dot{x}_k)$: 10 examples $((x_j, \dot{x}_j), u_j, (x_{j+1}, \dot{x}_{j+1}))$ with the smallest quadratic distance $(x_j - x_k)^2 + (\dot{x}_j - \dot{x}_k)^2$ are used for linear regression to compute the parameters a, b, c, d of the local linear model $\dot{x}_{k+1} = ax_k + b\dot{x}_k + cu_k + d$ used at the point $(x_k, \dot{x}_k)$. The points used for local learning are continuously updated during the control, i.e. starting with the set $\{x_k^{op}, \dot{x}_k^{op}, u_k^{op}\}$ and adding new points during the control. The control u_k^I , aiming to achieve the desired next state on the generalized trajectory is then computed as $u_k^I = (\dot{x}_{k+1}^T - ax_k - b\dot{x}_k - d)/c$.

From the execution trace $((x_k^{op}, \dot{x}_k^{op}, u_k^{op}), k = 0, \dots, N)$, where rule 4 was used, the generalized operator's action u_k^D and the generalized operator's trajectory $\dot{x}_k^T$ can be learned with the learning errors ϵ^D and ϵ^T , i.e. generalization errors due to the generalization of the learning system. So the predicted values of the induced predictors are $u_k^D = u_k^{op} + \epsilon_k^D$ and $\dot{x}_k^T = \dot{x}_k^{op} + \epsilon_k^T$.

When experimenting with direct controllers that generalize the operator's action (rule 4) with some learning error ϵ_k^D , we noticed that they are very brittle w.r.t. the learning error and result in much higher controller errors than these in Table 1. The reason is that the rule 4 is purely reactive. In the rest of the

Table 1. The robustness of direct and indirect controllers against the learning error: c^D and c^I denote the controller error for the direct and the indirect controllers with the learning errors modeled as Gaussian noise with std.dev. σ . c^{D*} and c^{I*} denote the controller’s error where the predictors’ error was modeled by *biased* Gaussian noise. The results are averages of 20 runs

learning	direct controller		indir. controller	
error σ	c^D	c^{D*}	c^I	c^{I*}
0.0	1.005	1.005	1.005	1.005
10^{-3}	1.08	1.22	1.08	1.17
10^{-2}	1.6	3.3	1.6	3.1
0.1	9.5	58.2	8.7	18.3
0.2	18.9	> 100	20.0	39.8
0.5	86.5	> 100	83.5	> 100

experiments we used better control rule for u_k^{op} . It uses the system dynamics (eq. 3) to achieve the desired $\dot{x}_{k+1}^{op}$ from rule 5. In this way direct controllers were given the advantage of exact system model, while indirect controllers used a very simple method to approximate the system dynamics.

To compare the robustness of direct and indirect controllers with respect to the learning error we performed a set of experiments, where we modeled different error distributions of the learning system and measured the performance of both controllers. We experimented with two prediction error models. First the error was randomly generated as zero mean Gaussian noise with various standard deviations σ . The second model is the same as Gaussian noise, but with all the errors (in the same control trace) in the same directions, that is errors in the same control trace are either positive or negative, with random absolute value. We call this *biased* Gaussian noise. Biased Gaussian noise is obtained from Gaussian noise just by adjusting the sign (making the sign uniform for the whole control trace). Note that biased Gaussian noise thus has the same σ , but of course no longer has zero mean.

The rationale for defining our experiment in this way is as follows. We wanted to make our experiment independent of the particular learning technique, so we modeled the "induced" predictors simply by taking the correct value and corrupting it with noise. Gaussian noise as error model seem to be an obvious choice. The second model, biased Gaussian noise is, however, not quite so obvious. We designed this error model to account for a frequent property of induced continuity predictors: namely, that small changes in the attribute values usually do not cause large changes in the predicted class values. Therefore the errors at neighbour points have the same sign. The results in Table 1 show that Gaussian noise in the respective predictors affect both direct and indirect controllers in terms of their control cost to a similar extent. Biased Gaussian noise is more damaging to both controllers than zero mean Gaussian noise with the same σ . However, the direct controller is affected much more than the indirect controller. This happens, surprisingly, in spite of the fact that in addition to the error in

generalized trajectory, the indirect controller also has to cope with the error in the induced dynamics model. This result provides a plausible explanation why indirect controllers have proved to be much more successful in the experiments with learning techniques in the crane [17,16] and acrobot [15] domains.

We believe that this superiority of the indirect controller follows from their goal-directness (generalized operator's trajectory can be seen as the continuous subgoal) and their use of the system's dynamics model, which is learned online and thus enable them to behave well also in the yet unseen regions of the state space.

5 Why Learning Indirect Controllers Is Better than Learning Direct Controllers?

Here we would like to comment on the plausible advantages of learning indirect controllers. We believe that most of these advantages are consequences of the decomposition of learning an indirect controller into two natural subproblems: learning what to do (the trajectory) and learning how to do it (the system's dynamics). The only difficult task is to learn the generalized trajectory. As confirm the experimental results, the system's dynamics can be approximated sufficiently well by instance based methods of learning from execution traces and can be also updated online during the control. Advantages of learning indirect controllers are:

1. An indirect controller is less prone to the departure from the desired trajectory. Due to the operator's inconsistency (mistakes, taking different actions in similar situations) and due to the learning errors the learned trajectory or the action is not perfect. When taking an incorrect action the system often departs from the desired trajectory and arrives to the yet unseen region of the state space. Indirect controller is goal directed and considers the system's dynamics to take an action back towards the trajectory. A direct controller has no clue how to get back to the known region of the state space and often just gets stuck in the unknown region.
2. An indirect controller is more robust against change in the system's dynamics and small changes in the task. The robustness against change in the system's dynamics is ensured by updating the system's dynamics model during the control. The robustness against small changes in the task (like changing the system start state) follows from the better robustness of the indirect controller against the departure from the desired trajectory. This of course assumes that the learned generalized trajectory is still appropriate for the changed system's dynamics or the changed task.
3. Generalizing the trajectory is often easier than generalizing the actions. In many systems different actions can result in similar effect on the system behavior. When selecting the action to take, an experienced operator keeps in mind only the desired effect. In similar situations he often uses different actions to achieve the same effect. These different actions in similar states are

noise for the learning system if it learns the mapping from the system's states to the actions, but are not noise when learning the mapping from the system's states to the desired effects, that is generalized trajectory. For example, when controlling a crane, the operator can use quite different actions to slow down the trolley: the sequence of oscillating actions around $Force_X=2000$ can result in similar behavior as a sequence of actions $Force_X=2000$.

4. Learning an indirect controller seems usually to be an easier task than learning a direct controller. When learning an indirect controller, the only difficult task is to learn the generalized trajectory, that is what the operator is doing. In the case of a direct controller the task is to learn what the operator is doing and how he does it. This seems to be a more difficult learning task.

In addition indirect controllers have other advantages:

1. The learned trajectory is easier to understand than the actions, which usually involve knowledge of the system's dynamics. The desired trajectory usually captures the knowledge of performing the given task in a more compact way. When driving the car on the parking space from one point to the other, it is easier to understand the (x,y) trajectory, than the sequence of actions like pushing the gas pedal, turning the wheel for 30 degrees and straightening it after, pushing the breaks, turning the wheel again,...
2. Knowledge learned by generalized symbolic trajectory can easily be adopted to new tasks. For example, in the crane domain, the trajectory for attaining the goal position at $X=60$ can easily be adopted to goal position at $X=80$.

6 Related Work

Another approach that deals with learning to control dynamic systems is reinforcement learning [9]. Unlike behavioural cloning, reinforcement learning learns control from scratch, through trial-and-error. The idea is to use dynamic programming to learn the value function or Q -function estimating how promising (in achieving the goal) each state or action is. In the most common approach they learn direct controllers. To reduce experimentation with the dynamic system, model-based reinforcement learning methods [12] also use learned model of system dynamics and therefore effectively learn indirect controllers. However, reinforcement learning does not use human operator's control skill and is not concerned with the comprehensibility of the learned strategy.

A similar idea of decomposing the cloning problem into two learning problems appears in [4]. They use abduction to learn effects of control actions and decision trees to learn subgoals. However, they do not report on benefits regarding the success of controllers constructed in this way.

Acknowledgments

The results in this paper are part of the first author's work on PhD thesis at University of Ljubljana, Faculty of Computer and Information Science. Both authors were supported by Slovenian Ministry of Science and Technology.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. *Machine Learning* 6:37–66 (1991). 387
2. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning for control. *Artificial Intelligence Review* 11:75–113 (1997). 387
3. Barto, A., Sutton, R., Anderson, C.: Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics* SMC-13(5):834–846 (1983). 385
4. Bain, B., Sammut, C.: A framework for behavioural cloning. *Machine Intelligence 15* Oxford University Press. Forthcoming. 390
5. Bertsekas, D.: *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs: Prentice Hall (1987). 385
6. Bratko, I., Urbančič, T., Sammut, C.: Behavioural cloning of control skill. In Michalski, R.; Bratko, I.; and Kubat, M., eds., *Machine Learning and Data Mining: Methods and Applications*, 335–351. John Wiley & Sons, Ltd (1998). 382, 385
7. Chambers, R., Michie, D.: Man-machine co-operation in a learning task. *Computer Graphics: Techniques and Applications* 179–186 (1969). 382
8. Donaldson, P.: Error decorrelation studies on a human operator performing a balancing task. *Med. Electron. Biol. Engng.* 2:393–410 (1964). 382
9. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–295 (1996). 390
10. Michie, D., Chambers, R.: Boxes: an experiment in adaptive control. In Dale, E., and Michie, D., eds., *Machine Intelligence 2*, 137–152. Edinburgh University Press (1968). 385
11. Sammut, C., Hurst, S., Kedzier, D., Michie, D.: Learning to fly. In *Proceedings of the 9th International Workshop on Machine Learning*, 385–393. Morgan Kaufmann (1992). 382
12. Sutton, R.: Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, 216–224. Morgan Kaufmann (1990). 390
13. Urbančič, T., Bratko, I.: Reconstructing human skill with machine learning. In Cohn, A., ed., *Proceedings of the 11th European Conference on Artificial Intelligence*, 498–502. John Wiley & Sons, Ltd (1994). 382
14. Šuc, D., Bratko, I.: Skill reconstruction as induction of LQ controllers with subgoals. In Mellish, C. S., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, volume 2, 914–920 (1997). 383
15. Šuc, D., Bratko, I.: Skill modelling through symbolic and qualitative reconstruction of operator's trajectories. Technical report, Artificial Intelligence Lab., Faculty of Computer and Information Sc., University of Ljubljana, Slovenia (1998). 383, 389
16. Šuc, D., Bratko, I.: Modelling of control skill by qualitative constraints. In *Proceedings of the 13th International Workshop on Qualitative Reasoning*, 212–220. University of Aberystwyth. Loch Awe, Scotland (1999). 385, 389
17. Šuc, D., Bratko, I.: Symbolic and qualitative reconstruction of control skill. *Electronic Transactions on Artificial Intelligence* 3. Forthcoming. 383, 385, 389

Dynamic Feature Selection in Incremental Hierarchical Clustering

Luis Talavera

Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya
Campus Nord, Mòdul C6, Jordi Girona 1-3, 08034 Barcelona, Spain
`talavera@lsi.upc.es`

Abstract. Feature selection has received a lot of attention in the machine learning community, but mainly under the supervised paradigm. In this work we study the potential benefits of feature selection in hierarchical clustering tasks. Particularly we address this problem in the context of incremental clustering, following the basic ideas of Gennari [8]. By using a simple implementation, we show that a feature selection scheme running in parallel with the learning process can improve the clustering task under the dimensions of accuracy, efficiency in learning, efficiency in prediction and comprehensibility.

1 Introduction

The performance of inductive learning algorithms heavily depends on the features used to describe the training data. As widely reported in the literature, most algorithms are known to degrade in performance when faced with features that are not useful for the task at hand. Ideally, one would like to provide the algorithm only with features containing useful information. However, there are many applications where experts make arbitrary choices or simply there are too many features to be processed by hand, so that automatic *feature selection* methods are needed.

Feature selection has received a lot of attention in the machine learning community as reflected in the huge number of works in the area as reviewed for instance in [1,2]. However, most of these works address the problem of supervised learning, and we can find only a few works devoted to unsupervised learning.

In this paper we address the particular problem of unsupervised feature selection in incremental hierarchical clustering. Given the nature of this sort of algorithms, we study a *dynamic* feature selection method that runs in parallel with learning. We follow the general guidelines proposed by Gennari [8] who proposed a dynamic feature selection method but only evaluated its merits with few datasets. We extend the testing procedure by adding some new dimensions to evaluate the benefits of feature selection and using several UCI datasets typically used for these purposes in supervised learning. Rather than develop an optimal method for feature selection, our work aims to explore the potential benefits of dynamic feature selection in clustering by using a more simple scheme than Gennari's but powerful and flexible enough to draw some interesting conclusions.

2 Supervised and Unsupervised Learning

The most common type of inductive learning problems are supervised *classification* problems. Given a set of labeled training examples the goal is to build a classification model that is able to correctly classify new, unseen instances. Evaluation of supervised learning systems is done by measuring the *accuracy* of the system. The system is provided with a separate set of instances usually called the test set, that is used to predict the label of each instance. Accuracy is estimated from the proportion of correct predictions over the total number of instances.

Unlike supervised learners, unsupervised algorithms do not have access to labeled examples. In unsupervised learning there are no target outputs associated with the inputs, and systems must resort to internal biases to decide which relationships should be represented in the output. This makes very difficult to define a widely accepted method for evaluating unsupervised learners and, particularly, clustering systems.

A first and widely used method for evaluating clustering systems is to compute predictive accuracy as is done for supervised classifiers. In order to apply this procedure a dataset with a known class structure must be used. The system is provided with a training set with the labels masked out from which a model is built. After learning, each cluster created by the system is labeled with the majority value for the class attribute and then the model is used to predict the label of instances in a test set. The resulting accuracy serves as a measure of how well the system has discovered the (known) underlying structure in the dataset. Alternatively, instead of using the system for prediction, after labeling the clusters, the proportion of incorrectly placed instances is computed as the accuracy of the system. This method is commonly used in statistics, since most statistical clustering approaches are not intended to make predictions.

A not so popular but not less interesting evaluation criterion is *flexible prediction* or *pattern completion* [6,11]. Since in unsupervised learning there is no *a priori* target feature, it appears natural to consider that clustering or unsupervised systems in general, may support inference of any unknown feature value. A performance measure for this task is the average prediction accuracy over all the features. In order to compute this measure, each feature present in the data is masked out and the discovered model is used to predict its value from the information provided for the rest of features as if it was a “label”. The average of each individual accuracy for all the features is then taken as the overall predictive accuracy of the system. Note that flexible prediction is a much more complex task than label prediction, since multiple targets must be predicted from a single model. As an example, if you consider that a supervised classifier might be given the concept of “tiger” and then recognize this animal from observed features, from the viewpoint of flexible prediction, a clustering system should be able to predict, for example, that the animal is dangerous from the other observed features without even knowing the concept of “tiger” in advance.

Of course, other concerns may be particularly interesting in unsupervised learners, such as the *comprehensibility* of the results for humans, given that they

discover completely new knowledge. In some cases users may be interested in the *descriptive* aspect of the results rather in their predictive power. Additionally, for particular applications there can be task-specific constraints. However, these aspects are usually very difficult to evaluate with numerical measures and often evaluation is very subjective.

3 Feature Selection in Hierarchical Clustering

At a conceptual level, the feature selection problem is similar for both supervised and unsupervised learners. Considering feature selection as a heuristic search in a space of feature subsets, any method, supervised or unsupervised, requires an starting point in the space, a search strategy, an evaluation function and a stopping criterion [1]. Under this view, unsupervised feature selection methods could be designed by adapting existing supervised methods and adding a few task-specific modifications. However, in practice, the adaptation of the evaluation function is not straightforward, since all the existing criteria rely on assessing how well a given feature subset discriminates among a set of predefined classes that are not available for unsupervised learners. In fact, the problem stems from a more general issue related to the performance task associated with each type of learning. As we have mentioned, in supervised learning, the predictive accuracy over class labels is a widely accepted performance task, so it is relatively easy to design evaluation functions. On the contrary there is a lack of a generally accepted performance task for clustering systems. For the rest of the paper we will focus on the three predictive tasks presented above: accuracy over labels, flexible prediction and comprehensibility.

Typically, the primary goal of feature selection is intended to make inductive learning algorithms more robust in the face of irrelevant features. There are a number of formal definitions of the relevance of features in the literature [9], although all of them are addressed to supervised tasks. There is no standard definition of irrelevance for flexible prediction tasks and, in fact, it is not clear if a system can built a clustering using a reduced feature set and still predict all the features originally present in the data. Therefore, for the rest of the paper we will resort to an intuitive notion of relevance, considering that a feature is relevant if it cannot be removed without loss of prediction accuracy of any kind.

There is an important factor related to the organization of the knowledge base in hierarchical clusterers. Commonly, hierarchical clusterings are *polythetic* classifiers, that is, they divide objects based on their values along multiple features. Particularly, they tend to use the full set of features at each node to decide how to classify a new object. Note that, while in *monothetic* classifiers such as decision trees, a redundant feature adds one additional test when classifying a new observation, in polythetic classifiers it adds a test for each node in the classification path. Clearly, improving performance may be a motivation for applying feature selection to clustering tasks, but not the only one. In general, the hierarchical organization, the polythetic nature of clusterings and the performance task determine several dimensions for evaluating the particular benefits of fea-

ture selection in conceptual clustering. Following [14,15] we can summarize these dimensions as follows:

- *Performance.* The set of features used in an inductive learning task is a powerful representational bias that determines the performance of a learning system. Irrelevant features may be particularly harmful in unsupervised systems, leading the system to form wrong patterns and having an impact in prediction that may be especially significant in a multiple inference task.
- *Efficiency in the learning task.* We have noted that hierarchical clusterings are polythetic classifiers. Since the decision of how to classify a new object has to be made along several nodes in the tree, the number of features present in the data strongly influences the complexity of the clustering process. If we apply feature selection to reduce this complexity, we should expect to obtain clusterings with at least similar performance that we would have obtained by using all the available features.
- *Efficiency in the performance task.* When using a hierarchical clustering to classify unobserved objects in order to infer unknown properties, the number of features has a strong influence in the complexity of the process in the same manner we have described above. Again, selecting an appropriate subset of features may reduce this complexity while maintaining the original performance level.
- *Comprehensibility of the results.* Clustering systems usually make use of all the available features at each node of the hierarchy. Reducing the number of features used in the clustering process allow to provide shorter cluster descriptions to the user. Short descriptions tend to be more readable and, hence more comprehensible.

4 Dynamic Feature Selection in Incremental Clustering

Typically, supervised feature selection methods are *static* in the sense that they are applied just once before the final induction task is carried out. The set of features obtained from the selection procedure is then fixed and never changes during learning. An alternative is to implement feature selection as a procedure that runs in parallel with learning. This approach allows the feature selection mechanism to *dynamically* adapt the set of selected features in the light of the knowledge gathered during the learning process. The dynamic feature selection procedure is then triggered at each *learning step*, that may differ from system to system. For example, in an incremental system, a learning step may be the incorporation of a new object, while in a batch agglomerative algorithm it may be a local merging operation. Interestingly, dynamic methods are the only methods that do not compromise the incremental operation of clustering systems that work in this way. Dynamic feature selection schemes may be very sensitive to wrong initial decisions biasing the system towards bad learning paths. However, potentially, they are a very attractive alternative since they can improve the clustering task on all the four dimensions presented in Section 3.

On the other hand, the hierarchical organization of the knowledge base in clustering allows to represent relatively complex descriptions of the environment at several levels of abstraction. By dividing the object space into local regions of variable generality, hierarchical clusterers provide a more expressive representation than flat clusterers. This property suggests that features that could be relevant at certain parts of the object space, might be useless at other regions. Thus, *local* feature selection methods that select different subsets of features for different nodes in the hierarchy appear particularly interesting, since they can be applied even when all the features in the data set are necessary for the clustering task. As polythetic classifiers, hierarchical clusterings may obtain great benefits from a local feature selection scheme even when none of the features in the original set is definitely removed. Dynamic feature selection naturally suggests to employ a local feature selection scheme, since they take local decisions at each learning step. It is worth noticing that local feature selection methods are more expensive than global ones. A local method must perform the same process that a global one as many times as nodes are in the tree. Moreover, if they are dynamically applied as well, one must take care of not employing very expensive procedures.

5 A Simple Dynamic Feature Selection Mechanism

In this section we will propose a simple implementation for a dynamic feature selection scheme. We implemented this method on the top of the well-known incremental clustering system COBWEB.

COBWEB is a hierarchical clustering system that constructs a tree from a sequence of objects. The system follows a strict *incremental* scheme, that is, it learns from each object in the sequence without reprocessing previously seen objects. An object is assumed to be a vector of nominal values V_{ij} along different features A_i . COBWEB employs *probabilistic concept* descriptions to represent the learned knowledge. In this sort of representation, in a cluster C_k , each feature value has an associated conditional probability $P(A_i = V_{ij} \mid C_k)$ reflecting the proportion of objects in C_k with the value V_{ij} along the feature A_i .

The strategy followed by COBWEB is summarized in Table 1. Given an object and a current hierarchical clustering, the system categorizes the object by sorting it through the hierarchy from the root node down to the leaves. At each level, the learning algorithm evaluates the quality of the new clustering resulting from placing the object in each of the existing clusters, and the quality resulting from creating a new cluster covering the new object. In addition, the algorithm considers two more actions that can restructure the hierarchy in order to improve its quality. *Merging* attempts to combine the two sibling clusters which were identified as the two best hosts for the new object; *splitting* can replace the best host and promote its children to the next higher level. The option that yields the high quality score is selected and the procedure is recursed, considering the best host as the root in the recursive call. The recursion ends when a leaf containing only the new object is created.

Table 1. The control strategy of COBWEB

Function Cobweb(object,root)

- 1) Incorporate object into the root cluster.
 - 2) **If** root is a leaf **then**
 return expanded leaf with the object.
 else choose the best of the following operators:
 - a) Incorporate the object into the best host
 - b) Create a new disjunct based on the object
 - c) Merge the two best hosts
 - d) Split the best host
 - 3) If a), c) or d) recurse on the chosen host.
-

In order to choose among the four available operators, COBWEB uses a cluster quality function called *category utility* defined for a partition $P = \{C_1, C_2, \dots, C_n\}$ of n clusters as

$$\frac{\sum_k P(C_k) \sum_i \sum_j [P(A_i = V_{ij} | C_k)^2 - P(A_i = V_{ij})^2]}{n} \quad (1)$$

This function measures how much a partition P promotes inference and rewards clusters C_k that increase the predictability of feature values within C_k . By using this metric, the system should be biased towards the construction of clusters allowing accurate predictions along any unobserved features.

As in supervised feature selection, feature selection in clustering can be done by using the so-called *filter* or *wrapper* models [9]. Briefly, filter models are independent of the induction algorithm that will use their output and they employ some metric dependent on intrinsic properties of the data. Typically, they measure the correlation of each feature with the class label by using distance, information or dependence measures. Obviously, the absence of class labels makes infeasible to compute these sort of measures in unsupervised learning and, therefore, alternative measures not using class information need to be defined.

On the other hand, in the wrapper model, the feature selection algorithm works as a wrapper around the induction algorithm. Alternative feature subsets are evaluated by using the induction algorithm as a black box over the training data in order to obtain an estimate of future performance. Usually, performance is estimated by measuring the predictive accuracy over class labels. Note that unsupervised learners cannot use these methods in the label prediction performance task, since they have no access to the labels during learning. Wrappers can be used for flexible prediction, albeit at the price of a high computational cost to estimate accuracy over the full feature set.

We propose a filter method of feature selection based on an *ordering* scheme. A weight is individually computed for each feature and features are ordered

Table 2. A method for feature selection based on an ordering scheme

Let $\mathcal{A}$ be a set of features
 Let τ be the feature selection threshold
Function `select_features( $\mathcal{A}, \tau$ )`
 `compute_feature_weights( $\mathcal{A}$ )`
 $max\_w = \max\{weight(A_i) \mid A_i \in \mathcal{A}\}$
 return $\{A_i \mid weight(A_i) \geq max\_w \times \tau\}$

according to these weights. We define a *feature selection threshold* τ in the $[0,1]$ range such that the weight required for a feature to be selected is higher for higher τ values. Our method uses the maximum computed weight as a baseline to determine which features are selected as shown in Table 2. Note that, if we assume relevances to be positive, when $\tau = 0$ there is no feature selection at all, so reducing the original algorithm to a special case of our approach.

This method can be easily incorporated into COBWEB by slightly modifying the control strategy showed in Table 1. First, we need to add an additional step between steps 2 and 3 of the existing algorithm. In this step a call to the *select_features* function is performed, obtaining a subset of relevant features to be stored in the current root node. Second, at each classification step, the computation of the quality function must be modified in such a way that only the subset of relevant features stored in the current root node is used.

The weighting function we use is the one proposed by Gennari [8] in the context of his CLASSIT system, an extension of COBWEB to deal with numeric features. Gennari refers to this measure as *salience*. He defines the relative salience of a feature as its contribution to category utility (see equation 1) in a clustering. More formally, for a given feature A_i , salience is defined as follows:

$$\frac{\sum_k P(C_k) \sum_j [P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2]}{n} \quad (2)$$

6 Experiments

In order to evaluate dynamic feature selection, we ran experiments on six datasets from the UCI repository: cleveland, crx, horse colic, hypothyroid, pima and wisconsin diagnostic breast cancer. Our aim was to test the potential of feature selection regarding the dimensions presented in Section 3. As regards to performance, we performed separated experiments for label and flexible prediction. In order to evaluate the efficiency of the learning and prediction processes, we computed the *average number of feature tests* needed to sort the instances in the training or testing set. This number is calculated by summing the total number of features involved in evaluating the category utility metric for the different

clustering choices. For instance, if an observation is being clustered in a root node with three children and using a subset of n features, we need to perform $3n$ feature tests to evaluate the CU of incorporating the observation to each of the siblings. In learning, additional feature tests are needed to evaluate creating a new cluster, merging and splitting. We think that this way of measuring efficiency give us a better empirical approximation of the complexity of the clustering process than, for instance, the average of features per node. On the other hand, we use this later measure as a measure of *comprehensibility* of the obtained clusterings, since fewer features per node indicate simpler cluster descriptions.

Table 3. Accuracy in label prediction, average number of tests in learning and prediction, features per node and number of nodes for several datasets and τ values

Dataset	τ	Accuracy	Tests learning	Tests pred	Feat./node	Nodes
cleve	n/a	74.73 (5.05)	1619.94 (42.94)	720.40 (45.02)	13.00 (0.00)	107.60 (2.91)
	0.1	75.27 (5.06)	1161.23 (96.60)	450.81 (59.67)	5.58 (0.09)	126.30 (2.26)
	0.2	76.04 (4.60)	928.64 (117.88)	363.80 (45.76)	5.22 (0.17)	129.30 (4.08)
	0.3	75.93 (3.17)	712.32 (55.35)	246.75 (24.73)	4.90 (0.12)	134.30 (2.71)
	0.4	74.18 (3.08)	531.78 (40.33)	190.97 (14.89)	4.53 (0.14)	135.60 (5.58)
	0.5	73.19 (3.56)	396.53 (18.12)	130.61 (7.05)	3.99 (0.20)	141.80 (2.74)
crx	n/a	80.24 (2.89)	2226.49 (59.25)	950.22 (35.62)	15.00 (0.00)	255.30 (5.96)
	0.1	78.74 (4.57)	1070.76 (66.12)	388.48 (31.81)	4.56 (0.08)	297.80 (7.44)
	0.2	79.86 (2.93)	852.89 (62.22)	286.32 (23.67)	4.20 (0.13)	302.60 (6.92)
	0.3	80.87 (3.35)	653.60 (33.76)	211.09 (15.19)	3.91 (0.15)	307.10 (4.77)
	0.4	78.89 (3.28)	486.00 (25.49)	156.69 (8.43)	3.69 (0.10)	314.00 (5.58)
	0.5	78.41 (2.75)	392.04 (23.29)	121.83 (8.00)	3.35 (0.08)	327.40 (5.13)
horse	n/a	74.23 (4.60)	3108.48 (79.31)	1371.85 (126.94)	22.00 (0.00)	124.30 (3.71)
	0.1	72.52 (2.66)	2011.35 (78.56)	797.53 (54.58)	9.18 (0.23)	149.40 (1.96)
	0.2	75.95 (3.85)	1548.66 (102.93)	602.92 (49.33)	8.52 (0.26)	150.60 (3.34)
	0.3	75.68 (3.89)	1151.87 (45.39)	420.08 (33.19)	7.95 (0.16)	158.40 (3.98)
	0.4	72.16 (3.05)	882.24 (86.90)	304.25 (24.66)	7.33 (0.17)	161.30 (3.33)
	0.5	72.61 (3.30)	612.11 (31.35)	200.78 (17.39)	6.54 (0.21)	170.50 (2.99)
hypo	n/a	97.65 (0.48)	8448.86 (248.21)	3952.57 (234.33)	25.00 (0.00)	1825.50 (13.95)
	0.1	97.65 (0.34)	3867.14 (229.60)	2024.41 (243.95)	18.46 (0.18)	1912.00 (14.45)
	0.2	97.54 (0.43)	3769.29 (311.04)	2035.68 (327.15)	18.33 (0.24)	1928.40 (14.03)
	0.3	97.61 (0.44)	3467.92 (220.02)	1873.79 (338.49)	18.13 (0.25)	1954.10 (11.51)
	0.4	97.47 (0.46)	3407.59 (250.65)	1868.41 (321.50)	17.89 (0.21)	1979.80 (10.67)
	0.5	97.46 (0.44)	3183.58 (343.08)	1643.02 (348.78)	17.60 (0.23)	2011.50 (8.66)
pima	n/a	65.11 (2.62)	1135.25 (21.92)	470.92 (15.74)	8.00 (0.00)	321.80 (7.32)
	0.1	64.94 (2.70)	723.84 (51.84)	256.16 (24.38)	3.61 (0.08)	347.70 (5.85)
	0.2	66.06 (2.94)	571.35 (45.10)	195.13 (19.45)	3.35 (0.08)	358.80 (5.49)
	0.3	64.85 (3.26)	448.13 (22.87)	150.23 (10.38)	3.24 (0.07)	365.70 (6.93)
	0.4	66.32 (2.10)	365.34 (30.23)	117.43 (8.09)	3.10 (0.10)	372.10 (6.28)
	0.5	65.93 (3.40)	274.59 (16.04)	89.56 (4.89)	2.93 (0.07)	384.40 (7.90)
wdbc	n/a	91.93 (1.55)	4287.82 (79.96)	1881.39 (70.54)	30.00 (0.00)	198.00 (6.73)
	0.1	91.93 (1.80)	2839.01 (196.73)	1123.98 (62.42)	12.67 (0.11)	235.10 (4.18)
	0.2	92.57 (1.20)	2249.09 (70.47)	864.46 (57.95)	11.68 (0.30)	240.30 (5.50)
	0.3	91.17 (2.69)	1858.66 (95.20)	663.93 (75.65)	10.97 (0.29)	247.70 (4.52)
	0.4	90.99 (2.34)	1362.12 (93.71)	472.08 (36.45)	9.95 (0.25)	256.00 (5.33)
	0.5	90.99 (1.66)	1013.66 (40.44)	335.47 (21.03)	8.91 (0.23)	266.70 (8.25)

In all the experiments, we used a 70% of the instances for training and a 30% for testing. All the results shown are averages over 20 independent runs using random splits. For each dataset, several values of the τ parameter were used to gain some insight into the effect of different degrees of selection on the

performance of the system. The system never had access to the labels neither in label nor flexible prediction, although in the later case they were used for evaluation purposes. In the case of flexible prediction, accuracy is always computed as the overall accuracy over all the original features in the data, regardless of the features removed during the feature selection process. Since the CU measure is only applicable to nominal features, all numeric features were discretized.

Table 3 shows the results for the label prediction performance task. At a glance, we can observe that in all datasets accuracy can be maintained or improved while reducing the number of features per node used to an average of the 40% of the original number of features. As expected, this reduction implies an improvement in the efficiency of the system in both learning and prediction. In average, dynamic feature selection provides an efficiency improvement of about the 50% in learning and prediction. Note that the feature selection scheme produces changes in the structure of the hierarchies created by increasing the number of inner nodes. This increment reduces partially the efficiency gains obtained with the removal of features, since the complexity of sorting an instance depends on the depth of the hierarchy as well.

As a conclusion, we can say that dynamic feature selection can provide benefits in the clustering task along the four proposed dimensions for evaluation as regards the label prediction task. Our results for this task as regards efficiency agree with the results of Gennari[8]. The potential for creating accurate clusterings for this task is also shown in [13], although using a more classical pre-processing approach. As we have noted, obtaining such results with a dynamic incremental scheme is quite impressive given the greedy nature of the method.

Table 4 shows the results for the flexible prediction task. Again, the feature selection scheme is able of creating simpler clusterings without hurting accuracy. These results are even more remarkable than the previous ones, given the multiple inference task now required for the clusterings. Efficiency in prediction is improved in a similar amount that for label prediction. Obviously, the improvements in the efficiency in learning and in the number of features per node remain the same, since the learning task is identical for both performance tasks.

Therefore, we can conclude also that dynamic feature selection is able to improve the clustering task along the four evaluation dimensions as regards flexible prediction. We have obtained similar results by using a postprocessing approach [15]. Although such an approach is less prone to bad decisions while simplifying the hierarchy, it cannot improve the efficiency of learning as dynamic feature selection does.

7 Related Work

The idea of focusing on particular features in incremental unsupervised learning can be traced back to early influential work by Kolodner [10] on CYRUS and Lebowitz [12] on UNIMEM. As we have pointed out, Gennari [8] proposed a more general and principled mechanism that inspired this work. Fisher *et al* [5] adapted

Table 4. Accuracy in flexible prediction, average number of test in learning and prediction, features per node and number of nodes for several datasets and τ values

Dataset	τ	Accuracy	Tests learning	Avg. tests pred	Feat./node	Nodes
cleve	n/a	48.78 (1.93)	1619.94 (42.94)	8631.64 (502.74)	13.00 (0.00)	107.60 (2.91)
	0.1	49.10 (2.44)	1161.23 (96.60)	5417.78 (719.98)	5.58 (0.09)	126.30 (2.26)
	0.2	49.04 (1.65)	928.64 (117.88)	4342.31 (515.53)	5.22 (0.17)	129.30 (4.08)
	0.3	48.83 (1.64)	712.32 (55.35)	3019.06 (295.23)	4.90 (0.12)	134.30 (2.71)
	0.4	48.86 (1.87)	531.78 (40.33)	2400.76 (171.30)	4.53 (0.14)	135.60 (5.58)
	0.5	50.23 (2.52)	396.53 (18.12)	1750.43 (84.59)	3.99 (0.20)	141.80 (2.74)
	0.6	49.88 (1.12)	312.49 (33.26)	1440.42 (86.37)	3.72 (0.15)	145.60 (5.21)
crx	n/a	60.72 (1.23)	2226.49 (59.25)	13302.06 (491.47)	15.00 (0.00)	255.30 (5.96)
	0.1	60.74 (1.28)	1070.76 (66.12)	5427.68 (442.28)	4.56 (0.08)	297.80 (7.44)
	0.2	61.25 (0.88)	852.89 (62.22)	4028.86 (307.44)	4.20 (0.13)	302.60 (6.92)
	0.3	61.17 (1.20)	653.60 (33.76)	3059.85 (202.27)	3.91 (0.15)	307.10 (4.77)
	0.4	60.99 (1.27)	486.00 (25.49)	2353.99 (81.73)	3.69 (0.10)	314.00 (5.58)
	0.5	60.80 (0.92)	392.04 (23.29)	1930.33 (85.65)	3.35 (0.08)	327.40 (5.13)
horse	n/a	59.17 (1.05)	3108.48 (79.31)	28738.65 (2643.82)	22.00 (0.00)	124.30 (3.71)
	0.1	59.94 (1.21)	2011.35 (78.56)	16762.54 (1159.86)	9.18 (0.23)	149.40 (1.96)
	0.2	59.30 (1.85)	1548.66 (102.93)	12618.22 (983.83)	8.52 (0.26)	150.60 (3.34)
	0.3	58.89 (0.93)	1151.87 (45.39)	8865.25 (646.47)	7.95 (0.16)	158.40 (3.98)
	0.4	58.12 (1.17)	882.24 (86.90)	6562.92 (516.68)	7.33 (0.17)	161.30 (3.33)
	0.5	58.10 (0.97)	612.11 (31.35)	4473.22 (320.96)	6.54 (0.21)	170.50 (2.99)
hypo	n/a	83.05 (1.05)	8448.86 (248.21)	87296.71 (5996.60)	25.00 (0.00)	1825.50 (13.95)
	0.1	84.71 (0.51)	3867.14 (229.60)	45223.19 (5571.86)	18.46 (0.18)	1912.00 (14.45)
	0.2	85.12 (0.83)	3769.29 (311.04)	46351.35 (7007.69)	18.33 (0.24)	1928.40 (14.03)
	0.3	84.44 (0.90)	3467.92 (220.02)	41475.52 (7199.13)	18.13 (0.25)	1954.10 (11.51)
	0.4	84.59 (0.67)	3407.59 (250.65)	41225.04 (6659.03)	17.89 (0.21)	1979.80 (10.67)
	0.5	83.69 (1.08)	3183.58 (343.08)	36539.31 (7635.42)	17.60 (0.23)	2011.50 (8.66)
pima	n/a	45.61 (1.27)	1135.25 (21.92)	3299.04 (104.96)	8.00 (0.00)	321.80 (7.32)
	0.1	45.37 (1.35)	723.84 (51.84)	1801.95 (160.45)	3.61 (0.08)	347.70 (5.85)
	0.2	45.82 (1.46)	571.35 (45.10)	1379.61 (126.57)	3.35 (0.08)	358.80 (5.49)
	0.3	45.85 (1.33)	448.13 (22.87)	1107.40 (81.37)	3.24 (0.07)	365.70 (6.93)
	0.4	45.98 (0.99)	365.34 (30.23)	935.74 (39.89)	3.10 (0.10)	372.10 (6.28)
	0.5	46.20 (1.35)	274.59 (16.04)	794.64 (22.02)	2.93 (0.07)	384.40 (7.90)
	0.6	46.34 (1.32)	232.07 (13.29)	733.15 (24.28)	2.79 (0.06)	393.70 (7.60)
wdbc	0.7	46.38 (1.28)	191.36 (6.65)	707.61 (24.26)	2.56 (0.09)	398.60 (8.98)
	n/a	62.34 (1.57)	4287.82 (79.96)	54604.95 (2023.58)	30.00 (0.00)	198.00 (6.73)
	0.1	62.96 (0.99)	2839.01 (196.73)	32584.53 (1781.38)	12.67 (0.11)	235.10 (4.18)
	0.2	62.89 (0.64)	2249.09 (70.47)	25051.45 (1661.10)	11.68 (0.30)	240.30 (5.50)
	0.3	62.41 (0.92)	1858.66 (95.20)	19308.05 (2229.57)	10.97 (0.29)	247.70 (4.52)
	0.4	61.90 (1.04)	1362.12 (93.71)	13703.37 (1061.79)	9.95 (0.25)	256.00 (5.33)
	0.5	61.43 (1.49)	1013.66 (40.44)	9833.95 (582.14)	8.91 (0.23)	266.70 (8.25)

Gennari’s procedure to a diagnosis task, where the intent was to minimize the number of probes necessary to diagnose a fault.

As in supervised learning, preprocessing approaches are more common as in [3], [4] or [13]. However, neither of these works have been extensively evaluated along all the dimensions proposed here. As regard the flexible prediction task, the only existing work is [16] with a weak evaluation and our own work in preprocessing and postprocessing methods [14,15]. Although the later works used different data sets to those that were used in this paper, at first sight, dynamic feature selection appears to be a good alternative to these methods.

8 Concluding Remarks

Feature selection methods have shown successful in supervised approaches and we have shown that they could be also useful in incremental hierarchical cluster-

ing despite the difficulties posed by unsupervised settings. Besides the traditional aim of increasing accuracy, we have proposed other dimensions for evaluating this benefits mainly concerned with efficiency and comprehensibility. In addition, hierarchical clusterings suggest a local feature selection scheme for each node in the hierarchy. Moreover, given that unsupervised systems support inference on more than one single dimension, we have shown the benefits of feature selection in both classical label prediction and flexible prediction, the later involving prediction of all the features in the data.

To maintain the incremental nature of the system we have applied a dynamic feature selection scheme that runs parallel to the clustering process instead of being a preprocessing step, as typically done in supervised learning. Results show that this mechanism can improve efficiency in learning, efficiency in prediction and comprehensibility while maintaining or improving performance in prediction.

All of these results have been obtained with a simple and rough implementation that can be clearly improved, although it has served well for the purpose of this study. Surely, a smarter feature selection method can be designed, ideally without having to set any thresholds. Gennari's original method is one possible alternative to test. Additionally, since the salience measure is derived from the objective function used in clustering (CU in this case), it would be interesting to test alternative objective functions to CU to see if they are better candidates not only for evaluating clusters but for evaluating features as well.

Finally, Fisher's work [7] suggests further implications about the relationship between feature selection and using feature frontiers for prediction. If a feature can be predicted accurately at a certain node without descending deeper into the hierarchy, this feature is not informative to discriminate between descendant nodes. We have noted the importance of the structure of the hierarchy only at a cursory level, but future work should explore this issue by including additional dimensions for cost assessment such as the branching factor or the depth of the hierarchies.

Acknowledgements

The author thanks one anonymous referee for providing important pointers to relevant work and its implications on feature selection. This work is partially supported by UPC grant PR99-09.

References

1. A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997. 392, 394
2. M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3), 1997. 392
3. M. Dash, H. Liu, and J. Yao. Dimensionality reduction for unsupervised data. In *Ninth IEEE International Conference on Tools with AI, ICTAI'97*, 1997. 401

4. M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Machine Learning: Proceedings of the Fourteenth International Conference*, Nashville, TN, 1997. 401
5. D. Fisher, L. Xu, J. Carnes, Y. Reich, S. Fenves, J. Chen, R. Shiavi, G. Biswas, and J. Weinberg. Applying ai clustering to engineering tasks. *IEEE Expert*, 8:51–60, 1993. 400
6. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987. 393
7. D. H. Fisher. Iterative optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, 4:147–179, 1996. 402
8. J. H. Gennari. Concept formation and attention. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 724–728, Irvine, CA, 1991. Lawrence Erlbaum Associates. 392, 398, 400
9. R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997. 394, 397
10. J. L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281–328, 1983. 400
11. P. Langley. *Elements of machine learning*. Morgan Kaufmann, San Francisco, CA, 1995. 393
12. M. Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2:103–138, 1987. 400
13. L. Talavera. Dependency-based feature selection for symbolic clustering. *Intelligent Data Analysis*. To appear. 400, 401
14. L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia, 1999. Morgan Kaufmann. 395, 401
15. L. Talavera. Feature selection as retrospective pruning in hierarchical clustering. In *Third International Symposium on Intelligent Data Analysis, IDA 99*, volume 1642 of *Lecture Notes in Computer Science*, Amsterdam, The Netherlands, 1999. Springer Verlag. 395, 400, 401
16. J. J. Furtado Vasco. Determining property relevance in concept formation by computing correlation between properties. In *Proceedings of the Tenth European Conference on Machine Learning, ECML98*, volume 1398 of *Lecture Notes in Artificial Intelligence*, pages 310–315, Chemnitz, Germany, 1998. Springer Verlag. 401

On the Boosting Pruning Problem

Christino Tamon¹ and Jie Xiang²

¹ Clarkson University, Potsdam, NY 13699-5815, USA
tino@clarkson.edu

² BCL Computers, Inc., USA
jiex@bcl-computers.com

Abstract. Boosting is a powerful method for improving the predictive accuracy of classifiers. The ADABOOST algorithm of Freund and Schapire has been successfully applied to many domains [2,10,12] and the combination of ADABOOST with the C4.5 decision tree algorithm has been called the *best off-the-shelf* learning algorithm in practice. Unfortunately, in some applications, the number of decision trees required by ADABOOST to achieve a reasonable accuracy is enormously large and hence is very space consuming. This problem was first studied by Margineantu and Dietterich [7], where they proposed an empirical method called Kappa pruning to *prune* the boosting ensemble of decision trees. The Kappa method did this without sacrificing too much accuracy. In this work-in-progress we propose a potential improvement to the Kappa pruning method and also study the *boosting pruning* problem from a theoretical perspective. We point out that the boosting pruning problem is intractable even to approximate. Finally, we suggest a *margin*-based theoretical heuristic for this problem.

1 Introduction

Boosting is a method for combining classifiers to improve prediction accuracy. The idea of boosting is to alter repeatedly the distribution on the training data so that the learning algorithm is forced to focus on harder examples. A boosting algorithm called ADABOOST (Freund and Schapire [1]) has been extensively studied both theoretically and empirically. The algorithm is proven to be theoretically sound and shown to be empirically appealing because of its simplicity and superior performance in many domains.

Many research have focused on boosting decision trees, notably using Quinlan's C4.5 [9] as the tree induction algorithm. The ADABOOST-C4.5 combination has been called the best off-the-shelf learning algorithm in practice because of its superior performance on many benchmark datasets [10,2]. Despite its good performance, Margineantu and Dietterich [7] observed that, in some domains, boosting needs to combine a large number of trees to lower the prediction error. More specifically, they observed that in the **letter** dataset, ADABOOST requires about 200 iterations of C4.5 to achieve a reasonable accuracy. So the final classifier is a weighted ensemble of about 200 decision trees (each being a nontrivially large tree). They asked if all 200 decision trees are necessary: is there a way of

pruning some of these trees from the final ensemble without deteriorating the performance.

Margineantu and Dietterich then proposed an interesting method of pruning the boosting ensemble using a statistic called the Kappa measure (see [7] and the references therein). Their heuristic idea is based on the assumption that boosting works by building *diversity* in its ensemble. The Kappa statistic is a measure of agreement between two classifiers. They create their pruned ensemble by greedily selecting pairs of decision trees with very diverse behavior until they reached the required pruning rate. Up to certain rates of pruning, the performance of the pruned ensemble is quite close to the original ensemble. In this paper we propose a slight modification to the Kappa method called *weight shifting*. Viewing the pruning process as a *clustering*-like process, we shifted the voting weights of pruned trees onto its unpruned neighbors. We conducted some preliminary experiments and observed some encouraging although mixed results.

Next we study some theoretical aspects of the boosting pruning problem. We show that the boosting pruning problem is NP-complete and is even hard to approximate. Then we propose a pruning scheme that is *margin*-based. Recent work by Schapire et al. [11] has shown that boosting achieves good generalization error by maximizing the minimum margin on the training sample. We suggest a theoretical heuristic derived using tools from the area of approximation algorithms, where a trade-off between the margin and the size of the pruned boosting ensemble is made explicit.

2 Boosting Decision Trees

Quinlan's C4.5 algorithm is a well-studied method for inducing decision trees from data (see [9]). It is a top-down method that continually splits the training data using the best attribute under an entropic measure. Several works have studied boosting decision trees by combining ADABOOST with C4.5 (including [10,2]). We follow Quinlan's boosting experiments [10] by making use of C4.5's ability to assign fractional weights to data items. This will be important in how we do boosting.

The ADABOOST algorithm (Freund and Schapire [1]) works by repeatedly calling the weak learning algorithm (in this case C4.5) on a newly reweighted training data. The reweightings are done so as to focus the weak learner's attention to examples where mistakes are still being made. This cycle repeats until all training data are correctly classified.

We introduce some notation before we describe the ADABOOST algorithm formally. Let X be the example domain and let Y be the label domain. A *labeled sample* S is a sequence of pairs $(x, y) \in X \times Y$. We assume that S is drawn according to some fixed but unknown distribution D over X and that the labels satisfy $y = f(x)$, for some unknown target function f . The *training error* of a function h with respect to sample S is defined as $\epsilon_S(h) = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{I}[h(x) \neq y]$, where $\mathbb{I}[\pi]$ is 1 if the statement π is true and 0 otherwise. The *generalization error* of a function h is defined as $\epsilon_D(h) = \Pr_{(x,y) \sim D}[h(x) \neq y]$. The ADABOOST

Input: A training sample $S = \{(x_i, y_i) \mid 1 \leq i \leq m\}$, where $x_i \in X$ and $y_i \in Y$.

Output: A classifier $H : X \rightarrow Y$ with small training error on S .

1. $D_1(x_i) = 1/m$, for all $1 \leq i \leq m$.
2. **for** $t = 1, 2, \dots, T$ **do**
3. call C4.5 on input S and D_t
4. get weak hypothesis $h_t : X \rightarrow Y$
5. $\epsilon_t = \sum_{i=1}^m D_t(i) \llbracket h_t(x_i) \neq y_i \rrbracket$
6. **if** $\epsilon_t \geq 0.5$ **then** set $T = t - 1$ and abort loop.
7. $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
8. reweight distribution: $D_{t+1}(x_i) = D_t(x_i) \beta_t^{\llbracket h_t(x_i) = y_i \rrbracket} / Z_t$, where Z_t is a normalization constant.
9. **end for**
10. output $H(x) = \arg \max_{y \in Y} \sum_{t: h_t(x)=y} \ln(1/\beta_t)$.

Fig. 1. The ADABOOST.C4.5 algorithm

algorithm is shown in Figure 1. In this paper we adopt Quinlan's strategy of boosting by *reweighting* [10] (instead of *resampling* [2]).

3 Kappa Pruning

The boosting pruning heuristic of Margineantu and Dietterich [7] proceeds as follows. First we define the Kappa measure between two classifiers h_i and h_j , where $h_i, h_j : X \rightarrow Y$. Consider the following $|Y| \times |Y|$ contingency table or matrix M : for $a, b \in Y$, define $M_{a,b}$ to be the fraction of examples $x \in S$ where $h_i(x) = a$ and $h_j(x) = b$. Let $\Theta_1 = \sum_{a \in Y} M_{a,a}$ and $\Theta_2 = \sum_{a \in Y} M_{a,*} M_{*,a}$, where $M_{a,*} = \sum_{b \in Y} M_{a,b}$ and $M_{*,a} = \sum_{b \in Y} M_{b,a}$. The parameter Θ_1 is a measure of $\Pr_S[h_i = h_j]$ and Θ_2 is a measure of $\sum_{a \in Y} \Pr_S[h_i = a] \Pr_S[h_j = a]$. Then the Kappa measure of agreement between h_i and h_j is defined as $\kappa(h_i, h_j) = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}$. A value of $\kappa = 0$ implies that $\Theta_1 = \Theta_2$ and the two classifiers are considered to be different (or independent). A value of $\kappa = 1$ implies that $\Theta_1 = 1$ which means total agreement between the two classifiers. It is possible for κ to be negative although it was noted that this rarely occurs [7].

Using this distance measure, the Kappa pruning method [7] proceeds as follows. It computes all pairwise Kappa distances between the decision trees in the boosting ensemble. After sorting these distance values, the algorithm greedily includes the pairs of hypotheses that correspond to small Kappa distances. This continues until a certain pruning rate is achieved. The resulting boosting ensemble consists of all decision trees included from the greedy selection stage. In effect, the Kappa pruning algorithm sets to zero all the voting weight of the pruned decision trees (the α 's in the final hypothesis of ADABOOST).

3.1 Weight Shifting

Here we propose an alternative heuristic for performing Kappa pruning based on a *weight shifting* strategy. While Kappa pruning sets to zero the weights of all pruned decision trees in the boosting ensemble, we propose the following variant: transfer the voting weight of a pruned decision tree to the unpruned ones. This strategy views the pruning process as a clustering process whereby a collection of diverse classifiers are selected to represent the original ensemble. We adopt the following *soft assignment* method of shifting the weight of a pruned hypothesis onto the collection of unpruned ones: each unpruned hypothesis receives a fraction of weight proportional to its *similarity* to the pruned hypothesis. So, in the soft assignment, each pruned classifier computes the set of distances from itself to the collection of unpruned classifiers. The pruned classifier then distributes its voting weight using the distribution of distances (after normalization). More weight is given to classifiers that are *closer* (similar or $\kappa \sim 1$) to the pruned classifier.

We conjecture that the weight shifting process helps produce a more faithful final ensemble, especially when the pruning rate is high. We conducted some preliminary experiments on the effectiveness of Kappa pruning with weight shifting using soft assignment. We report our findings in the next section.

3.2 Experiments

The real-world datasets that we used in our experiments were obtained from the University of California at Irvine (UCI) Machine Learning Repository [8]. Some information about the datasets are given in Table 1.

Table 1. UCI datasets

name	examples		classes	attributes		
	train	test		disc	cont	missing
auto	205	0	7	11	15	yes
crx	490	200	2	9	6	yes
letter	20000	0	26	0	16	none
monk1	124	432	2	6	0	none
monk2	169	432	2	6	0	none
promoter	106	0	2	57	0	none
soybean	316	0	19	35	0	yes
waveform	5000	0	3	0	21	no

In Table 2 we report a 10-fold cross validation estimate of the generalization error for plain C4.5, ADABOOST and C4.5 with no pruning, and ADABOOST and C4.5 with the two pruning options. We have used the conservative choice of

using 30 boosting iterations¹. Plots of these comparisons are omitted from this abstract due to lack of space.

The basic Kappa pruning algorithm is denoted **kp** and the weight-shifted version is denoted **ws**. The pruning rates that we used are 0.9, 0.8, 0.7, 0.6, 0.5. Here a pruning rate of α means that we eliminate at least $1 - \alpha$ fraction of the ensemble. So a pruning rate of 0.9 eliminates 10% of the ensemble.

We focused on some UCI datasets where boosting (with 30 rounds) showed a definite improvement upon C4.5 alone. The datasets we used are **auto**, **crx**, **letter**, **monk1**, **monk2**, **promoter**, **soybean**, and **waveform**. We will seek those pruning rates where error rates are still lower than the case without pruning. Our future plans include making comparisons between ensembles of the same size (obtained with and without pruning).

Table 2. 10x-val comparison of C4.5, ADABOOST, Kappa, and weight shifting

name	C4.5	AdaBoost	.9		.8		.7		.6		.5	
	pruned	T=30	kp	ws	kp	ws	kp	ws	kp	ws	kp	ws
auto	22.4	17.4	18.4	18.4	19.4	19.4	19.4	18.9	21.9	20.9	22.4	22.4
crx	16.5	13.5	13.0	13.2	13.3	13.6	13.2	13.0	12.9	12.3	13.6	13.6
letter	12.22	4.43	4.5	4.51	4.69	4.66	5.0	4.96	5.45	5.47	5.91	5.86
monk1	3.8	0.0	25.5	24.9	25.5	25.5	25.5	25.5	25.5	25.5	26.0	25.5
monk2	33.6	32.1	31.6	31.6	32.6	32.6	32.8	32.4	31.9	32.3	32.4	31.4
promoter	25.0	21.0	21.0	21.0	22.0	22.0	23.0	24.0	28.0	27.0	31.0	27.0
soybean	7.2	5.46	5.7	5.9	5.7	5.7	5.6	5.6	5.9	5.9	6.5	6.6
waveform	25.28	19.50	19.50	19.50	19.64	19.62	20.3	20.26	20.54	20.42	21.82	21.74

The comparison on the datasets **auto**, **crx**, **letter**, and **waveform** showed that weight shifting could help improve the Kappa method in certain pruning rates (mainly for aggressive rates). However, the performance of both methods on **letter** is too similar and hence the improvement is perhaps too negligible. We would like to see if an increased number of boosting iterations might improve this situation.

Furthermore, pruning seemed to cause erratic behavior in the **monk** datasets. We are not sure if this is caused by the special form of the **monk** datasets or a subtle error in our experiment. In **monk1**, pruning caused a marked increase in the error rate. In **monk2**, the improvement of weight shifting is a bit erratic after pruning showed an encouraging promise at low pruning rates. Both methods of pruning also do not seem to work well on **promoter** and **soybean** (although in the former case, weight shifting was better than Kappa on high pruning rates).

¹ We plan to run further experiments using higher number of boosting iterations (e.g., Margineantu and Dietterich [7] used 50 iterations in their experiments).

4 The Abstract Boosting Pruning Problem

In this section we turn to theoretical considerations of the boosting pruning problem. A *boosting ensemble* H is a collection of hypotheses $h : X \rightarrow \{-1, +1\}$ from a known class C of classifiers (for instance, decision trees) where each h has an associated weight $\alpha \in \mathbb{R}$. So let $H = \{\langle \alpha_i, h_i \rangle \mid 1 \leq i \leq T\}$ be a boosting ensemble of size T . We identify the ensemble H with the function $H(x) = \text{sgn}(\sum_{i=1}^m \alpha_i h_i(x))$, where $\text{sgn}(x) = +1$ if $x \geq 0$ and $\text{sgn}(x) = -1$ otherwise. We also identify any subset A of H with the function $H_A(x) = \text{sgn}(\sum_{i \in A} \alpha_i h_i(x))$.

We will first make the assumption that minimizing training error leads to the minimization of generalization error (or true error). Under this assumption, we formalize the boosting pruning problem as follows. Assume that the example domain X and the label domain Y are fixed.

ENSEMBLE PRUNING

input: A boosting ensemble $H = \{\langle \alpha_i, h_i \rangle \mid 1 \leq i \leq T\}$, where, for each $i = 1, 2, \dots, T$, $\alpha_i \in \mathbb{R}$ and $h_i : X \rightarrow \{-1, +1\}$, and a sample set $S = \{\langle x_i, y_i \rangle \in X \times Y \mid 1 \leq i \leq m\}$.

output: A subset A of H minimizing the training error of $H_A(x)$ on S .

For simplicity, we consider an associated problem called MATRIX COVER. Associate with each boosting set of T hypotheses and each sample set of m points, a matrix M of size $T \times m$ where $M_{i,j} = -1$ if $h_i(x_j) = y_j$, and $M_{i,j} = +1$ if $h_i(x_j) \neq y_j$. Assume that M satisfies the *positive column-sum* property, i.e., for all $j \in [m]$, $\sum_{i=1}^T M_{i,j} > 0$. This last property means that the boosting ensemble associated with the T rows of M is perfect on the m training points. The question now is to find the smallest subset of the rows of M so that the positive column-sum property is maintained.

MATRIX COVER

input: An integral matrix M of size $T \times m$ such that, for all $j \in [m]$, $\sum_{i=1}^T M_{i,j} > 0$.

output: A minimal subset A of the rows of M such that, i.e., for all $j \in [m]$, $\sum_{i \in A} M_{i,j} > 0$.

Claim. MATRIX COVER is NP-complete.

Proof. Reduction from SET COVER (see [3]). □

Given the NP-completeness of MATRIX COVER, it is natural to ask for the next best solution: an approximation algorithm. For $\alpha > 0$, we say that an algorithm is an approximation algorithm for MATRIX COVER if for any input M to MATRIX COVER it outputs a subset B so that $|B| \leq \alpha \text{OPT}(M)$, where $\text{OPT}(M)$ is the value of the optimal solution. A very strong hardness result can be proven about approximating MATRIX COVER.

Claim. MATRIX COVER is unapproximable to within n^ϵ , $\epsilon > 0$, unless $P = NP$.

Proof. Reduction using the MINIMUM PB 0-1 PROGRAMMING (see [6]). □

4.1 A Margin-Based Heuristic

Although MATRIX COVER is highly intractable to approximate, we suggest in this section a theoretical heuristic for the boosting pruning problem. Note that MATRIX COVER imposes the condition that the resulting final hypothesis must have zero error on the training data. Implicitly, the performance of the boosting hypothesis is measured in terms of the number of mistakes. A recent work by Schapire et al. [11] has shown that an alternative measure called *margin* is a better indicator of the generalization error (or true error) of the boosting hypothesis.

Let us assume now that we have a binary prediction problem, where $Y = \{-1, +1\}$, but that each weak hypothesis can use confidence-rated predictions (as in Schapire and Singer's work [12]), i.e., $h : X \rightarrow \mathbb{R}$. Here the sign of h reflects its prediction while its magnitude reflects its confidence in that prediction. Note that the final boosting hypothesis (before thresholding) is $H(x) = \sum_i \alpha_i h_i(x)$. The margin of H on the example $(x, y) \in X \times \{-1, +1\}$ is defined as $m(x) = yH(x)$. A positive margin on an example means that H predicts correctly on that example and the magnitude of the margin reflects the magnitude of its correctness. Schapire et al [11] proved that a hypothesis with large positive margin on all training examples is a hypothesis with low generalization error.

Using margin theory, we suggest a different heuristic to ENSEMBLE PRUNING. In defining the matrix in our MATRIX COVER instance, let $M_{i,j} = y_j h_i(x_j)$ be the margin of the i -th hypothesis h_i on the j -th example (x_j, y_j) . Now the j -th column-sum of M is the margin of H on the j -th example (x_j, y_j) .

MATRIX COVER

input: A positive constant $\theta > 0$ and a real-valued matrix M of size $T \times m$ such that, for all $j \in [m]$, $\sum_{i=1}^T M_{i,j} > \theta$.

output: A minimal subset A of the rows of M such that, for all $j \in [m]$, $\sum_{i \in A} M_{i,j} > \theta$.

We now attempt to design a heuristic for this new MATRIX COVER problem. Borrowing some ideas from the approximation algorithms literature [4], here is a well-known approach using mathematical programming: (a) express the problem as an integer program; (b) relax the integer program as a linear program and solve it using a polynomial-time algorithm; (c) (randomly) round the linear programming solution to get an integral solution. The integer program (IP) associated with MATRIX COVER is given as: MINIMIZE $\sum_{i=1}^T z_i$ SUBJECT TO $\sum_{i=1}^T m_{i,j} z_i \geq \theta$, for $j \in [m]$, and $z_i \in \{0, 1\}$, for $i \in [T]$. The linear programming relaxation (LP) is obtained by letting $z_i \in [0, 1]$, for $i \in [T]$.

Letting $Z \in [0, 1]^T$ be the optimal LP solution and $Z^* \in \{0, 1\}^T$ be the optimal IP solution. Denote the value of the optimal solutions by $z = \sum_i Z_i$ and $z^* = \sum_i Z_i^*$, respectively. Note that z is a lower bound to z^* . We apply a method called *randomized rounding* to obtain an integral solution from the LP solution. Given Z , let $\hat{Z}$ be the integral solution as follows: for each i , let $\hat{Z}_i = 1$, with probability Z_i , and $\hat{Z}_i = 0$, with probability $1 - Z_i$. Note that the expected value of this integral solution equals to the value of the LP solution: $E[\hat{Z}] =$

$E[\sum_i \hat{Z}_i] = \sum_i Z_i = Z$. Moreover, the constraints are satisfied *on average*: for all j , $E[\sum_i m_{i,j} \hat{Z}_i] > \theta$. Using standard large deviation inequalities [5], we claim that $\hat{Z}$ is concentrated near Z and that the constraints are somewhat satisfied. More specifically, $\Pr[|\sum_i \hat{Z}_i - \sum_i Z_i| \geq c\sqrt{T}] \leq 1/4$, whenever $c > 0.6$, and $\Pr[(\exists j)(\sum_i m_{i,j} \hat{Z}_i \leq \alpha\theta)] \leq 1/4$, by a judicious choice of dependence between α and θ . Note that α represents a *slackness* parameter on the constraints whereas θ is related to the margin of the boosting ensemble.

So with non-negligible probability, a semi-feasible solution is obtained and $\hat{Z}$ will be within an additive factor of $\mathcal{O}(\sqrt{T})$ from the optimal LP solution. This approach allows us to trade optimality (smallness of the boosting ensemble) with feasibility (goodness of its margin).

5 Conclusion and Future Work

In this paper we revisited the boosting pruning problem [7]. We proposed a minor modification of the powerful Kappa pruning method and reported some preliminary observations of our *weight-shifting* variant. We plan to conduct further and more extensive experiments on this problem. In addition, we have also considered the boosting pruning problem theoretically, proving that the problem is highly intractable, even to approximate. Using ideas from approximation algorithms, we proposed a theoretical heuristic. This heuristic differs from the Kappa method in that it is driven by margin considerations (instead of discrete error). This approach allows one to trade the *size* of the boosting ensemble and the *margin* of the ensemble. We plan to carry out experimental work on this margin-based algorithm.

References

1. Y. Freund and R.E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *J. Comp. System Sciences*, 55(1):119-139, 1997. 404, 405
2. Y. Freund and R.E. Schapire. Experiments with a New Boosting Algorithm. *Proc. 13th Int. Conf. on Machine Learning*, 148-156, 1996. 404, 405, 406
3. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979. 409
4. D. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997. 410
5. W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *J. American Stat. Assoc.*, 58:13-30, 1963. 411
6. V. Kann. Polynomially bounded minimization problems that are hard to approximate. *Nordic Journal of Computing*, 1:317-331, 1994. 409
7. D. Margineantu and T.G. Dietterich. Pruning Adaptive Boosting. *Proc. 14th Int. Conf. Machine Learning*, 211-218, 1997. 404, 405, 406, 408, 411
8. C.J. Merz and P.M. Murphy. UCI Repository of Machine Learning Databases. Tech. Report, U.C. Irvine, CA. 407

9. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. [404](#), [405](#)
10. J.R. Quinlan. Bagging, Boosting, and C4.5. *Proc. 13th Nat. Conf. Artificial Intelligence*, 725-730, 1996. [404](#), [405](#), [406](#)
11. R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the Margin: a new explanation of the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651-1686, 1998. [405](#), [410](#)
12. R.E. Schapire and Y. Singer. Improved Boosting Algorithms using Confidence-rated Predictions. *Proc. 11th Ann. Conf. Comp. Learning Theory*, 80-91, 1998. [404](#), [410](#)

An Empirical Study of MetaCost Using Boosting Algorithms

Kai Ming Ting

School of Computing and Mathematics, Deakin University,
Vic 3168, Australia
kmting@deakin.edu.au

Abstract. MetaCost is a recently proposed procedure that converts an error-based learning algorithm into a cost-sensitive algorithm. This paper investigates two important issues centered on the procedure which were ignored in the paper proposing MetaCost. First, no comparison was made between MetaCost’s final model and the internal cost-sensitive classifier on which MetaCost depends. It is credible that the internal cost-sensitive classifier may outperform the final model without the additional computation required to derive the final model. Second, MetaCost assumes its internal cost-sensitive classifier is obtained by applying a minimum expected cost criterion. It is unclear whether violation of the assumption has an impact on MetaCost’s performance. We study these issues using two boosting procedures, and compare with the performance of the original form of MetaCost which employs bagging.

1 Introduction

MetaCost (Domingos, 1999) is a recently proposed method for making an arbitrary classifier cost-sensitive. The method has an interesting design that uses a “meta-learning” procedure to relabel the classes of the training examples, and then employs the modified training set to produce a final model. MetaCost has been shown to outperform two forms of stratification which change the frequency of classes in the training set in proportion to their cost.

MetaCost depends on an internal cost-sensitive classifier in order to relabel classes of training examples. But the study by Domingos (1999) made no comparison between MetaCost’s final model and the internal cost-sensitive classifier on which MetaCost depends. This comparison is worth making as it is credible that the internal cost-sensitive classifier may outperform the final model without the additional computation required to derive the final model.

A simple method of converting an error-based classifier to a cost-sensitive classifier is to apply an additional minimum expected cost criterion (Michie, Spiegelhalter & Taylor, 1994). Because this criterion only needs to be applied during classification, this method has the advantage of re-using the same learned model when the cost changes. MetaCost assumes its internal cost-sensitive classifier is obtained this way, and thus inherits the advantage. However, previous study (Ting & Zheng, 1998) has shown that this simple approach is not the

best method to minimize cost. It is therefore unclear whether the performance of MetaCost will be affected if the assumption is violated.

Boosting (Quinlan, 1996; Schapire, Freund, Bartlett, & Lee, 1997) can be effective and can be better than bagging (Bauer & Kohavi, 1999) in minimizing errors. MetaCost (Domingos, 1999) uses bagging internally. Using a boosting procedure in MetaCost may improve MetaCost’s performance. This is the reason why we choose to use boosting procedures in MetaCost in this paper.

This paper has two aims. First, to make a direct comparison between MetaCost and the internal cost-sensitive classifier on which MetaCost relies. Second, to investigate whether a violation of MetaCost’s assumption has an impact on its performance. In the next section, we describe MetaCost and boosting. In Section 3, we present the empirical comparison in four separate subsections. A discussion of the results is given in Section 4, followed by conclusions in the last section.

2 MetaCost and Boosting Procedures

2.1 MetaCost

MetaCost (Domingos, 1999) is based on the Bayes optimal prediction that minimizes the expected cost $R(j|x)$ (Michie, Spiegelhalter & Taylor, 1994):

$$R(j|x) = \sum_i^I P(i|x) \text{cost}(i, j),$$

where $P(i|x)$ is the probability of class i given example x and $\text{cost}(i, j)$ is the cost of misclassifying a class i example as class j .

The Bayes optimal prediction rule implies a partition of the example space into I regions, such that class i is the minimum expected cost prediction in region i . If misclassifying class i becomes more expensive relative to misclassifying others, then parts of the previously non-class i regions shall be re-assigned as region i since it is now the minimum expected cost prediction.

The MetaCost procedure estimates class probabilities using bagging (Bauer & Kohavi, 1999), and relabels the training examples with their minimum expected cost classes, and finally relearns a model using the modified training set.

We interpret the process of estimating class probabilities and applying the Bayes optimal prediction rule as constructing an internal cost-sensitive classifier for MetaCost. With this interpretation, we formalize the MetaCost procedure as a three-step process depicted in Figure 1.

The procedure begins to learn an internal cost-sensitive model by applying a cost-sensitive procedure which employs a base learning algorithm. Then, for each of the training examples, assign the predicted class of the internal cost-sensitive model to be the class of the training example. Finally, learn the final model by applying the same base learning algorithm to the modified training set.

Given $\mathcal{T}$: a training set containing N examples (x_n, y_n) where x_n is a vector of attribute-values and y_n is the class label, L : a base learning algorithm, $\mathcal{C}$: a cost-sensitive learning procedure, and $cost$: a cost matrix.

MetaCost($\mathcal{T}, L, \mathcal{C}, cost$)

(i) Learn an internal cost-sensitive model by applying $\mathcal{C}$:

$$H^* = \mathcal{C}(\mathcal{T}, L, cost).$$

(ii) Modify the class of each example in $\mathcal{T}$ from the prediction of H^* :

$$y_n = H^*(x_n).$$

(iii) Construct a model H by applying L to $\mathcal{T}$.

Output: H .

Fig. 1. The MetaCost procedure

The performance of MetaCost relies on the internal cost-sensitive procedure in the first step. Getting a good performing internal model in the first step is crucial to getting a good performing final model in the third step.

MetaCost in its original form (Domingos, 1999) assumes that its internal cost-sensitive procedure is obtained by applying the Bayes optimal prediction rule to an existing error-based procedure. Thus, the cost-sensitive procedure $\mathcal{C}$ consists of first getting the class probability from a model h defined as follows. Choose an error-based procedure $\mathcal{E}$ which employs a training set $\mathcal{T}$ and a base learning algorithm L to induce the model $h = \mathcal{E}(\mathcal{T}, L)$, without reference to cost. Given a new example x , h produces a class probability for each class:

$$P(i|x) = h(x), \text{ for each class } i.$$

Then, apply the Bayes rule or minimum expected cost criterion:

$$H^*(x) = \arg \min_j \sum_i P(i|x) cost(i, j).$$

However, a cost-sensitive procedure, that takes cost directly into consideration in the training process, is another option. In this paper, both types of cost-sensitive procedures are used to evaluate their effects on MetaCost. We use the error-minimizing boosting algorithm AdaBoost in place of $\mathcal{E}$, and a cost-sensitive version of the boosting procedure in place of $\mathcal{C}$. AdaBoost is often effective in minimizing errors. Both procedures are described in the next section.

2.2 Boosting Procedures

AdaBoost induces multiple individual classifiers in sequential trials, and a weight is assigned to each training example. At the end of each trial, the vector of

Given $\mathcal{T}$: a training set containing N examples (x_n, y_n) , L : a base learning algorithm, and $cost$: a cost matrix.

AdaBoost($\mathcal{T}, L, cost, K$)

Initialization: all instance weights $w_1(n) = 1$.

For $k = 1, \dots, K$

- (i) Learn a model h_k by applying L to $\mathcal{T}$ under the weight distribution w_k . Let $h_k(x)$ denotes the predicted class, and $\hat{h}_k^i(x) \in [0, 1]$ denotes the confidence level of the prediction for class i .
- (ii) $\mathcal{T}$ is classified using h_k . The error of this model, ϵ_k , is defined as

$$\epsilon_k = \frac{1}{N} \sum_{x_n \in \mathcal{T}: h_k(x_n) \neq y_n} w_k(n). \quad (1)$$

If $\epsilon_k \geq 0.5$ or $\epsilon_k = 0$, then all $w_k(n)$ is reset using bootstrap sampling, i.e., $w_k(n)$ is set zero and incremented one unit every time instance n is selected in the sampling with replacement process to select N samples, and the process continues from step (i).

- (iii) The instance weight $w_{(k+1)}$ for the next trial is created from w_k as follows:

$$w_{(k+1)}(n) = \begin{cases} w_k(n)F_k & \text{if } h_k(x_n) \neq y_n \\ w_k(n)/F_k & \text{otherwise} \end{cases} \quad (2)$$

where, $F_k = \sqrt{\frac{(1-\epsilon_k)}{\epsilon_k}}$.

Output:

$$P(i|x) \propto \sum_k \log(F_k) \hat{h}_k^i(x). \quad (3)$$

$$H^*(x) = \arg \min_j \sum_i \sum_k \log(F_k) \hat{h}_k^i(x) cost(i, j). \quad (4)$$

Fig. 2. The AdaBoost procedure that uses the minimum expected cost criterion

weights is adjusted to reflect the importance of each training example for the next induction trial. This adjustment effectively increases the weights of misclassified examples and decreases the weights of the correctly classified examples. These weights cause the learner to concentrate on different examples in each trial and so lead to different classifiers. Finally, the individual classifiers are combined to form a composite classifier. The AdaBoost procedure is shown Figure 2. Note that the weight adjustment formula in Equation (2) is from a version of AdaBoost described in Schapire, Freund, Bartlett, & Lee (1997). To use the simple method to convert the error-based AdaBoost procedure to a cost-sensitive version, a minimum expected cost criterion is applied as shown in Equation (4). This is made possible by assuming the weighted votes for each class is proportional to the class probability, as shown in Equation (3).

Initialization: The weight of every class j instance at $k = 1$ is initialized as follows.

$$w_1(n) = w^j = C^j \frac{N}{\sum_i C^i N^i},$$

where $C^i = \sum_j \text{cost}(i, j)$, is the cost of misclassifying a class i instance, and N^i is the number of class i instances; $N/\sum_i C^i N^i$ is the normalizing term such that $\sum_j w^j C^j = N$

The weight update rule:

$$w_{(k+1)}(n) = \begin{cases} w_k(n) \text{cost}(y_n, h_k(x_n)) & \text{if } h_k(x_n) \neq y_n \\ w_k(n) & \text{otherwise} \end{cases}$$

$w_{(k+1)}(n)$ needs to be normalized so that the sum of all $w_{(k+1)}(n)$ equals to N .

Fig. 3. Weight initialization and weight update rule for cost-sensitive boosting CSB

A cost-sensitive boosting procedure that takes cost directly into consideration in the training process can be obtained by replacing the weight update rule of Equation (2) in the AdaBoost procedure to a new rule depicted in Figure 3. The rule increases the weight by a factor of the misclassification cost if an example is misclassified, otherwise the weight from the previous trial is retained. Note that this is an improved version of the one proposed by Ting & Zheng (1998). We called this cost-sensitive boosting procedure, CSB. Although the changes also include a weight initialization process, also shown in Figure 3, the weight update rule has the dominating influence on the performance of CSB.

We denote MetaCost_A as the algorithm that uses AdaBoost in MetaCost, and MetaCost_CSB uses CSB. The base learning algorithm, L , we used to conduct our experiments is the well-known decision tree learning algorithm, C4.5 (Quinlan, 1993). Only the default settings of C4.5 are used. The parameter K controlling the number of classifiers generated in both boosting procedures is set at 10 for all experiments described in the following section, unless stated otherwise. When bagging is used, we also employ 10 classifiers in each run.

3 Experiments

In this section, we empirically evaluate the performance of MetaCost's final model H and its internal classifier H^* produced by boosting and bagging. Twenty-four natural datasets, which consists of fourteen two-class datasets and ten multi-class datasets, from the UCI machine learning repository (Blake, Keogh & Merz, 1998) are used in the experiments. The description of this test suite is shown in Table 1.

Table 1. Description of datasets

Dataset	Size	No. of Classes	No. of Attributes	
			Numeric	Nominal
breast cancer (Wisconsin)	699	2	9	0
liver disorder	345	2	6	0
credit screening (Australian)	690	2	6	9
echocardiogram	131	2	6	1
solar flare	1389	2	0	10
heart disease (Cleveland)	303	2	13	0
hepatitis prognosis	155	2	6	13
horse colic	368	2	7	15
house-voting 84	435	2	0	16
hypothyroid diagnosis	3168	2	7	18
king-rook-vs-king-pawn	3169	2	0	36
pima indians diabetes	768	2	8	0
sonar classification	208	2	60	0
tic-tac-toe end game	958	2	0	9
abalone	4177	3	7	1
annealing process	898	6	6	32
glass identification	214	6	9	0
lymphography	148	4	0	18
nettalk-stress	5438	5	0	7
nursery	12960	5	0	8
satellite	6435	6	36	0
soybean large	683	19	0	35
splice junction	3177	3	0	60
wine recognition	178	3	13	0

For each of the two-class datasets, we report the sum of three averages of two 10-fold cross-validations using three fixed cost ratios. Costs are assigned such that misclassifying a minority class example costs more than misclassifying a majority class example. Suppose i is the majority class, that is, $P(i) > P(j)$, then $cost(i, j) = 1$ and $cost(j, i) = r$. The fixed cost ratios used to obtain the three averages are $r = 2, 5$, and 10 . This means that misclassifying a minority class example is r times more costly than misclassifying a majority class example. In this way, we simulate the situation often found in practice where it is most important to correctly classify the rare classes. In this paper, all correct classifications are assumed to have no cost, that is, for all i , $cost(i, i) = 0$.

For each of the multi-class datasets, we report the average of two 10-fold cross-validations. We emulate the similar condition as in the two-class datasets. Each cost matrix is randomly initialized at the beginning of each run as follows. For each pair of i, j and $i \neq j$ and $P(i) > P(j)$, we assign $cost(i, j) = 1$ and $cost(j, i) = r$, where r is a randomly selected integer from 2 to 10.

We use two measures to evaluate the performance of the algorithms employed for cost-sensitive classification. The first measure is the *total cost of misclassifications* made by a classifier on a test set (i.e., $\sum_m cost(actual(m), predicted(m))$). The second measure is the *number of high cost errors*. It is the number of misclassifications associated with cost higher than 1 made by a classifier on a test

Table 2. Comparison of MetaCost, AdaBoost and CSB (Cost)

Dataset	MetaC_A	AdaB vs MetaC_A	MetaC_CSB	CSB vs MetaC_CSB	CSB vs AdaB	MetaC_CSB vs MetaC_A
	cost	cost ratio	cost	cost ratio	cost ratio	cost ratio
breast cancer	18.50	.85	20.35	.74	.96	1.10
liver disorder	57.75	.99	57.90	.99	1.00	1.00
credit	64.25	1.00	61.10	.97	.92	.95
echocardio.	23.15	.97	24.10	1.01	1.08	1.04
solar flare	215.15	1.02	226.05	1.02	1.05	1.05
heart (C)	39.30	.96	39.70	.95	.99	1.01
hepatitis	25.15	.90	21.95	.95	.91	.87
horse colic	53.40	.95	50.65	.99	.99	.95
house-voting	13.45	1.14	11.30	1.02	.75	.84
hypothyroid	27.60	1.11	25.45	1.01	.84	.92
kr-vs-kp	54.45	1.15	21.95	.86	.30	.40
pima	110.70	.99	112.35	.96	.98	1.01
sonar	26.40	.99	32.25	.76	.94	1.22
tic-tac-toe	114.95	.73	120.05	.68	.97	1.04
Mean Geomean	60.30	.98	58.94	.91	.87	.93
abalone	238.05	.99	231.90	1.01	.99	.97
anneal	18.45	.82	14.45	.71	.68	.78
glass	16.05	.85	15.05	.84	.93	.94
lymphography	8.40	.77	8.85	.76	1.03	1.05
nettalk-stress	257.75	.78	252.95	.79	.99	.98
nursery	149.65	.90	80.00	.86	.51	.53
satellite	201.65	.86	218.10	.72	.90	1.08
soybean	18.85	.72	13.70	.68	.69	.73
splice junction	74.10	.81	38.10	1.01	.64	.51
wine	2.65	1.70	3.60	.72	.58	1.36
Mean Geomean	98.56	0.89	87.67	.80	.80	.86
w/t/l p of w/l	18/1/5 .005		18/0/6 .0113		20/1/3 .0002	13/1/10 .3388

set. Both measures are presented because a classifier that minimizes the cost does not necessarily minimize the number of high cost errors. Depending on the need of the user, a good cost-sensitive classifier should have either low total misclassification cost or small number of high cost errors.

3.1 Total Misclassification Cost

Table 2 shows the results of the comparison between MetaCost_A and AdaBoost, and between MetaCost_CSB and CSB in terms of misclassification costs and cost ratios. A cost ratio for A vs B is a ratio of costs due to A and to B. A ratio of less than 1 for AdaBoost vs MetaCost_A, for example, represents an improvement

due to AdaBoost. The relative performance between CSB and AdaBoost, and between the two versions of MetaCost are shown in the last two columns. The first fourteen datasets are two-class datasets. A summary of the mean costs and geometric mean ratios over the fourteen two-class datasets are also shown. A similar summary is provided for the ten multi-class datasets in the second half of the table. A count of wins/ties/losses over the total twenty-four datasets, and the result of one-tailed sign tests on win/loss records are shown in the last two rows.

In terms of cost, we have the following observations:

- MetaCost usually does not perform better than its internal classifier. AdaBoost performs better than MetaCost_A in eighteen datasets, performs worse in five datasets, and has equal performance in one dataset. CSB performs better than MetaCost_CSB in eighteen datasets and worse in six datasets. Both cases are significant at a level better than 98%. MetaCost retains only a portion of performance of its internal classifier. Using CSB, which is a better performing cost-sensitive classifier than AdaBoost, MetaCost_CSB retains between 68% and 99% of CSB's performance. In only six out of twenty-four datasets, MetaCost_CSB performs comparably to CSB, with a maximum gain of 2% in relative cost.
- MetaCost performs better in two-class datasets than in multi-class datasets. On average, MetaCost_A retains 98% of AdaBoost's performance in two-class datasets, in comparison to only 89% in multi-class datasets. Similarly, MetaCost_CSB retains an average of 91% of CSB's performance in two-class datasets, in comparison to only 80% in multi-class datasets. This is because it is easier to concentrate on minimizing the total cost of problems with two classes, one high cost and one low cost, than that with multiple classes. In addition, an ensemble of multiple models, used in AdaBoost and CSB, is more readily adjustable to the multi-class multi-cost problems than a single model produced by MetaCost.
- Although CSB is significantly better than AdaBoost, MetaCost_CSB is not significantly better than MetaCost_A with thirteen wins and ten losses. Nevertheless, MetaCost_CSB performs better than MetaCost_A on average across the twenty-four datasets. In cases where MetaCost_CSB performs better, the gain can be as much as 60% in relative cost as in the kr-vs-kp dataset, or as much as 69 unit cost as in the nursery dataset. In cases where MetaCost_CSB performs worse, the maximum loss is only 16 unit cost in the satellite dataset. Thus, a better performing internal classifier does give MetaCost a better chance of producing a better performing final model.

3.2 Number of High Cost Errors

Table 3 shows the results of the same comparison in terms of the number of high cost errors. We have the following observations:

- MetaCost almost always performs worse than its internal classifier. MetaCost produces an average of over 30% more relative errors in two-class datasets

Table 3. Comparison of MetaCost, AdaBoost and CSB (High cost errors)

Dataset	MetaC_A	AdaB vs MetaC_A	MetaC_CSB	CSB vs MetaC_CSB	CSB vs AdaB	MetaC_CSB vs MetaC_A
	#hce	#hce ratio	#hce	#hce ratio	#hce ratio	#hce ratio
breast cancer	1.40	.43	2.30	.43	1.67	1.64
liver disorders	3.80	.86	4.90	.80	1.20	1.29
credit	4.35	1.11	6.40	1.02	1.34	1.47
echocardio.	2.00	.75	2.25	.93	1.40	1.13
solar flare	30.10	.99	32.75	.92	1.01	1.09
heart (C)	3.60	.74	4.30	.79	1.28	1.19
hepatitis	3.05	.87	2.80	.75	.79	.92
horse colic	3.40	1.04	4.35	1.09	1.34	1.28
house-voting	0.80	1.00	1.05	.90	1.19	1.31
hypothyroid	2.70	.94	3.65	.92	1.31	1.35
kr-vs-kp	1.40	.82	2.40	.60	1.26	1.71
pima	9.20	.87	9.75	.81	.99	1.06
sonar	2.50	.28	4.55	.33	2.14	1.82
tic-tac-toe	8.00	.16	10.05	.19	1.56	1.26
Mean	5.45		6.54			
Geomean		.69		.69	1.29	1.30
abalone	9.60	.95	7.35	1.13	.91	.77
anneal	0.95	.42	1.15	.52	1.50	1.21
glass	1.35	.63	1.35	.70	1.12	1.00
lymphography	0.45	.44	0.60	.50	1.50	1.33
nettalk-stress	18.45	.67	19.30	.67	1.05	1.05
nursery	2.85	.63	3.90	.56	1.22	1.37
satellite	16.00	.57	23.25	.40	1.03	1.45
soybean	1.00	.40	1.30	.42	1.38	1.30
splice junction	1.85	.73	2.65	.83	1.63	1.43
wine	0.05	.00	0.35	.29	* > 1.00	7.00
Mean	5.26		6.12			
Geomean		.58		.56	†1.24	†1.19
w/t/1		21/1/2		21/0/3	3/0/21	2/1/21
p of w/1		.0000		.0001	.0001	.0000

* divide by zero. † computed from the first nine figures of the multi-class datasets.

than its internal classifier, and over 40% in multi-class datasets. With twenty-one wins out of twenty-four datasets, the differences are significant at a level of 99.99% no matter the internal classifier is CSB or AdaBoost.

- As for the cost measure, MetaCost performs better in two-class datasets than in multi-class datasets. In two-class datasets, MetaCost retains an average of about 70% of the performance of its internal classifier. In multi-class datasets, MetaCost retains an average of about 60% of the performance of its internal classifier.
- A different from the cost-based measure is that MetaCost_A is significantly better in terms of the number of high cost errors than MetaCost_CSB at a level of 99.99%. MetaCost_A achieves a relative error of 30% improvement

over MetaCost_CSB in two-class datasets, and a relative error of more than 19% improvement in multi-class datasets. AdaBoost achieves a similar level of improvement over CSB. This shows that the performance of MetaCost improves in terms of the number of high cost errors if a better performing internal classifier is used.

3.3 The Effect of K

It has been shown that increasing K , the number of classifiers, in the boosting procedure can reduce the number of errors. It is interesting to see the effect of K on MetaCost and its internal boosting procedures in terms of cost and high cost errors.

Figure 4 shows an example of performance comparison between MetaCost_CSB and CSB as K in the boosting procedure increases from 5, 10, 20, 50, 75 to 100 classifiers in the satellite dataset. In terms of high cost errors, both MetaCost_CSB and CSB initially reduce the errors as K increases and then stabilize. Although CSB stabilizes earlier at $K = 20$, with comparison to MetaCost_CSB which stabilizes at $K = 75$, CSB always has fewer errors than MetaCost_CSB. Both MetaCost_CSB and CSB have similar profiles in the figures. As K increases cost initially falls, but then increases. For MetaCost_CSB, the cost increases beyond the point at which the high cost errors stabilized at $K = 75$; for CSB it is at $K = 20$. The increased total cost is due to the increase in low cost errors while the boosting procedure continues its effort to reduce high cost errors, eventually without success.

In terms of tree size, MetaCost_CSB produces a smaller tree as K increases, from a size of 550 nodes at $K = 5$ to 398 at $K = 100$. On the other hand, CSB produces a combined tree size of 2166 nodes at $K = 5$, and increases to 18881 at $K = 100$.

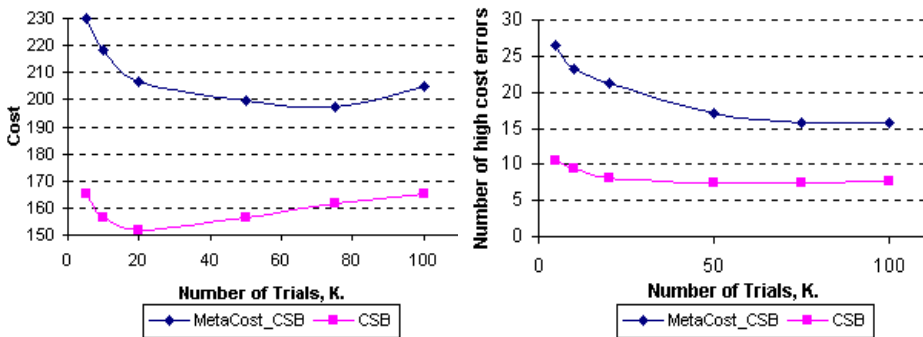


Fig. 4. Satellite: Comparing MetaCost_CSB with CSB in terms of cost and the number of high cost errors as K in the boosting procedure increases

3.4 MetaCost Using Bagging

MetaCost originally uses bagging as its internal classifier (Domingo, 1999). The aims of this section are to investigate how well it performs compared with MetaCost using boosting, and whether MetaCost’s final model performs better than bagging. Table 4 shows the result summary across twenty-four datasets. Detailed results can be found in Ting (2000).

Table 4. Result summary for MetaCost using Bagging, AdaBoost and CSB

	Bagging vs MetaC_B	Bagging vs AdaB	MetaC_B vs MetaC_A	MetaC_B vs MetaC_CSB
Cost ratio geomean	.82	.88	.96	1.06
w/t/l	20/1/3	14/1/9	11/0/13	18/0/6
p of w/l	.0002	.2024	.4194	.0113
#hce ratio geomean	.61	1.50	1.73	1.28
w/t/l	23/1/0	2/0/22	0/0/24	18/3/3
p of w/l	.0000	.0000	.0000	.0007

MetaCost using bagging is found to perform significantly worse than bagging, both in terms of cost and high cost errors. Bagging performs better than AdaBoost in terms of cost and the result carries over to MetaCost, but the differences are not significant. On the other hand, the reverse is true in terms of high cost errors and the differences are significant at a level better than 99.99%. In comparison to that using CSB, MetaCost using bagging performs significantly worse both in terms of cost and high cost errors.

In addition, we compute the percentage of training examples in which the original class is altered to a different class in step (ii) of the MetaCost procedure. Bagging modified an average of 9% of training examples across the twenty-four datasets, and AdaBoost modified an average of 22%. The additional modifications directly contribute to the better performance of MetaCost using AdaBoost over that using bagging in terms of high cost errors.

4 Discussion

The results in this paper show that using a weaker internal classifier, in terms of cost, such as AdaBoost may mislead one to suggest that MetaCost’s final model performs comparably with its internal classifier in terms of cost, especially in two-class datasets. Our results suggest strongly that a better performing cost-sensitive classifier should be used with MetaCost.

MetaCost’s assumption—of using an error-based procedure and then applying the minimum expected cost criterion—has an advantage of re-using the error-based model whenever the cost changes. Unfortunately, the best performing cost-sensitive classifier cannot be obtained this way, and require cost to be taken into consideration in the training process. Our results and the results of previous study (Ting & Zheng, 1998), though using different cost assignment for cost matrix, show that cost-sensitive boosting performs better than AdaBoost in

terms of cost. Further, the results carry over to MetaCost. Even in terms of high cost errors, where AdaBoost is significantly better than CSB, the performance of MetaCost_A can be further improved by taking cost directly into consideration in the training process by changing Equation (2) to

$$w_{(k+1)}(n) = \begin{cases} w_k(n)F_k \text{cost}(y_n, h_k(x_n)) & \text{if } h_k(x_n) \neq y_n \\ w_k(n)/F_k & \text{otherwise.} \end{cases}$$

This improved version of MetaCost_A gains a relative high cost error of 9% against MetaCost_A in both the two-class and multi-class datasets. The difference is significant at a level better than 95% with 17 wins and 7 losses.

Boosting has been shown to outperform bagging (Bauer & Kohavi, 1999) in reducing total errors. Our experiments in Section 3.4 clearly shows that the result does carry over to cost-sensitive classification and MetaCost in terms of reducing high cost errors. However, this result does not necessarily imply that AdaBoost is a better probability estimator than bagging. This is because poor probability estimates can still lead to optimal classification, as long as the class that minimizes expected cost (given the estimated probabilities) is the same as that which minimizes cost given the true probabilities (Domingos, 1999).

5 Conclusions

This paper has investigated two important issues centered on the MetaCost procedure which were ignored in the previous study. First, we find that MetaCost retains only part of the performance of the internal classifier on which it relies, both in terms of cost and high cost errors. This occurs for both boosting and bagging, as its internal classifier. We also find that the better performance of the internal classifier, the better is the chance of improving the final model for MetaCost. Second, using an internal cost-sensitive procedure, MetaCost is expected to perform better than with the original version involving an error-based procedure. This follows as a cost-sensitive procedure usually performs better than an error-based procedure for cost-sensitive classification.

Based on our results, we do not recommend using MetaCost when the aim is to minimize the misclassification cost or the number of high cost errors. MetaCost is only recommended if the aim is to have a more comprehensible model and the user is willing to sacrifice part of the performance.

Acknowledgement

The anonymous reviewers, Geoff Webb and Russell Rimmer provided helpful suggestions to improve the paper. Ardena assisted with the experiments. This research is supported in part by a Deakin University Seeding Grant.

References

1. Bauer, E. & Kohavi, R. (1999), An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36, pp. 105-139, Kluwer Academic Publishers.
2. Blake, C., Keogh, E. & Merz, C.J. (1998), *UCI Repository of machine learning databases* [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, Dept. of Information and Computer Science.
3. Domingos, P. (1999), MetaCost: A General Method for Making Classifiers Cost-Sensitive, in *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, ACM Press.
4. Michie, D., D.J. Spiegelhalter, & C.C. Taylor (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Limited.
5. Quinlan, J.R. (1993), *C4.5: Program for Machine Learning*, Morgan Kaufmann.
6. Quinlan, J.R. (1996), Bagging, boosting, and C4.5, in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 725-730, AAAI Press.
7. Schapire, R.E., Y. Freund, P. Bartlett, & W.S. Lee (1997), Boosting the margin: A new explanation for the effectiveness of voting methods, in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 322-330.
8. Ting, K.M. (2000), An Empirical Study of MetaCost using Boosting Algorithms. *Technical Report C01/00*, Deakin University.
9. Ting, K.M. & Z. Zheng (1998), Boosting Cost-Sensitive Trees, *Proceedings of the First International Conference on Discovery Science*, LNAI-1532, Berlin: Springer-Verlag, pp. 244-255.

Clustered Partial Linear Regression

Luís Torgo ¹ and Joaquim Pinto da Costa ²

LIACC - FEP

R. Campo Alegre, 823, 2º - 4150 PORTO PORTUGAL

Phone : (+351) 22 607 8830, Fax : (+351) 22 600 3654

ltorgo@ncc.up.pt

<http://www.ncc.up.pt/~ltorgo>

LIACC – DMA/FC

R. Campo Alegre, 823, 2º - 4150 PORTO PORTUGAL

Phone : (+351) 22 607 8830, Fax : (+351) 22 600 3654

jpcosta@ncc.up.pt

Abstract. This paper presents a new method that deals with a supervised learning task usually known as multivariate regression. The main distinguishing feature of this new technique is the use of a clustering method to obtain sub-sets of the training data before the learning phase. After this “resampling” process a different regression model is fitted to each found cluster. We call the resulting method *clustered partial linear regression*. Predictions using this technique are preceded by a cluster membership query for each test case. The cluster membership probability of a test case is used as a weight in an averaging process that calculates the final prediction. This averaging process involves the predictions of the regression models associated to the clusters for which the test case may belong. We have tested this general multi-strategy approach using several regression techniques and we have observed significant accuracy gains in several data sets. We have also compared our method to bagging that also uses an averaging process to obtain predictions. This experiment showed that the two methods are significantly different. Finally, we present a comparison of our method with several state-of-the-art regression methods.

Keywords: Regression, Clustering, Multi-strategy learning, Multiple models.

Category: Long paper.

1 Introduction

This paper describes *clustered partial linear* models. This is a new method for addressing multivariate regression problems. Multivariate regression is a supervised learning task that can be loosely defined as the search for a model of the relationship between a target continuous variable and a set of other input variables (attributes, features). The technique we describe deals with this problem using a multi-strategy approach. In an initial stage a set of samples of the available training data is obtained using a clustering method. This step is motivated by the assumption that on reasonably

complex domains it should be easier to model sub-sets of “similar” training cases than to try to fit a single model to all data. After this initial resampling phase we fit a regression model to each of the found clusters. Although this general two-stage schema can be applied to any regression method (and even to other supervised learning tasks), in this paper we concentrate our description on *partial linear* regression models (e.g. [11, 16]). Still, we also report some results using other regression techniques. Partial linear models belong to the class of semiparametric approaches that integrate parametric with non-parametric techniques. In the case of partial linear models, a standard least squares linear polynomial (e.g. [8]) is integrated with a kernel smoother [13, 17]. The main motivation behind these models is to retain as much as possible the comprehensibility of linear polynomials, while trying to improve their accuracy by adding a smoothing component that compensates, on a query-base, for the local inadequacies of the linearity assumption of first order polynomials.

The use of *clustered partial linear* models for predicting the target value of a test case also involves two stages. In a first step we collect a set of cluster membership values that represent the probability of the test case belonging to each cluster. Using these probabilities and the predictions of the regression models in each cluster we calculate the final prediction of our model through a weighed average of these individual predictions.

This paper is organized as follows. The next section describes partial linear regression that is the basic technique that we use within our clustered approach to regression. Section 3 presents clustered partial linear models. In Section 4 we describe a series of experiments with these models. A further analysis of clustered partial linear regression is given in Section 5. Finally, we present the main conclusions of this work.

2 Partial Linear Models

Partial linear regression [11, 16] is a semiparametric technique that integrates a linear polynomial with a kernel smoothing component. A prediction for a query case using these models is obtained by summing the value predicted by the linear model with the value resulting from smoothing over the residuals (errors) of the linear model in the neighbouring training points. The more inadequate the linear model is to the given training sample the larger the importance of the smoothing component. In the extreme case where the linear component perfectly fits the training data, a partial linear model reduces to a standard least squares linear polynomial.

Given a data set, $\{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^n$, where $\mathbf{x}_i$ is a vector of attribute values, a linear regression model of the form $Y = \beta_0 + \beta_1 X_1 + \dots + \beta_a X_a$, can be obtained using a least squares error criterion. This consists of finding the vector of parameters $\boldsymbol{\beta}$ that minimizes the sum of the squared error, i.e. $(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$, where $\mathbf{X}'$ denotes the transpose of matrix $\mathbf{X}$. After some matrix algebra the minimization of this expression with respect to $\boldsymbol{\beta}$ leads to the following set of equations, usually referred to as the *normal equations* (e.g. [8]),

$$(\mathbf{X}'\mathbf{X})\beta = \mathbf{X}'\mathbf{Y} . \quad (1)$$

The parameter values can be obtained solving the equation,

$$\beta = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Y} \quad (2)$$

where $\mathbf{X}^{-1}$ denotes the inverse of matrix $\mathbf{X}$.

As the inverse matrix does not always exists this process suffers from numerical instability. A better alternative [14] is to use a set of techniques known as *Singular Value Decomposition* (SVD), that can be used to find solutions of systems of equations with the form $\mathbf{X}\beta = \mathbf{Y}$.

A kernel smoother [13, 17] can be seen as a form of lazy learner [1] that delays learning till prediction time. Given a query point $\mathbf{x}_q$, a prediction is obtained using the following expression,

$$ks(\mathbf{x}_q) = \frac{1}{SKs} \sum_{i=1}^n K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times y_i \quad (3)$$

where,

$d(.)$ is the distance function between two instances;

$K(.)$ is a kernel function;

h is a bandwidth value;

$\langle \mathbf{x}_i, y_i \rangle$ is a training instance;

and SKs is the sum of the weights of all training cases, *i.e.*

$$SKs = \sum_{i=1}^n K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) .$$

This formula is a weighed average over the target values of the training cases that are nearer to the query point. The notion of neighborhood implies the definition of a metric over the multi-dimensional space defined by the input variables and of a distance function between any two cases in this space. The bandwidth size, h , defines the size of the neighborhood that “enters” the weighed average. The kernel function, $K(.)$, provides a smoothing effect, giving more “importance” to nearer training cases. Many different variants of all these “parameters” of kernel smoothing are described in the literature (*e.g.* [2]).

Partial linear models integrate a linear polynomial with a kernel smoother applied on the residuals (errors) of the polynomial. The role of the kernel smoother is to provide an estimate of the error of the linear polynomial for the particular query case under consideration. This estimated error is then added to the linear polynomial prediction giving the predicted value of the partial linear model. Formally, this can be described by,

$$pl(\mathbf{x}_q) = \beta \mathbf{x}_q - \frac{1}{SKs} \sum_{i=1}^n K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times e_i \quad (4)$$

where,

e_i is the error of the linear model in case $\langle \mathbf{x}_i, y_i \rangle$, given by $e_i = \beta \mathbf{x}_i - y_i$.

Thus, to obtain a prediction for a query case using a partial linear model, we start by obtaining the predicted value of the linear polynomial, $\beta \mathbf{x}_q$. We then calculate the error of this linear polynomial in the training cases that are nearer to the query point, $\mathbf{x}_q$. Using these errors we obtain a kernel prediction of the error for the query case. Finally, this predicted error is added to the initial value predicted by the linear polynomial giving the prediction of the partial linear model.

Compared to linear polynomials, partial linear models have significant advantages in terms of predictive accuracy when the domain is non-linear. Contrary to kernel smoothers, partial linear models have some degree of comprehensibility due to the use of a linear polynomial. However, as they also incorporate a kernel component (that is not comprehensible), they are less understandable than linear regression. In effect, the polynomial component of partial linear models can only be regarded as a rough description of the true surface approximated by these models. The accuracy of this description is proportional to the linearity of the domain under study. In highly non-linear domains the predominance of the kernel “corrections” is so high that the linear polynomial is a very poor “explanation” of the predictions of the partial linear model.

3 Clustered Partial Linear Models

The regression method we propose integrates a clustering technique with a partial linear model. The key idea is to obtain partial linear models for clusters of data instead of fitting a single model to all data.

The first step of our methodology consists of obtaining a clustering of the data. For this purpose we have used system AUTOCLASS C¹ [5, 6]. The main motivation for this choice was the fact that AUTOCLASS C provides the features that we need to implement our method, namely, cluster membership probabilities and automatic choice of the number of clusters. As the goal of this clustering stage is to find groups of “similar” training cases, we have not used the information about the target variable values in this task. This means that AUTOCLASS C only receives information on the input variables. This system outputs the number of found clusters and attaches to each training case the probability of belonging to each of the clusters. Using this information we create a set of training samples, one for each cluster. If a training case has some probability of belonging to more than one cluster it is included in the respective training samples. This means that there may exist some degree of overlap between these training samples.

¹ The program is freely available at <http://ic-www.arc.nasa.gov/ic/projects/bayes-group/autoclass/>.

After this clustering-based resampling of the training data, we fit a partial linear model to each of the samples. This consists of obtaining a least squares linear polynomial for each cluster, because the kernel component does not involve any “training”. Thus, clustered partial linear models consist of a set of partial linear models built in different clusters of the data. If the clustering algorithm is able to produce a symbolic representation of each cluster it is possible to consider clustered partial linear models as a single model. In effect, the symbolic representation of each cluster can be seen as a condition for applying the respective partial linear model, which means that we can look at a clustered partial linear model as a set of rules of the form “IF <cluster representation> THEN <partial linear model>”.

Regarding predictions using clustered partial linear models they are obtained as follows. Given a query case, $\mathbf{x}_q$ we obtain the probabilities of belonging to each of the clusters. For each cluster with membership probability higher than zero, the respective partial linear model is used to obtain a prediction for the query case. These predictions are then averaged to obtain the final predicted value using the following formula,

$$CPL(\mathbf{x}_q) = \sum_{k=1}^j (P_k(\mathbf{x}_q) \times pl_k(\mathbf{x}_q)) \quad (5)$$

where,

j is the number of clusters;

$P_k(\mathbf{x}_q)$ is the probability of the query case belonging to cluster k ;

$pl_k(\mathbf{x}_q)$ is the prediction of the partial linear model of cluster k (c.f. Eq. 4).

4 Experimental Analysis

In this section we describe a series of experiments with clustered partial linear models. The experiments we report are carried out using the data sets shown in Table 1. With respect to the experimental methodology all reported results are averages of five repetitions of a 10-fold Cross Validation experiment. Significance of the differences in mean squared error (MSE) are asserted through paired t -tests.

Table 1. The used data sets

Data Set	Main characteristics
Housing	506 case; 13 continuous variables. Predicting housing values in Boston.
Abalone	4177 cases; 7 cont. vars.; 1 nominal var. Predicting the age of abalone.
Elevators	8752 cases; 40 cont. vars. Aircraft control problem (prediction of elevators level).
F	3000 cases; 5 cont. vars. Artificial domain with marked clusters of data points.
Kinematics	8192 cases; 8 cont. vars. Robot arm control problem.
Computer	8192 cases; 22 cont. vars. Prediction of CPU activity level in a computer network.
Computer (small)	8192 cases; 12 cont. vars. Simplified version of the previous data set.
Telecomm	15000 cases; 49 cont. vars. Telecommunications problem.

The first experiment we report was designed to answer the following question:

- *Is there a significant difference in accuracy between applying the models in clustered training samples or to all available training data?*

We have tested our clustering-based method with three different types of regression models: partial linear models; linear regression models; and regression trees. For each of these trials we have compared the “clustered” variant, obtained using the methodology described in Section 3, with the alternative of simply applying the regression model to all available training data. The results are shown in Table 2. Significant wins (99% confidence) of the “clustered” versions are marked with “+” signs, while the opposite appears with “-“ signs.

Table 2. The advantages of clustering the data

Data Set	Partial Linear Models			Linear Regression			Regression Trees		
	Clustered	All data		Clustered	All data		Clustered	All data	
Housing	13.04	16.94	+	14.89	22.83	+	25.42	20.13	
Abalone	2.28	4.75	+	2.22	5.00	+	5.55	5.41	
Elevators	5.61	15.07	+	9.95	37.23	+	9.73	17.38	+
F	1.56	3.45	+	5.99	28.96	+	4.31	7.56	+
Kinematics	0.010	0.012	+	0.030	0.041	+	0.032	0.039	+
Computer	24.38	22.07	-	152.89	220.89	+	8.21	12.35	+
Computer (small)	13.98	13.93	-	80.30	241.16	+	7.65	14.36	+
Telecomm	89.86	53.72	-	631.51	823.97	+	76.71	61.90	-

These experiments confirm the advantages of pre-clustering the data as we propose. In effect, with all three regression methods (that are quite different) there is a general trend towards a significant gain in predictive accuracy. However, clustering the given training data is just one of the distinguishing factors of our methodology. Another difference to the non-clustered methods, is the averaging of the predictions of different models. Regarding this issue our methodology resembles bagging [3], where predictions are obtained by averaging over a set of models obtained using different bootstrap samples of the given data. According to Breiman [3] bagging is expected to give good results when the base prediction method is sensible to small perturbations on the learning set, as it is the case of regression trees. However, that is not the case of partial linear models that are quite robust to these variations, as the results of Table 3 confirm. This table shows the results of a comparison between partial linear models and regression trees with their respective “bagged” versions. The results confirm Breiman’s statement. The “bagged” models were obtained using 50 bootstrap replicates of the data. Significant wins of the bagged versions are marked with ‘+’ signs.

Table 3. Bagging regression models

Data Set	<i>Partial Linear Models</i>			<i>Regression Trees</i>		
	Single	Bagged		Single	Bagged	
Housing	16.94	17.12		20.13	13.02	+
Abalone	4.75	4.68	+	5.41	4.63	+
Elevators	15.07	15.11		17.38	10.18	+
F	3.45	3.41		7.56	4.04	+
Kinematics	0.012	0.012		0.039	0.024	+
Computer	22.07	22.05		12.35	8.20	+
Computer (small)	13.93	14.14	-	14.36	9.75	+
Telecomm	53.72	54.56	-	61.90	37.08	+

The results of these experiments with bagging show that the accuracy advantages of clustered partial linear models that were observed in Table 2, can only be caused by the effects of clustering the training data. In effect, the results of Table 3, indicate that there is nothing to gain with averaging over several partial linear models.

Having shown the advantages of a pre-clustering of the training sample, it remains an open question whether the results of clustered partial linear models are good when compared to other existing approaches to regression. The following experiment has the goal of answering this question. We have compared clustered partial linear models with several state-of-the-art regression systems, namely, CUBIST [<http://www.rule-quest.com>], MARS [10] and a bagged version of CART [4]. The results of this experiment are shown in Table 4. Significant wins of clustered partial linear models are marked with ‘+’ signs.

Table 4. Clustered partial linear regression (CPL) versus other approaches

Data Set	CPL	CUBIST	MARS	BaggedCART
Housing	13.04	14.24	18.32 +	13.02
Abalone	2.28	4.67 +	4.54 +	4.63 +
Elevators	5.61	12.91 +	6.04	10.18 +
F	1.56	13.64 +	19.95 +	4.04 +
Kinematics	0.010	0.027 +	0.036 +	0.024 +
Computer	24.38	6.49 -	10.22 -	8.20 -
Computer (small)	13.98	9.71 -	14.00	9.75 -
Telecomm	89.86	91.51	? ²	37.08 -

Clustered partial linear models achieved quite competitive accuracy in most domains. Some results are particularly outstanding, namely in the *Abalone*, *F* and *Kinematics* domains. Moreover, in the case of *Abalone* and *F*, the excellent scores are clearly caused by the clustering step because neither the single models nor the “bagged” versions obtain similar results (*c.f.* Tables 2 and 3). However, the results obtained on the two *Computer* domains and in the *Telecomm* application are a bit disappointing. A possible cause of these results is the complete inadequacy of linear polynomials to these domains, which can be confirmed by the results of Table 2. Although partial linear models include a smoothing component that could overcome this mismatch, there are situations where this is not possible. In effect, the lack of symmetry near the boundaries of the input space causes well known difficulties to kernel smoothers [12]. The extrapolation capabilities of linear polynomials may also lead to “wild” predictions near these boundaries. These two factors together may explain the poor performance on these domains. This explanation is consistent with the fact that clustered regression trees (that do not have such difficulties) do not achieve such disappointing results (*c.f.* Table 2). Thus, we claim that these poor results are caused by the lack of adequacy of the base regression models to the domains and not by any difficulty of our proposed methodology.

5 Discussion

The main motivation behind our proposal is the hypothesis that modeling a small set of nearby cases is easier than modeling a larger sample of instances. Although we do not provide any theoretical proof of this hypothesis we have collected a series of experimental results that we claim to provide good indications to support this. Moreover, there is a strong relation between this hypothesis and the way lazy learners proceed. In effect, this type of techniques can be seen as obtaining a local model around the neighborhood of each query case. These methods are able to easily capture the local regularities of the regression surface and thus achieving excellent accuracy.

² The version we have of MARS (3.6) gives a “segmentation fault” on this data set.

Our methodology differs from these approaches in that the local neighborhoods are pre-defined at the start through a clustering method instead of being a function of each query case. This has large computational advantages over “pure” lazy learners. Moreover, as there is a fixed (and usually small) set of pre-defined clusters we can obtain a comprehensible model of the data, which is not the case of lazy learners.

In spite of the promising accuracy results our methodology also has some disadvantages. The more noticeable is the increment of computation time when compared to applying a regression model to all training data. This additional cost is caused by the clustering step. AUTOCLASS C has many parameters that could be explored in order to try to improve this speed issue. Still, clustering is always a heavy task and its weight in the overall computation time of clustered partial linear models will always be high.

As we have already mentioned there are some relations of our method with multiple model approaches like bagging [3]. These relations have to do with the construction of different models based on possibly overlapping samples of cases and in averaging the predictions of these models as a form to obtain predictions for test cases. However, the method used to obtain the individual samples is totally different. In bagging a bootstrap random sampling process is used to obtain samples with the same size as the original training sample. In our methodology the samples are not obtained randomly and are usually smaller than the original training sample. The type of resampling carried out by our clustering step changes the distribution of the cases in the original sample, which is not the case with bagging. From this perspective, our method is related to boosting [15, 9], where the same distribution change is done through a system of weights. However, contrary to boosting our method is not sequential and thus it is possible to construct the individual models in parallel.

Devogelaere *et al.* [7] describe a related approach to regression. Their GAdC system performs a genetic algorithm driven clustering of the training data. The evaluation function that drives the genetic algorithm-based search for the clusters includes several factors like cluster distance penalty, prediction error, etc. This means that, unlike our method, GAdC uses information of the regression accuracy to guide the search for clusters. Within each cluster, GAdC uses a kind of kernel model to obtain predictions. Another difference of our work is the probabilistic approach to cluster membership that leads to overlap of clusters and also to the averaging of different clustered models.

The method described in this paper can also be related to typical partition-based learners, like tree-based models or rule-based systems. These methods also partition the given sample in a set of local regions and fit some kind of model within each of these regions. However, there are some fundamental differences to our method. The more important is the criteria used to form the partitions. Regression trees (*e.g.* [4]), for instance, search for regions of low variance in the target variable. Our method does not use any information regarding the target variable to obtain the partitions. On the contrary the partitions are obtained based on information concerning the input variables. Other differences to these approaches are the possible overlap of the regions, the probabilistic approach to decide on which cluster to “place” a test case, and the averaging over different regions in prediction tasks.

In the near future we intend to extend our experimental evaluation of the method, with the goal of clearly understanding what are the key factors affecting its results. We also intend to further experiment with tuning of the clustering step. Finally, based on the poor results of clustered partial linear models in some domains, that we claim to be caused by the regression model, we intend to explore the possibility of using different regression models for each cluster.

6 Conclusions

We have described an approach to multivariate regression whose main distinguishing feature is the use of a clustering algorithm to obtain samples of the available training data. These samples are then modeled individually through partial linear models. Predictions using the resulting clustered partial linear models are obtained by averaging over the models in the clusters for which the membership probability of the test cases is higher than zero.

We have tried this clustering-based approach to regression with three different types of regression models. With all three methods we have observed a similar pattern of results, showing the advantages of pre-clustering the training data.

We have compared our method to bagging that also uses averaging over multiple models. The results show that there are significant differences between the two methods, with some clear advantages of our approach.

Compared to existing state-of-the-art regression approaches our method achieved quite competitive results in the tested domains. The accuracy in some domains can be considered quite outstanding, deserving further analysis.

References

1. Aha,D.: *Lazy Learning*. Kluwer Academic Publishers, 1997.
2. Atkeson,C., Moore,A., Schaal,S.: Locally Weighted Regression. *Artificial Intelligence Review*, **11**, 11-73, 1997.
3. Breiman, L.: Bagging predictors. In *Machine Learning*, **24** (2). Kluwer Academic Publishers, 1996.
4. Breiman,L. , Friedman,J.H., Olshen,R.A. and Stone,C.J.: *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, California, USA, 1984.
5. Cheeseman,P., Kelly,J., Self,M., Stutz,J.: Autoclass: a bayesian classification system. In Proceedings of the 5th International Conference on Machine Learning. Morgan Kaufmann, 1988.
6. Cheesman,P., Stutz,J.: Bayesian classification (AUTOCLASS): theory and results. In Advances in Knowledge Discovery. AAAI Press, 1995.
7. Devogelaere,D., Van Bael,P., Rijckaert,M.: Regression through genetic algorithm driven clustering. In Proceeding of the 7th European Congress on Intelligent Techniques & Soft Computing (EUFIT'99), 1999.
8. Drapper,N., Smith,H.: *Applied Regression Analysis* (2nd edition). John Wiley & Sons, 1981.

9. Freund, Y., Shapire, R.: Experiments with a new boosting algorithm. In Proceedings of the 13th International Conference on Machine Learning. Morgan Kaufmann, 1996.
10. Friedman, J. (1991): Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, **19** : 1-141.
11. Hardle, W.: *Applied Nonparametric Regression*. Econometric Society Monographs. Cambridge University Press, 1990.
12. Hastie, T., Loader, C.: Local regression : automatic kernel carpentry. *Statistical Science*, **8**, 120-143, 1993.
13. Nadaraya, E.A.: On estimating regression. *Theory of Probability and its Applications*, **9**, 141-152, 1964.
14. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: *Numerical receipes in C*. Cambridge University Press, 1992.
15. Shapire, R.: The strength of weak learnability. *Machine Learning*, **5**, 197-227. Kluwer Academic Publishers, 1990.
16. Spiegelman, C.H.: Two techniques for estimating treatment effects in the presence of hidden variables: adaptive regression and a solution to Reiersol's problem. Unpublished Ph.D. Thesis, Northwestern University, Dept. of Mathematics, 1976.
17. Watson, G.S.: Smooth regression analysis. *Sankhya: The Indian Journal of Statsitics, Series A*, **26**, 359-372, 1964.

Knowledge Discovery from Very Large Databases Using Frequent Concept Lattices

Kitsana Waiyamai¹ and Lotfi Lakhal²

¹ Faculty of Engineering, Kasetsart University, Bangkok Thailand

² LIMOS, University of Clermont-Ferrand II, France

Abstract. Data clustering and association rules discovery are two related problems in data mining. In this paper, we propose to integrate these two techniques using the *frequent concept lattice* data structure - a formal conceptual model that can be used to identify similarities among a set of objects based on their frequent attributes (frequent items). Experimental results show that clusterings and association rules are generated efficiently from the frequent concept lattice, since response time after lattice construction is measured almost in seconds.

1 Introduction and Motivation

Recently, several clustering methods have been developed in the framework of the concept lattice [1,4,2] which focus on discovering all possible concepts. These methods are inefficient under the context of large databases. First, they have been designed to work in main memory with small datasets, thus limiting their suitability for data mining in large databases. Second, most of them perform an exhaustive search of all possible concepts, whereas only part of them is considered useful by users [5]. An efficient learning method in a real-world context, that does not carry out an exhaustive search of the whole concept lattice, must be provided. The idea of generating associations between items from concept lattices has also been addressed early in several works, e.g., [11,3]. However, the rules generated by these methods are particular cases of association rules, i.e. they are association rules with confidence equal to 100%. Further, their associated algorithms work only in main memory.

In this paper, we propose a data mining framework using a concept lattice [10] as a tool for the knowledge discovery, with an emphasis on the integration of data clustering and association rule discovery from large databases. The idea is to preprocess the database and derive a *frequent concept lattice* - a data structure that encodes the information needed for our discovering tasks. A frequent concept lattice is a part of the concept lattice in which each concept covers at least some initial minimum number of objects of the database, that say the items (in the concept's intent) shared by those objects in the concept's extent must have a support greater or equal the user-specified initial support *init_sup*. The support of an itemset is defined as the percentage of objects in the database containing that itemset.

From the frequent concept lattice, *pertinent concepts* and *strong association rules*, with respect to the user point of view, are generated. We developed a collection of operators to generate all pertinent concepts and strong association rules directly from the frequent concept lattice without further access to the original database. Experimental results show that knowledge are discovered from the frequent concept lattice, with response time after lattice construction measured almost in seconds. The rest of the paper is organized as follows: Section 2 begins by formally defining the frequent concept lattice model. Then, an algorithm for building a frequent concept lattice from a given a database is described. Section 3 provides experimental results with data extracted from the statistical Census data of Kansas USA¹. Section 4 concludes with a summary and future works.

2 Frequent Concept Lattice Based Data Clustering

In this section, we begin by formally defining the frequent concept lattice model. Then, we discuss a process for identifying clusters form a data mining context based on the frequent concept lattice construction.

2.1 The Frequent Concept Lattice Model

Following, we give the formal definitions of *data mining context*, *Galois connection*, *concept*, *frequent concept* and finally *frequent concept lattice*.

Data mining context A data mining context (a database) is defined as a triple $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, where $\mathcal{O}$ and $\mathcal{I}$ are finite sets of objects (or transactions) and database items, $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$ is a binary relation. Each couple (o, i) of $\mathcal{R}$ denotes the fact that the object o of $\mathcal{O}$ has the item i of $\mathcal{I}$. Figure 1 and 2 represent the data mining context example using horizontal and vertical representations.

Object ID	Items
1	{A F}
2	{A F G}
3	{A B F G}
4	{B F G H}
5	{A C E}

Fig. 1. Data mining context using a horizontal representation

Item	OIDs
A	{1 2 3 5}
B	{3 4}
C	{5}
D	{}
E	{5}
F	{1 2 3 4}
G	{2 3 4}
H	{4}

Fig. 2. Data mining context using a vertical representation

Galois connection Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ be a data mining context. For $O \subseteq \mathcal{O}$ and $I \subseteq \mathcal{I}$, we define $f : 2^{\mathcal{O}} \rightarrow 2^{\mathcal{I}}, f(O) = \{i \in \mathcal{I} \mid \forall o \in O \cdot o\mathcal{R}i\}$ and conversely $g : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{O}}, g(I) = \{o \in \mathcal{O} \mid \forall i \in I \cdot o\mathcal{R}i\}$. That is, $f(O)$ is the set of all items, called *itemset*, common to all objects in O , and $g(I)$ is the set of

¹ <ftp://ftp2.cc.ukans.edu/pub/ippr/census/pums/pums90ks.zip>

all objects which have all items in I . The pair (f, g) forms a Galois connection between $2^{\mathcal{O}}$ and $2^{\mathcal{I}}$, the power sets of $\mathcal{O}$ and $\mathcal{I}$, respectively. Furthermore, the compositions fog and gof ² are closure operators.

Concept A concept of the data mining context $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ is defined as a pair $(Extent, Intent)$, where $Extent \subseteq \mathcal{O}$, $Intent \subseteq \mathcal{I}$, $f(Extent) = Intent$ and $g(Intent) = Extent$. Hence, a concept c is formed by two parts: an extent represents a subset of objects, denoted as $Extent(c)$, and an intent represents the common items between this subset of objects, denoted as $Intent(c)$. That is, a concept is a maximal collection of objects sharing common items.

Frequent concept Let L be a set of all concepts formed from $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, and c a concept of L . Let $init\_sup$ be a user-specified initial support. A *support* threshold associated to a concept c , denoted as $supp(c)$, is the percentage of objects in $\mathcal{O}$ having precisely the same set of items in the intent of c , which we can be defined as: $supp(c) = \frac{\|Extent(c)\|}{\|\mathcal{O}\|}$. A concept is said *frequent* if its support threshold is greater or equal to $init\_sup$.

Frequent concept lattice Let L be a set of all concepts formed from $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, $init\_sup$ a user-specified initial support, and FL a set of all frequent concepts, i.e. $FL = \{c \in L \mid supp(c) \geq init\_sup\}$. The frequent concept lattice $\mathcal{FL} = (FL \cup \{\perp\}, \leq)$ of a data mining context $\mathcal{D}$, is a complete lattice of frequent concepts derived from $\mathcal{D}$. *Proof* of this property is given in [8]. Figure 3 shows the frequent concept lattice (using a graph-oriented representation) derived from the data mining context example with $init\_sup$ equal to 35% (at least 2 objects are contained in each concept).

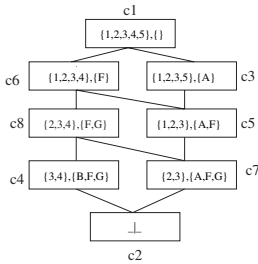


Fig. 3. Graph-oriented representation of the *frequent concept lattice* with the support threshold $\geq 35\%$ (2/5)

Concept-ID	Extent	Intent
c1	{1,2,3,4,5}	{}
c2	{}	{A,B,F,G}
c3	{1,2,3,5}	{A}
c4	{3,4}	{B,F,G}
c5	{1,2,3}	{A,F}
c6	{1,2,3,4}	{F}
c7	{2,3}	{A,F,G}

Fig. 4. Database-oriented representation (nested relation) of the *frequent concept lattice*

2.2 The Lattice Clustering Algorithm

In contrast with existing works [11,4,6,5,2] where concept lattices are represented using a graph-oriented representation (where sub-superconcept relationships between concepts are explicitly represented), we address concept lattice representation in database systems from a set-oriented perspective. In our approach,

² We use the following notations: $fog(O) = g(f(O))$ and $gof(I) = f(g(I))$.

concept lattices are represented as finite sets of concepts in which *concept lattice constraints* [9] are given, insuring that the sets effectively preserve the lattice structures. Figure 4 shows the frequent concept lattice using a database-oriented representation. From a data mining context, we want to derive the frequent concept lattice associated to that context, that is to say finding all the frequent concepts (Ext, Int) such that Ext is the maximal set of objects which all have precisely the same set of items in Int and such that the number of those objects represents a percentage no less than the initial support $init\_sup$. The idea is to successively add a new frequent item (i.e. item that has its support greater or equal to the initial support $init\_sup$) to the current frequent concept lattice. Formal characterization of inserting a new frequent item into a frequent concept lattice is given in [8].

procedure Insert_frequent_item(i : new frequent item, $g(\{i\})$: object set containing i , $init\_sup$: initial support);

begin

```

0)  $Exist = \{\}$ ; //  $Exist$  contains extended concepts or new concepts
1) for  $j = |Intent(\perp)|$  downto  $|Intent(Sup)|$  do
2)    $levelj = \text{select } c \text{ from } c \text{ in } \mathcal{FL} \text{ where } |Intent(c)| == j$ ;
3)   for all concept  $c \in levelj$  do
4)     if  $(Extent(c) \subseteq g(\{i\}))$  then //  $c$  is a extended frequent concept
5)        $Intent(c) = Intent(c) \cup \{i\}$ ;
6)       if  $(Extent(c) == g(\{i\}))$  then exit procedure endif;
7)        $Exist = Exist \cup Extent(c)$ ;
8)     else
9)        $inter = g(\{i\}) \cap Extent(c)$ ;
10)      if  $(|inter| \geq (init\_sup \times |\mathcal{D}|))$  and  $inter \notin Exist$  then
11)        Create a new frequent concept  $nc$ ;
12)         $Extent(nc) = inter$ ;  $Intent(nc) = Intent(c) \cup \{i\}$ ;
13)         $\mathcal{FL} = \mathcal{FL} \cup \{nc\}$ ;
14)        if  $(inter == g(\{i\}))$  then exit procedure endif;
15)         $Exist = Exist \cup inter$ ;
16)      endif;
17)    endif;
18)  end;
19) end;
end

```

Algorithm Lattice_clustering($init\_sup$: initial support, $\mathcal{D}$: data mining context);

begin

```

0) Initialize  $Sup$  and  $\perp$ ;
1)  $\mathcal{FL} = \{Sup, \perp\}$ ;
2) for  $k = 1$  to  $|\mathcal{I}|$  do
3)   if  $(supp(i_k) \geq init\_sup)$  then
4)      $Insert\_frequent\_item(i_k, g(\{i_k\}), init\_sup)$ ;
5)   endif;
6) end;
end

```


The algorithm for building a frequent concept lattice from a database given above takes as input (i) an item i to be inserted, (ii) its associated object set $g(\{i\})$, (iii) initial support $init\_sup$, (iv) a frequent concept lattice $\mathcal{FL}$ to be updated, and a database $\mathcal{D}$. It starts by initializing the lattice with the Top and $\perp$ elements. The Top element corresponds to the most general (with respect to set inclusion) frequent concept in the lattice, i.e. its extent contains all the objects of the database All). The $\perp$ element corresponds to the most specific (with respect to set inclusion) frequent concept in the lattice, i.e. its intent contains all the frequent items of the database All). Then, the construction of $\mathcal{FL}$ consists of successively calls to the procedure *Insert_frequent_item* which inserts a new frequent item i to the current frequent concept lattice. Each insertion may discover new frequent concepts and/or augment the intents of existing frequent concepts. The main loop of the procedure *Insert_frequent_item* (lines 1-19) visits the concepts of the lattice $\mathcal{FL}$ level by level in decreasing cardinality of their intent. At each level, an SQL query is executed on $\mathcal{FL}$ in order to load all the concepts with the same cardinality of their intent in main memory (line 2 of the procedure). Then, it examines successively all the concepts of the same level (lines 3-18 of the procedure). For each concept c of $\mathcal{FL}$, the procedure tests how it relates to $g(\{i\})$:

- if $Extent(c) \subseteq g(\{i\})$ (i.e. its extent is more general than or equal to the set of objects associated to the new item i), c is an *augmented concept*, then its intent is augmented by the new item i (lines 4-5 of the procedure),
- else the new extent $inter = Extent(c) \cap g(\{i\})$ is calculated. To verify if $inter$ is not already appeared in any concept of $\mathcal{FL}$, just examine all the frequent concepts that have been augmented or created previously: that's the role of the list *Exist* which keeps all concepts newly augmented or created. If $inter \notin Exist$ and $|inter| \geq (init\_sup \times |\mathcal{O}|)$, then c is a generator of the *new frequent concept* $(inter, Intent(c) \cup \{i\})$. This is valid because all existing concepts are treated in decreasing cardinality of their intent, the first concept encountered which gives a new intersection is the generator of the new concept because it is necessary the smallest concept (with respect to number of its objects).

The execution of the procedure *Insert_frequent_item* terminates when a frequent concept is encountered (line 6) or created (line 14) for which its extent is equal to $g(\{i_k\})$. Using the database example (c.f. Figure 2), the iterative steps of building a frequent concept lattice with initial support = 35%(2/5), i.e. each concept of the lattice must covers at least 2 objects of the database, is illustrated³ below in Figure 5. The algorithm starts by initializing the lattice with the two elements: $Sup = c1 = (All, \emptyset)$ and $\perp = c2 = (\{3\}, All)$ (line 0). Then, for each frequent item, the procedure *Insert_frequent_item* is called. Inserting the item A results in the insertion of a new frequent concept $c3$ generated by the concept generator $c1$. Inserting the item B results in discovering of two new frequent concepts:

³ We use a Graph-oriented representation of the lattice in order to facilitate understanding.

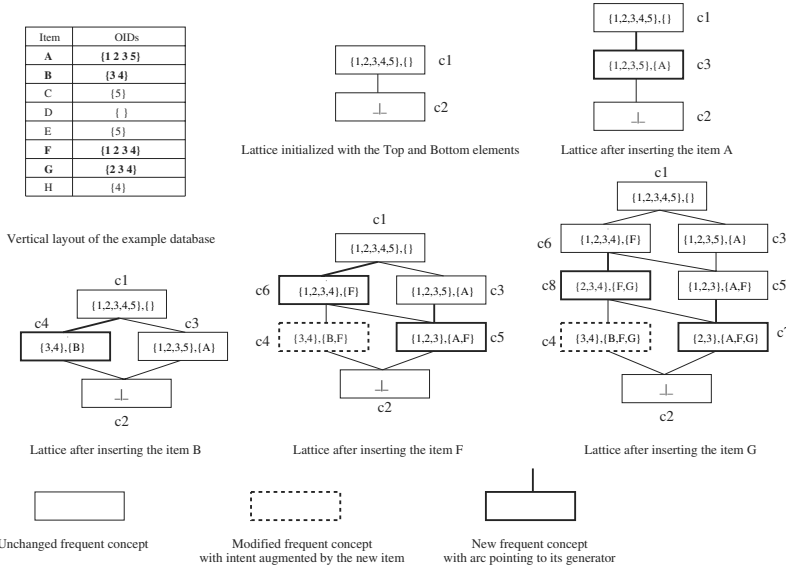


Fig. 5. Iterative steps of building a frequent concept lattice with initial support $= 35\%(2/5)$

$(\{3\}, \{A, B\})$ generated by $(\{1, 2, 3, 5\}, \{A\})$ and $(\{3, 4\}, \{B\})$ generated by $c1$. However only $c4 = (\{3, 4\}, \{B\})$ is added to the lattice since its support threshold is greater than the initial support *init_sup* given by the users. Inserting the item F results in the insertion of two new frequent concepts: $c5$ generated by $c3$ and $c6$ generated by $c1$, and the modification of the old frequent concept: $c4$. Inserting the item G results in the insertion of two new frequent concepts: $c7$ generated by $c5$ et $c8$ generated by $c6$, and the modification the old frequent concept: $c4$.

3 Knowledge Discovering from the Frequent Concept Lattice

Once constructed, the frequent concept lattice can be used as a support for generating pertinent concepts and strong association rules. We propose a collection of operators for the tasks of knowledge discovering: lattice operators (UB, LB, MEET, JOIN, ...) and rule discovery operator. For further details of the formal definitions of all these operators, interested reader should consult ([9, 8, 7]). To assess their relative performances of the proposed operators, our algorithms were implemented in O_2C object database programming language provided by the O_2 OODBMS. The platform we used was a 43P240 bi-processor IBM PowerPC running AIX 4.1.5 and O_2 system version 4.6 with a CPU clock rate of 166 MHz, 1GB of main memory and a 9 GB disk. Only one processor was used since

the application was single-threaded. The test program was allowed a maximum of 128 MB. Swapping and buffering mechanisms are provided by the O_2 system. We ran our tests using the Census data which belong to the domain of statistical databases. Our objective is to classify the set of persons according to several characteristics such as sex, age, profession, etc. The Census data were extracted from the Kansas 1990 PUMS file (Public Use Microdata Samples). We took two datasets containing the first 100000 persons: c10d100k and c15d100k. Each person in c10d100k has 10 items (from the total of 281 items). Each person in 15d100k has 15 items (from the total of 309 items). We treat each person as a single transaction, where the items are the characteristics associated to persons.

3.1 Frequent Concept Lattice Construction

Although in the worst case, the size of a frequent concept lattice (i.e. the number of concepts it contains) can be exponential with respect to the number of database objects, its size is linear with respect to the number of database objects when there exists an upper bound k which is the average size (number of items) of a database object. In our experiments, k was set to respectively 10 and 15. Figure 6 shows the CPU time (including disk access) measured the total time (in second) necessary to build the lattice by adding new items one by one calling the lattice clustering algorithm (c.f. Figure 2.2). The important difference between the two curves can be explained by the difference in the size of the two resulting lattices. Indeed, the algorithm visits almost all concepts of the lattice and performs the intersection operation at each encountered concept. A more judicious implementation, without visiting every concept of the lattice, should accelerate the construction time.

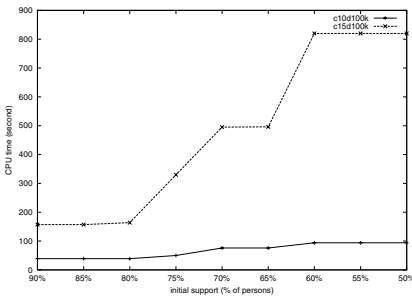


Fig. 6. CPU time of frequent concept lattice construction on c10d100k and c15d100k with different supports

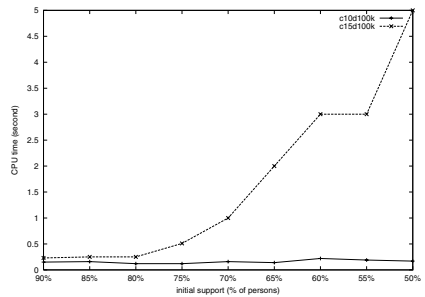


Fig. 7. CPU time of discovering association rules from the two lattices with different supports

3.2 Querying Strong Association Rules

At this step, the frequent concept lattice is constructed and stored in the O_2 database system, we can then use it to generate all frequent itemsets and then derive all association rules. Experiments were conducted on the two databases c10d100k and c15d100k using different minimum supports ranging from 50% to 90% to get meaningful response times. The minimum confidence is fixed to 75%. In Figure 7, one can see the running times of the experiments. Logically, we observe that the bigger the support is the shorter the association rule generation time.

4 Conclusion and Future Works

This paper proposes a framework to integrate data clustering and association rule discovery. The heart of the framework is the use of frequent concept lattice data structure during the process of knowledge discovery. Experimental results show that our method can generate pertinent concepts and strong association rules efficiently, since response time after lattice construction is measured almost in seconds. In our future work, we will focus on updating techniques for maintenance of the frequent concept lattice to handle dynamic databases. This approach avoids the repetitive process of discovering all frequent concepts from scratch each time a new object is introduced in the databases, and allows incremental data clustering and association rule discovery.

References

1. I. Bournaud and J.G. Ganascia. Conceptual clustering of complex objects: A generalization space based approach. In *3th Inter. Conf. on Conceptual Structures (ICCS'95)*, 1995. 437
2. C. Carpineto and G. Romano. A lattice conceptual clustering and its application to browsing retrieval. *Machine Learning*, (24):95–122, 1996. 437, 439
3. R. Godin and R. Missaoui. An incremental concept formation approach for learning from databases. *Theoretical Computer Science: Special Issue on Formal Methods in Databases and Software Engineering*, 133:387–419, 1994. 437
4. R. Godin, R. Missaoui, and H. Alaoui. Incremental Concept Formation Algorithms based on Galois (Concept) Lattices. *Computational Intelligence*, 11(2):246–267, 1995. 437, 439
5. T.B. Ho. An approach to concept formation based on formal concept analysis. *IEICE Trans. Information and Systems*, (5):553–559, 1995. 437, 439
6. R.E. Kent. Automatic classification. Technical report, Department of Computer and Information Science, University of Arkansas, 1995. 439
7. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Journal of Information Systems*, 24(1):25–46, 1999. 442
8. K. Waiyamai. *Knowledge Discovery and Management from very large Databases: A Frequent Concept Lattice Approach (In French)*. Ph.d. thesis, University of Clermont-Ferrand II, France, 1999. 439, 440, 442

9. K. Waiyamai, R. Taouil, and L. Lakhal. Querying concept lattices in object databases. In *3th Biennial World Conference on Integrated Design & Process Technology (IDPT'98)*, pages 280–289, Berlin, Germany, 1998. 440, 442
10. R. Wille. Restructuring lattices theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Dordrecht-Boston, 1982. 437
11. R. Wille. Concept lattices and conceptual knowledge systems. In *Computers Math. Applications*, volume 23, pages 493–515. 1992. 437, 439

Some Improvements on Event-Sequence Temporal Region Methods

Wei Zhang

The Boeing Company
P.O. Box 3707, MS 7L-66
Seattle, WA 98124-2207, USA

Abstract. Finding hidden temporal structures from event sequences is a difficult task, particularly when events occur irregularly over time and temporal dependencies may exist in a long time horizon. The tasks involved are not only to find event patterns represented in the form of temporal orders, but more importantly to find patterns that are described with precise *time conditions* and rules that can be applied to predict *when* a future event will occur. Recent study has shown that a new approach based on learning temporal regions is a good solution for this problem. This paper investigates this approach in a greater depth and makes several improvements. It introduces multiple rule selection methods to better uncover hidden relations. It also introduces heuristic rule pruning methods to speed up search to solve large-scale problems. Experimental results are presented which show the effectiveness of the new methods.

1 Introduction

Event sequence problems are ubiquitous in the real world. With increasing, massive amount of data being made available, solutions for such problems have become highly desired. This has led to many exciting recent developments ([Mannila *et al.*1995], [Agrawal and Srikant1995], [Srikant and Agrawal1996], [Oates *et al.*1997], [Howe and Somlo1997], and [Zhang1999]). Methods developed have been applied to a variety of problems such as telecommunication, sales transaction, and manufacturing.

Focusing on the generic event sequence problem where events are typically irregularly distributed over time, this paper investigates Zhang's (1999) temporal region approach in a great depth and makes several improvements to the existing methods. It introduces multiple rule selection methods to better uncover hidden relations. It also introduces heuristic rule pruning methods to speed up search to solve large-scale problems.

This paper is organized as follows. The next section reviews basic methods developed in the temporal region-based approach. The third section presents the methods developed for multiple rule selection and rule selection pruning. An algorithm incorporating these enhancements is also presented. After then, the paper presents an empirical study on the enhanced functionalities, followed by a conclusion pointing interesting future research.

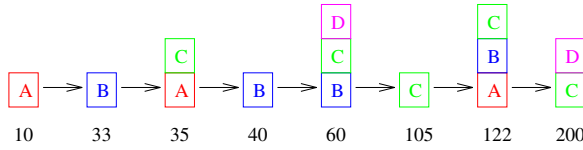


Fig. 1. An event sequence example

2 Background

2.1 The Problem

An event sequence is an ordered list of *objects* each represented with a time associated with a list of *events* that occur at the time. This means at any given time one or multiple events in different types may occur. Each event is of one (and only one) *type*. The problem is to find all significant correlations or dependency between different types of events. Such a relation could be *forward* that tells if an event of one type, say A, occurs now then what will be the chance an event of another type (could be the same type), say B, will occur in a future time, say in 5 minutes or between 5 and 10 minutes. A relation could also be *backward* that tells if an event of one type occurs now an event of another type must have occurred in a past time. When there is no time delay in a dependency of two events, we often call such a relation an *association* ([Agrawal et al.1996]). Sometimes there could be a mutual effect between two types of events such that one leads to the other and vice versa. Such a dependency provides a stronger association between two types of events.

The generic problem typically has the following features: The events are sparse over time, and they are irregularly distributed. Figure 1 shows a simple example how such an event sequence may present. The example contains four types of events: A, B, C, and D. Sometimes events may occur in a quite adjacent time (e.g., at time step 33 and 35). Sometimes there could be no event occurrence in a long period of time (e.g., no event between 123 and 199).

2.2 Minimal Temporal Regions

There are two basic ideas applied in the temporal region-based approach. One is rely on input event data to first develop all potential strong event correlations in the notion of *minimal temporal regions* as the hypothesis. The second is apply a set of evaluation criteria to test these hypothesis to select those of most significant correlations. Let us first look at minimal temporal regions.

A *temporal region rule* defines a temporal region condition for a *target* event type E_T , where a condition is represented in the form of a condition event type E_C associated with a period of time $[a, b]$ ($a, b \in \mathbb{R}; 0 \leq a \leq b$). This rule can be written as

$$E_C[a, b] \Rightarrow E_T,$$

which says two things: first, if an event of E_C occurs now then there will be *at least* one event of E_T to occur between future a and b time scope, and second, if an event of E_T occurs now then there must be *at least* one event of E_C occurred between past a and b time scope.

While there is infinite number of temporal regions that can be defined for a pair of events, our idea is to only look at rules with *minimal temporal regions* with respect to a given data set. Let S be a given sequence of events where each event records two pieces of information, the type of the event and the time this event occurs. For all pair of events of types E_C and E_T where the E_C event occurs no later than the E_T event, we can compute the *lag* between them and obtain a *set of lag values* S_{lag} . For any subset s' of S_{lag} , we can define its *minimal temporal region*, which is the smallest time interval that covers all the values in the subset, or $[min(s'), max(s')]$. The complete set of minimal temporal regions for $E_C \Rightarrow E_T$ for a given sequence is comprised of all different minimal temporal regions. Therefore, there are a total of $\binom{m+1}{2}$ minimal temporal regions for a lag set S_{lag} of size m ($m = |S_{lag}|$).

For example, for the sequence in Figure 1, the lag set for rule $C \Rightarrow A$ is $\{0, 17, 62, 87\}$. This results in 10 minimal temporal regions: $[0,0]$, $[0,17]$, $[0,62]$, $[0,87]$, $[17,17]$, $[17,62]$, $[17,87]$, $[62,62]$, $[62,87]$, and $[87,87]$.

2.3 Metrics

Six metrics have been developed for assessing temporal region rules.

1. **Prediction Accuracy (ACCP):** This computes the percentage of cases that a target event occurs in the time region over all cases that a condition event occurs.

2. **Recall Accuracy (ACCR):** This computes the same metric in the opposite temporal direction. It is the percentage of cases that a condition event occurred in the time region earlier over all cases that a target event occurs.

3. **Prediction Bonus (BNSP):** Sometimes target events may occur multiple times in a given time region. This metric provides a score for additional occurrences of target events. Let $FwdCnt$ be the number of cases satisfying a rule (condition-target event pair) while each condition event occurrence is only allowed to be counted at most once (this is the count used in ACCR computation), and let $AllCnt$ be the count including all cases satisfying the rule where a condition event occurrence may be counted multiple times because of multiple target event occurrences. We compute the bonus as $1 - FwdCnt/AllCnt$.

4. **Recall Bonus (BNSR):** Similarly, we define $BwdCnt$ by counting at most once for each target event occurrence. This bonus is defined as $1 - BwdCnt/AllCnt$.

5. **Range (RNG):** While both ACCP and ACCR reward larger regions (their values increase monotonously as the size of a temporal region grows), it is important to have some metric encouraging smaller regions. RNG is the one. The BESTREGIONRULES algorithm (see the updated version later) lets users define a lag scope in searching temporal relations, which is specified by a minimal lag $MinLag$

and a maximal lag **MaxLag**. RNG is defined as $1 - \text{Intv}(r) / (\text{MaxLag} - \text{MinLag} + 1)$, where $\text{INTV}(r)$ is the region size of rule r .

6. **Coverage (COV)**: This metric computes the rate of cases covered by a rule over all cases that are covered by the same condition-target pair but with the full search scope defined by **MinLag** and **MaxLag**. We denote the latter as *AllCntScp*. Then COV is $\text{AllCnt} / \text{AllCntScp}$.

Briefly, both ACCP and BNSP assess the predicting power of events. On the other hand, both ACCR and BNSR are designed for causal analysis and diagnosis, finding reasons on event occurrences. RNG narrows down temporal regions, which is important for finding key structures of data. While sometimes this parameter may be slightly tricky to apply, fortunately, accordingly to previous study, there is often a wide range of selections available for achieving similar, good results [Zhang1999]. COV is designed for finding regions with large coverage of cases over all cases in the search scope.

3 Rule Selection Methods

3.1 Multiple Rules

The early BESTREGIONRULES algorithm has a limitation that only one region rule (the best one) may be returned for a pair of event types. To enable finding more complete set of significant relations, we add the multiple rules functionality.

The approach developed here is comprised of two steps. The first is to segment the temporal region space of a rule into several portions. After segmentation is completed, the next step is to select rules from the segmented space. We select one best rule for each segment. We also select rules across segment boundaries. This ensures the system be able to find correlations with a large lag range.

Two questions need to be answered in segmentation. First, how many segments should be selected? And second, how should we determine segments and segment boundaries? In general, answers to the first question depend on application problems. A nice feature of our approach is that since we select rules across segment boundaries, the number of segments does not affect the results on the best rules.

For the second question, several simple segmentation methods can be applied.

- **Uniform segmentation.** The simplest method is to segment a lag space uniformly. A uniform segmentation divides a lag space in the scope of **MinLag** and **MaxLag** into a number of equal-sized segments.
- **Clustering.** We can also apply clustering methods like K-MEAN to segment a space. This may improve selection of rules but certainly adds some computational cost.

A problem one needs to keep in mind when applying more sophisticated segmentation methods is that there are three different counting methods applied here: *Cnt*, *FwdCnt*, and *BwdCnt*. While *Cnt* sums over all number of the cases

covered under a region, both *FwdCnt* and *BwdCnt* apply a different aggregation operator—the set **Union** operation—to avoid multiple counting of an event occurrence. Segmentation using different counting methods may come up with different results. Applying which type of methods should be determined based on whether an analysis is for discovery of a forward model or a backward model and whether multiple occurrences are important.

After segmentation is finished, selection of multiple rules is trivial. We apply the following simple procedure. For each of k segments we have determined, we select the best rule in terms of the combined metric value *score*, which is a weighted sum of the six metric values. We also select a best rule across each of $k - 1$ segmentation boundaries. For the i th boundary, a pre-condition event must occur in segment i and a target event must occur in segment $i + 1$ or later. Furthermore, there is another parameter in the rule selection procedure. The **Minimal Score** parameter specifies that only rules with scores larger than or equal to this value are selected. Therefore, the rule selection procedure may return a maximum of $2k - 1$ rules for each pair of event types.

3.2 Heuristic Pruning

The standard rule selection procedure tests over all minimal temporal regions to find a set of best region rules for each event pair. When an event sequence is very long, the number of cases under an event pair can become very large. This may result in a large set of distinctive lag values. In this situation, the minimal temporal region method no longer becomes efficient. Test over all $\binom{m+1}{2}$ minimal temporal regions is costly when the number of lag values m is very large.

To make rule selection more efficient, we introduce another parameter δ to define the *granularity* of the rule selection process. The following describes the method based on **Integer** typed lag space (It is not difficult to extend the method to the $\mathbb{R}$ domain). The δ parameter defines the maximal number of lag steps that can be *skipped* in forming minimal temporal regions. When $\delta = 1$, we do not skip any lag step, so we will test all minimal temporal regions. When $\delta > 1$, lag values may be pruned. For any lag value, if the difference between its next larger lag value and its next smaller lag value is larger than δ , then this lag value must be selected. Otherwise, selection of this lag value is *optional*.

Specifically, let us look at a simple example. Let $\{0, 1, 2, 3, 4, 5, 6, 15, 25\}$ be a given lag set and let $\delta = 5$. We start the selection process from lag 0. First, we have to select 0 because there is no lag value smaller than 0. After then we consider lag 1. Selection of 1 is optional because the difference between its next larger lag 2 and previous selected lag value 0 is 2, smaller than 5. Likewise, selection of 2, 3, or 4 is all optional. Let us suppose we do not select all these. Then next we consider lag 5. At this time we have to select it because if 5 is not selected then this will cause a skip of the lag space with 6 time steps. After 5 is selected, next we have to select 6 because the difference between the next larger value 15 and the previous selected value 5 is 10, larger than 5. We can see that the selection process just went through a cluster and both boundaries of the cluster, 0 and 6, are selected. Following this, we consider 15 and 25 subsequently and

we have to choose them. Now we have obtained a pruned lag set $\{0,5,6,15,25\}$. This reduces future rule test from $\binom{10}{2}$ tests to $\binom{6}{2}$.

We develop a simple heuristic to determine whether or not to skip a lag when its selection is optional. We compare the number of cases under the current lag c_t with the number of cases under the previously selected lag c_{t-1} . If $c_t > 1.5 * c_{t-1} + 2$, then we select the current lag. Otherwise we skip it. When the number of cases under the current lag is significantly larger than the previous one, it makes sense to select this one because a temporal region either starting or ending at this point is likely to provide a better score.

Table 1. The Updated Best Region-Rules Algorithm

```

procedure BESTREGIONRULES( $S, M_1, M_2, W, v, k, \delta$ )
inputs:    $S$                 // Event sequence
            $M_1$              // Minimal Lag, default = 0
            $M_2$              // Maximal Lag
            $W$                // Weights for the score function
            $v$                // Minimal Score
            $k$                // The number of segments
            $\delta$             // The pruning factor
if ( $S$  not empty) PUSH( $L, \text{POP}(S)$ ) // Move first element of  $S$  to list  $L$ 
while ( $S$  not empty) do
   $e := \text{POP}(S)$ 
  for all  $e' \in L$ 
    let  $t$  be time of  $e$  and  $t'$  be time of  $e'$ 
    let  $d = t - t'$ 
    if ( $M_1 \leq d \leq M_2$ )
      let  $E$  and  $E'$  be the type of events  $e$  and  $e'$ 
      ADDLAG( $E, E', d, t'$ )
      if ( $d = 0$ ) ADDLAG( $E', E, d, t'$ )
    PUSH( $L, e$ )
end while
SELECTRULES( $W, v, k, M_1, M_2, \delta$ )
end procedure

procedure ADDLAG( $E_{tar}, E_{cond}, d, t$ )
inputs:    $E_{tar}$           // A target event type
            $E_{cond}$          // A condition event type
            $d$               // Lag between the two events
            $t$               // Start time, used as the index to represent a case
   $r := \text{FINDRULE}(E_{tar}, E_{cond}, R)$  //  $R$  maintains a list of rules
  if ( $r = \text{NIL}$ )           // if this rule does not exist
     $r := \text{MAKERULE}(E_{tar}, E_{cond})$ 
    ADDRULE( $r, R$ )         // add the rule into  $R$ 
  SUBADDLAG( $d, t, r$ )      // add the lag and index into  $r$ 
end procedure

procedure SELECTRULES( $W, v, k, M_1, M_2, \delta$ )
for all  $r \in R$ 
   $g := \text{SEGMENT}(k, M_1, M_2, r)$  //  $g$  defines segment boundaries
   $l := \text{PRUNELAGSET}(r, \delta)$      //  $l$  is a pruned lag set
  for ( $i := 1, i \leq 2k - 1, i++$ )  $\text{best}[i] := \text{NIL}$  // Initialize best
  for all  $c \in$  pair-wise combinations of  $l$  //  $c$  develops a region
     $i := \text{SEGMENTID}(c, g)$  // determine the segment of  $c$ 
    if ( $\text{SCORE}(c, W) \geq v$  and  $\text{SCORE}(c, W) \geq \text{SCORE}(\text{best}[i], W)$ )  $\text{best}[i] := c$ 
  for ( $i := 1, i \leq 2k - 1, i++$ ) PRINTREGIONRULE( $r, \text{best}[i]$ )
end procedure

```

3.3 Algorithm

Table 1 shows the updated BESTREGIONRULES algorithm that includes new rule selection methods. Besides taking parameters S, M_1, M_2, W , and v , the updated BESTREGIONRULES procedure also takes the segmentation parameter k and pruning parameter δ . In SELECTRULES, SEGMENT call is in the loop for all rule in the complete rule set R . This step can be moved to the outside of the loop if uniform segmentation is applied. PRUNELAGSET computes a set of pruned lags. After then, temporal regions are generated by pair-wise combination of the pruned lag values and then evaluated. For each temporal region, the evaluation step first determines its segment and then computes its score—the weighted sum of the six metric values using weights W . Finally, the best evaluated region for each segment is selected if its score is larger or equal to the Minimal Score parameter.

4 Experimental Study

We conducted a series of experiments to observe the behavior of the new methods. These experiments used the same data sets as in the [Zhang1999] work: event sequences generated based on 10 M15–18 models (15 event types and 18 *direct* temporal relations), 10 M30–36, and 10 M60–72. Figure 2 shows one model in the M15–18 set. Among 18 direct temporal relations, three are associations ($A_4 \Rightarrow B_2$, $B_2 \Rightarrow B_5$, and $B_5 \Rightarrow C_2$). The rest all is involved with a time lag, described with lag region $[a, b]$. Both types of relations are associated with a probability to represent the dynamics of event occurrences.

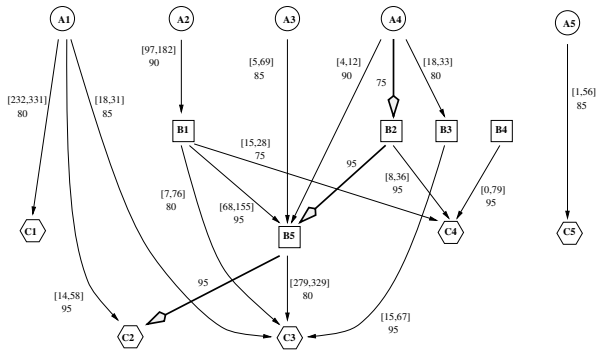


Fig. 2. An example of forward temporal models

4.1 Model Discovery

Again, this paper presents the results on forward model discovery (Similar results have been obtained on learning backward models). For forward model discovery, we do not need to apply the metrics on recall. We set weight 0 for ACCR and BNSR and 1 for all other four metrics. As discussed earlier, the only slightly tricky parameter is RNG. We did some simple experiment that found setting it at 1 is reasonably good.

Also, we set the minimal-score parameter 140, the minimal lag 0 and the maximal lag 500. We set the number of segments 4. This may give a maximum of 7 region rules for each event pair. δ is set at 1 in this experiment, so no pruning is performed. For all model in the three sets, simulation was conducted with a total of 8000 events. To observe the behavior of learning, we extract learned rules when simulation is finished with 1000, 2000, 4000, and finally 8000 events respectively.

Table 2. This table gives detailed information in comparison of the learned models and the underlying models

<i>Data Set</i>		Match	Near Match	Region Overlap	Term Match	Unmatch	Fitness Score
M15-18 Set	Min	11.00	0.00	0.00	0.0	0.0	0.8194
	1st Qu	14.75	0.75	0.00	0.0	0.0	0.9132
	Median	17.00	1.00	1.00	0.0	0.0	0.9583
	Mean	15.70	1.30	1.00	0.0	0.0	0.9403
	3rd Qu	17.00	1.25	1.25	0.0	0.0	0.9861
	Max	18.00	4.00	3.00	0.0	0.0	1.0000
M30-36 Set	Min	29.00	0.00	0.00	0.0	0.0	0.9306
	1st Qu	32.00	1.00	0.00	0.0	0.0	0.9514
	Median	34.00	2.00	0.00	0.0	0.0	0.9757
	Mean	33.05	2.30	0.55	0.0	0.1	0.9698
	3rd Qu	34.00	3.25	1.00	0.0	0.0	0.9861
	Max	36.00	7.00	2.00	0.0	1.0	1.0000
M60-72 Set	Min	56.00	2.00	0.0	0.0	0.0	0.9306
	1st Qu	59.75	6.75	0.0	0.0	0.0	0.9514
	Median	61.50	8.50	0.0	0.0	0.0	0.9566
	Mean	62.35	8.95	0.5	0.1	0.1	0.9611
	3rd Qu	65.00	12.25	1.0	0.0	0.0	0.9757
	Max	70.00	15.00	2.0	1.0	1.0	0.9931

After learning is completed, we compared the learned models with the underlying models. For all direct temporal relation in a underlying model, we find rules in the corresponding learned model with the same names of condition event and target event. If we find a rule that has at least 80% region overlapping with this relation and vice versa (80% region of the relation is also in the rule) and the prediction accuracy of the learned rule is close to the probability of the underlying rule with difference no bigger than 0.1, then we say the two rules *match*. Otherwise, if they both have at least 50% region overlapping with each other and the probability difference is no bigger than 0.25, we say they are “*near-match*”.

Otherwise, we say the two rules “*region-overlap*” if there is at least at one point covered by the both rules. If we do not find rules that overlap with this relation then we say there is a “*term-match*” that describes a correct event temporal order. If we do not find a rule with the same condition and target names in the learned model, then we say this relation in the underlying model is *unmatch* to the learned model.

After this assessment, we computed a *fitness* function for each underlying model in comparing to its learned model. For each relation in a underlying model, we gave 1 point for a match, 0.75 point for a near-match, 0.25 point for a region-overlap, 0.1 point for a term-match, and 0 for unmatch. The fitness score is the sum of all the points divided by the number of relations in the underlying model.

Table 2 shows the performance assessed by these six measures. We separate problems from three different sets. Each set shows six statistics including **Min**, **Mean**, and **Max** on the six measures. The table concludes that learning was well performed on all three sets of problems. These statistics are made on learned rules at 4000 and 8000 event simulations. The match rate is high, in average it is about 87%. For the rest of the relations, most are near-match. The ratios for region-overlap, term-match, and unmatch are almost 0.

4.2 One Rule vs. Multiple Rules

The next experiment compares the multiple rules method (**MultiBest**) versus one using the single rule method (**OneBest**). Both procedures use the same parameter settings as described earlier. Figure 3 compares how they perform in terms of their fitness scores over simulation. It plots the median statistic plus the errorbars using the 1st Qu and 3rd Qu values. These statistics are computed over all three sets of the problems. We can see that clearly **MultiBest** outperforms **OneBest** in uncovering underlying models.

While **MultiBest** finds rules in underlying models more accurately, it also comes up with a larger set of other rules. Figure 4 compares all rules found by both procedures. Here, we use **One** and **Multi** to represent **OneBest** and **MultiBest** respectively. **True** represents the underlying model. We plot statistics on three sets of the problems separately. For **True** we plot the average number of direct relations and the average number of *transitive* relations (Transitive relations are those that can be derived from directed relations, e.g., $A \Rightarrow C$ is the transitive relation derived from $A \Rightarrow B$ and $B \Rightarrow C$). For **One** and **Multi**, we plot the average number of rules matching direct relations, transitive relations, both types of relations, and none of the relations respectively, which are denoted as **Direct**, **Transit**, **Both**, and **Others** respectively. When a rule belongs to **Both**, it is not counted in either **Direct** or **Transit**. Here, either **Match** or **Near-Match** is considered as a match.

The plot gives a general summary about the discovered rules. **Multi** performs better than **One** not only on direct relations, but more substantially on transitive relations. For transitive rules, since the implementation is set to attempt to find temporal relations in the time scope between 0 and 500 and with range under

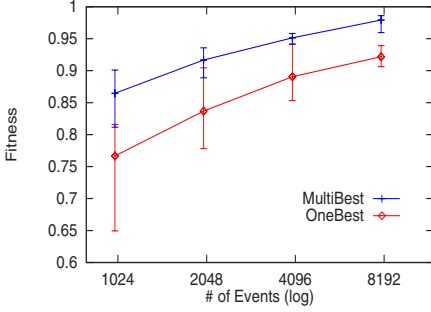


Fig. 3. MultiBest outperforms OneBest in uncovering underlying models

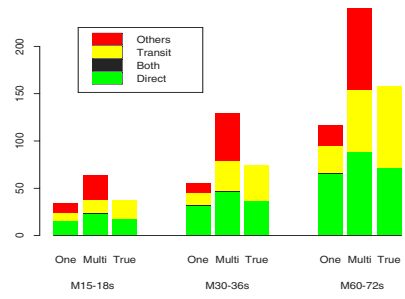


Fig. 4. A summarized comparison of the two procedures

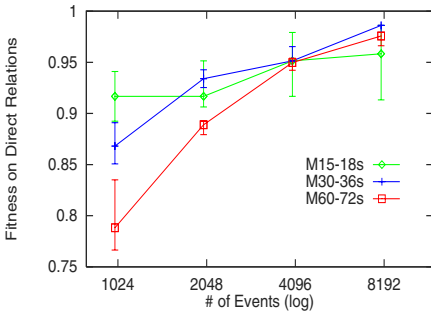


Fig. 5. Performance on finding direct relations

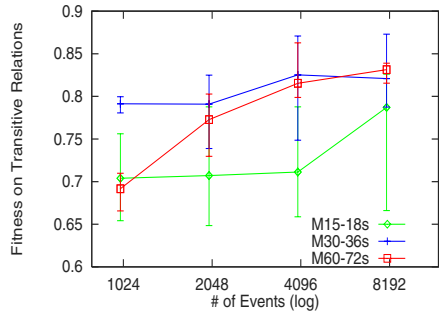


Fig. 6. Performance on finding transitive relations

100 (range is controlled by the RNG parameter), transitive relations beyond the scope are not selected. This is why the number of rules found for this part is smaller than the one in the underlying model. It is good to see that the number of rules that Multi selected is not substantial. Besides, the number of rules belonging to Both is almost 0, which means that basically direct relations and transitive relations are quite different and the algorithm can work properly to find distinctive rules for different relations.

We also observed how the multiple rules procedure performs during simulation processes. Figure 5 and Figure 6 shows the performance on both direct relations and transitive relations respectively. Again, they plots the median, 1st Qu, and 3rd Qu statistics. We compare the performance with respect to the size of models. Intuitively, a big model needs longer simulation. The plots in general suggests that for problems of size M15-18 the most gain can be obtained by simulation to about 1000 events. The median fitness score on direct relations for M15-18 reaches 0.92 at 1000 events. But for larger-sized problems in M30-36 and M60-72, it is worth to train up to 8000 events. For these problems, the performance gain is substantial for simulation from 1000 events to 8000 events.

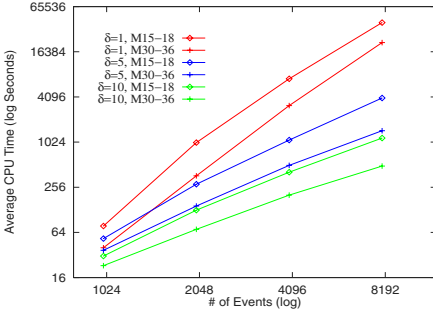


Fig. 7. CPU Time compared on with and without pruning

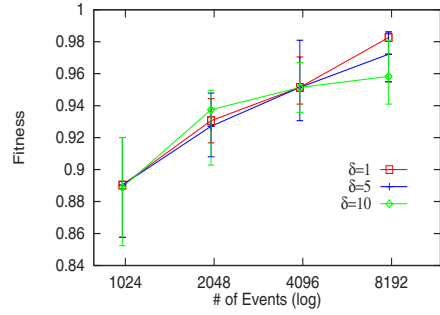


Fig. 8. With pruning the change on fitness scores is small

4.3 Speed Up

Now let us examine how the pruning factor δ affects the performance on model discovery and search time. Without surprising, we found that for the same amount of event data, in general, learning on fewer event types takes more time than on more event types. This is because with fewer event types, more cases and lags are developed for a pair of event types, thus much more temporal regions need to be evaluated. Our experiments found that on average for the same amount of data, learning for a M15-18 problem takes about twice as much time as learning for a M30-36 problem takes. This provides another reason to discourage longer simulation on small models. Similarly, a M30-36 problem simulation needs about nearly twice as much time as a M60-72 simulation with same number of events.

Accordingly, experiments for testing the δ parameter focus on smaller problems. A large model in M60-72 probably may not generate many lags in a event pair so pruning may not be crucial. However, this does not mean for a real problem with many event types (say over several hundred types) pruning is not useful. A particular manufacturing problem in Boeing involves more than 300 types of events, but about 40% of event occurrences fall into a small set of 32 event types. In this case, we found pruning is very helpful for quick and scale-up analysis.

Figure 7 and Figure 8 plots the performance on both M15-18 and M30-36 problems. They compares learning without pruning ($\delta = 1$) and with pruning on two settings, $\delta = 5$ and $\delta = 10$. First, let us look at the time needed for each of the settings. Clearly, time reduction with pruning is substantial. For simulation with 8000 events on a M15-18 model, on average it takes 40180 seconds for without pruning, but 3963 seconds with pruning on $\delta = 5$ and 1159 seconds with pruning on $\delta = 10$.

Let us now look at the performance on model discovery. We employ median, 1st Qu, and 3rd Qu on the direct-relation fitness for all problems in both sets. We can see that the performance on the three different processes is very close. Pruning does not affect the performance up to 4000 event simulation. $\delta = 5$ and

$\delta = 10$ both reach high fitness scores at about 0.95 (median) at 4000, which is the same as the un-pruned process. At 8000 events, we see $\delta = 10$ can not continue to improve the performance well, while $\delta = 5$ still keeps close to $\delta = 1$.

The plots showed clear evidence that pruning is very effective on both M15-18 and M30-36 problems. In conclusion, when a data set is reasonably large with respect to the number of event types, pruning is always suggested. The pruning mechanism allows this temporal region method to scale up to mine over a large amount of data.

5 Concluding Remarks

This paper investigates region-based methods for discovering temporal structures in data in a greater depth. It introduces rule selection methods to allow selection of multiple rules for more complete uncovering of hidden relations. It also introduces pruning methods for quick and scale-up analysis.

In particular, this paper showed how close direct and transitive relations can be found using the updated BESTREGIONRULES procedure. Indeed, this procedure also discovers other relations which are neither direct nor transitive—They are *parallel* in the general sense. While all significant relations can be found under a general pool, extracting direct relations among them still remains in big challenges. In real-world applications, such analysis often relies on extensive domain knowledge. Investigating how domain knowledge should be applied and how such analysis may be formalized is a very interesting research issue.

References

- Agrawal and Srikant1995. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*. IEEE Press, 1995. 446
- Agrawal *et al.*1996. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT, 1996. 447
- Howe and Somlo1997. A. Howe and G. Somlo. Modeling discrete event sequences as state transition diagrams. In *Proceedings of the Second Conference on Intelligent Data Analysis*, 1997. 446
- Mannila *et al.*1995. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episode in sequences. In *Proceedings of the First International Conference of Knowledge Discovery in Databases and Data Mining*. AAAI Press, 1995. 446
- Oates *et al.*1997. T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen. A family of algorithms for finding temporal structure in data. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, 1997. 446
- Srikant and Agrawal1996. R. Srikant and R. Agrawal. Mining sequential patterns: generalizations and performance improvements. In *Proceedings of the Fifth International Conference on Extending Database Technology*, 1996. 446

Zhang1999. W. Zhang. A region-based learning approach to discovering temporal structures in data. In *Proceedings of the Sixteenth International Conference on Machine Learning*. Morgan Kaufmann, 1999. [446](#), [449](#), [452](#)

Author Index

- Aimeur, Esma 23
Angulo, Cecilio 31
- Béjar, Javier 47
Badea, Liviu 39
Blanzieri, E. 55
Boley, Daniel 263
Bréhélin, Laurent 75
Bratko, Ivan 382
Brazdil, Pavel B. 63
Brijs, Tom 88
Bryant, Christopher H. 300
Bueno, Rafael Morales 280
- Caraux, Gilles 75
Carney, John 109
Català, Andreu 31
Chidlovskii, Boris 96
Costa, Joaquim Pinto da 426
Cunningham, Pádraig 109
- Dahl, Fredrik A. 117, 171
Domingos, Pedro 3
- Eccher, C. 55
Edwards, Peter 331
Escudero, Gerard 129
- Fan, Wei 142
Forti, S. 55
- Gascuel, Olivier 75
Gelbrich, Katja 154
Geurts, Pierre 162
- Halck, Ole Martin 117, 171
Herbst, Joachim 183
- Izworski, Andrzej 195
- Kalnishkan, Yuri 203
Kim, DaeEun 211
- López, Llanos Mora 280
Lakhal, Lotfi 437
Langley, Pat 220
Lanquillon, Carsten 229
Latourrette, Mathieu 238
Lee, Jaeho 211
Lee, Wenke 142
Lu, Hongjun 271
Ludl, Marcus-Christopher 246
- Marinchev, Ivo 254
Markov, Zdravko 254
Màrquez, Lluís 129
Martienne, Emmanuelle 292
McKenna, Elizabeth 357
McKoskey, David 263
Meretakis, Dimitris 271
Miller, Matthew 142
Morik, Katharina 4
Morin, Emmanuel 292
Muggleton, Stephen H. 300
- Núñez, Marlon 323
Nakhaeizadeh, Reza 154
Neri, Filippo 313
- Paslawski, Andrzej 195
Payne, Terry R. 331

- Ragetli, Jon 96
 Rigau, German 129
 de Rijke, Maarten 96
 Ruiz, Inmaculada Fortes 280
 Ruiz, Francisco Triguero 280

 Sboner, A. 55
 Scheffer, Tobias 344
 Smyth, Barry 357
 Soares, Carlos 63
 Srinivasan, Ashwin 300
 Stolfo, Salvatore J. 142
 Stone, Peter 369
 Stromsten, Sean 220
 Šuc, Dorian 382
 Swinnen, Gilbert 88

 Tadeusiewicz, Ryszard 195
 Talavera, Luis 392

 Tamon, Christino 404
 Ting, Kai Ming 413
 Torgo, Luís 426

 Vézeau, Mathieu 23
 Vanhoof, Koen 88
 Veloso, Manuela 369

 Waiyamai, Kitsana 437
 Wehenkel, Louis 162
 Wets, Geert 88
 Widmer, Gerhard 246
 Wüthrich, Beat 271

 Xiang, Jie 404

 Zhang, Wei 446