Fabio Roli
Josef Kittler
Terry Windeatt (Eds.)

LNCS 3077

Multiple Classifier Systems

5th International Workshop, MCS 2004
Cagliari, Italy, June 2004
Proceedings

Springer

Lecture Notes in Computer Science 3077

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Fabio Roli Josef Kittler
Terry Windeatt (Eds.)

Multiple Classifier Systems

5th International Workshop, MCS 2004
Cagliari, Italy, June 9-11, 2004
Proceedings

Springer

eBook ISBN: 3-540-25966-X
Print ISBN: 3-540-22144-1

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

The fusion of different information sources is a persistent and intriguing issue. It has been addressed for centuries in various disciplines, including political science, probability and statistics, system reliability assessment, computer science, and distributed detection in communications. Early seminal work on fusion was carried out by pioneers such as Laplace and von Neumann. More recently, research activities in information fusion have focused on pattern recognition. During the 1990s, classifier fusion schemes, especially at the so-called decision-level, emerged under a plethora of different names in various scientific communities, including machine learning, neural networks, pattern recognition, and statistics. The different nomenclatures introduced by these communities reflected their different perspectives and cultural backgrounds as well as the absence of common forums and the poor dissemination of the most important results.

In 1999, the first workshop on multiple classifier systems was organized with the main goal of creating a common international forum to promote the dissemination of the results achieved in the diverse communities and the adoption of a common terminology, thus giving the different perspectives and cultural backgrounds some concrete added value. After five meetings of this workshop, there is strong evidence that significant steps have been made towards this goal. Researchers from these diverse communities successfully participated in the workshops, and world experts presented surveys of the state of the art from the perspectives of their communities to aid cross-fertilization. The term multiple classifier systems currently appears in the list of topics of several international conferences, these workshop proceedings are often cited in journal and conference papers, and tutorials on multiple classifier systems have been given during relevant international conferences such as Information Fusion 2002. Last, but not least, in the pattern recognition community, the term multiple classifier systems has been adopted as the main reference to this subject area.

Following its four predecessors published by Springer-Verlag, this volume contains the proceedings of the 5th International Workshop on Multiple Classifier Systems (MCS 2004), held at Tanka Village, Cagliari, Italy, on June 9–11, 2004. Thirty-five papers out of the 50 submitted from researchers in diverse research communities were selected by the scientific committee, and they were organized into sessions dealing with bagging and boosting, combination and design methodologies, analysis and performance evaluation, and applications. The workshop program and this volume were enriched by two invited talks given by Ludmila I. Kuncheva (University of Wales, Bangor, UK), and Nageswara S.V. Rao (Oak Ridge National Laboratory, USA).

This workshop was supported by the Department of Electrical and Electronic Engineering of the University of Cagliari, Italy, the University of Surrey, Guildford, UK, and Gruppo Vitrociset. Their support is gratefully acknowledged. We also thank the International Association for Pattern Recognition and its Techni-

cal Committee TC1 on Statistical Pattern Recognition Techniques for sponsoring MCS 2004.

We wish to express our appreciation to all those who helped to organize MCS 2004. First of all, we would like to thank all the members of the Scientific Committee whose professionalism was instrumental in creating a very interesting scientific program. Special thanks are due to the members of the Organizing Committee, Giorgio Giacinto, Giorgio Fumera, and Gian Luca Marcialis, for their indispensable contributions to the MCS 2004 Web site management, local organization, and proceedings preparation.

June 2004 Fabio Roli, Josef Kittler, Terry Windeatt

Workshop Chairs

F. Roli (University of Cagliari, Italy)
J. Kittler (University of Surrey, UK)
T. Windeatt (University of Surrey, UK)

Scientific Committee

J.A. Benediktsson (Iceland)
H. Bunke (Switzerland)
L.P. Cordella (Italy)
B.V. Dasarathy (USA)
R.P.W. Duin (The Netherlands)
C. Furlanello (Italy)
J. Ghosh (USA)
V. Govindaraju (USA)
T.K. Ho (USA)
S. Impedovo (Italy)
N. Intrator (Israel)
A.K. Jain (USA)
M. Kamel (Canada)
L.I. Kuncheva (UK)
D. Partridge (UK)
A.J.C. Sharkey (UK)
C.Y. Suen (Canada)
K. Tumer (USA)
G. Vernazza (Italy)

Organizing Committee

G. Giacinto (University of Cagliari, Italy)
G. Fumera (University of Cagliari, Italy)
G.L. Marcialis (University of Cagliari, Italy)

Organizers

Dept. of Electrical and Electronic Engineering of the University of Cagliari
Centre for Vision, Speech and Signal Processing of the University of Surrey

Sponsors

Dept. of Electrical and Electronic Engineering of the University of Cagliari
Centre for Vision, Speech and Signal Processing of the University of Surrey
International Association for Pattern Recognition
IAPR Technical Committee TC1

Supporters

Dept. of Electrical and Electronic Engineering of the University of Cagliari
Centre for Vision, Speech and Signal Processing of the University of Surrey
Gruppo Vitrociset

This page intentionally left blank

Table of Contents

Invited Papers

Bagging and Boosting

Combination Methods

Design Methods

Performance Analysis

Applications

Classifier Ensembles for Changing Environments

Ludmila I. Kuncheva

School of Informatics, University of Wales, Bangor
Bangor, Gwynedd, LL57 1UT, United Kingdom
l.i.kuncheva@bangor.ac.uk

Abstract. We consider strategies for building classifier ensembles for non-stationary environments where the classification task changes during the operation of the ensemble. Individual classifier models capable of online learning are reviewed. The concept of "forgetting" is discussed. Online ensembles and strategies suitable for changing environments are summarized.

Keywords: classifier ensembles, online ensembles, incremental learning, non-stationary environments, concept drift.

1 Introduction

> *"All things flow, everything runs, as the waters of a river, which seem to be the same but in reality are never the same, as they are in a state of continuous flow."*
>
> The doctrine of Heraclitus.

Most of the current research in multiple classifier systems is devoted to static environments. We assume that the classification problem is fixed and we are presented with a data set, large or small, on which to design a classifier. The solutions to the static task have marvelled over the years to such a perfection that the dominance between the classification methods is resolved by a fraction of percent of the classification accuracy. Everything that exists changes with time and so will the classification problem. The changes could be minor fluctuations of the underlying probability distributions, steady trends, random or systematic, rapid substitution of one classification task with another and so on.

A classifier (individual or an ensemble)[1], if intended for a real application, should be equipped with a mechanism to adapt to the changes in the environment. Various solutions to this problems have been proposed over the years. Here we try to give a systematic perspective on the problem and the current solutions, and outline new research avenues.

The paper is organized as follows. Section 2 sets the scene by introducing the concept of changing environment. Online classifier models are presented in Section 3. Section 4 details some ensemble strategies for changing environments.

[1] Unless specified otherwise, a *classifier* is any mapping $D : \Re^n \rightarrow \Omega$ from the feature space, $\Re^n$, to the set of class labels $\Omega = \{\omega_1, \ldots, \omega_c\}$.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 1–15, 2004.

2 The Changing Environment

Changing environments pose the main hurdle in many applications. One of the most acute examples is detecting and filtering out spam e-mail[2]. The descriptions of the two classes of e-mail, "spam" and "non-spam", evolve with time; they are user-specific, and user preferences may also evolve with time. Besides, the important discriminating variables used at time t to classify spam may be irrelevant at a future moment $t+k$. To make matters even worse, the variability of the environment in this scenario is hardly due to chance. The classifier will have to face an active opponent – the "spammers" themselves – who will keep coming up with ingenious solutions to trick the classifier into labeling a spam e-mail as legitimate.

2.1 Concept Drift and Types of Changes

Viewed in a probabilistic sense, a classification problem may change due to the changes in [13]

- Prior probabilities for the c classes, $P(\omega_1), \ldots, P(\omega_c)$;
- Class-conditional probability distributions, $p(\mathbf{x}|\omega_i)$, $i = 1, \ldots, c$; or
- Posterior probabilities $P(\omega_i|\mathbf{x})$, $i = 1, \ldots, c$.

Not every change in the distribution is going to degrade the performance of a classifier. Consider the minimum-error classifier which labels $\mathbf{x}$ as the class index of the largest posterior probability $P(\omega_i|\mathbf{x})$. If the largest posterior probability for every $\mathbf{x} \in \Re^n$ keeps its class index, then the decision of the classifier for this $\mathbf{x}$ will guarantee the minimum error no matter what the changes in the distributions are. Kelly et al. [13] call the changes in the probabilities *population drift* and remark that a notion of *concept drift* used in machine learning literature is the more general one.

While in a natural system we can expect gradual drifts (e.g., seasonal, demographic, habitual, etc.), sometimes the class description may change rapidly due to *hidden contexts.* Such contexts may be, for example, illumination in image recognition and accents in speech recognition [24]. The context might instantly become highly relevant. Consider a system trained on images with similar illumination. If images with the same type of content but a different illumination are fed to the system, the class descriptions might change so as to make the system worse than a random guess. The type of changes can be roughly summarized as follows

- Random noise [1, 23]
- Random trends (gradual changes) [13]
- Random substitutions (abrupt changes) [23]
- Systematic trends ("recurring contexts") [23]

[2] The term SPAM is coined to denote unsolicited e-mail, usually of commercial or offensive nature.

Depending on the type of changes, different strategies for building the classifier may be appropriate. The noise must not be modelled by the classifier but filtered out. On the other hand, if there are systematic changes whereby similar class descriptions are likely to reappear, we may want to keep past successful classifiers and simply reuse them when appropriate. If the changes are gradual, we may use a moving window on the training data. If the changes are abrupt we may choose to use a static classifier and when a change is detected, pause and retrain the classifier. This scenario is important when verification of the class labels of the streaming data is not easily available. In general, the more is known about the type of the context drift, the better the chances of devising a successful updating strategy.

2.2 Detecting a Change

Unlabeled Data. Sometimes online labeling is not straightforward. For example, in scanning mammograms for lesions, a verified diagnosis from a specialist is needed if we want to reuse the processed images for further learning. Another example is the credit application problem [13] where the true class label (good/bad) becomes known two years after the classification has taken place. In spam e-mail filtering, the user must confirm the legitimacy of every message if online training is intended.

In case of unknown labels of the streaming data the classifier should be able to signal a potential concept drift based on the unlabeled data. This is typically based on monitoring the unconditional probability distribution, $p(\mathbf{x})$. This problem is called *novelty detection* [18]. It is related to outlier detection in statistics.

One possible practical solution is to train an additional "classifier" to model $p(\mathbf{x})$ and compare the value for each input $\mathbf{x}$ with a threshold θ. If $\mathbf{x}$ is accepted to having come from the distribution of the problem $(p(\mathbf{x}) > \theta)$, the system proceeds to classify it. Else we refuse to classify $\mathbf{x}$ and increment the count of novel objects. We can keep the count over a window of past examples. When the proportion of novel examples reaches a certain level, the system should either go to a halt or request a fresh labeled sample to retrain itself (if this option is incorporated). This addition to the original classifier requires 3 parameters to be specified by the user: θ, the threshold for the novelty; the threshold for the proportion of novel examples; and finally the size of the window. In the simplest case, the distance to the nearest neighbor from the current training data set can be used and θ could be a threshold distance. In theory this technique is equivalent to nonparametric modelling of $p(\mathbf{x})$. More sophisticated modelling approaches include Gaussian Mixture Modelling, Hidden Markov Models, kernel approximation, etc. (see the survey [18]).

Note that using only the unconditional distribution, $p(\mathbf{x})$, we may miss important changes in the class distributions which may alter the classification task but not $p(\mathbf{x})$.

Labeled Data. The most direct indication that there has been an adverse change in the classification task is a consistent fall in the classification accuracy, either sudden or gradual.

2.3 Learn to Forget

Suppose that the classifier has the ability to learn on-line. Instead of trying to detect changes, the classifier is kept constantly up-to-date, which is called "*any time learning*". This means that if the system is stopped at time t, we have the best classifier for the classification problem as it is at time t. The classifier should be able to learn new class descriptions and "forget" outdated knowledge, or "unlearn". The main problem is how to choose the rate of forgetting so that it matches the rate and the type of the changes [15, 23]. If the character of the changes is known or at least suspected, e.g., gradual seasonal changes, then an optimal forgetting strategy can be designed.

Forgetting by Ageing at a Constant Rate. The most common solution is forgetting training objects at a constant rate and using the most recent training set to update (re-train) the classifier. The current classifier uses the past w objects. When a new object arrives, the classifier is updated so as if trained on a data set where the oldest observation is replaced by the newest one.

How large a window do we need? If the window is small, the system will be very responsive and will react quickly to changes but the accuracy of the classifier might be low due to insufficient training data in the window. Alternatively, a large window may lead to a sluggish but stable and well trained classifier. The compromise between the two is viewed as the "stability-plasticity dilemma" [15] whose solution lies with adjusting the "forgetting" parameters for the concrete problem.

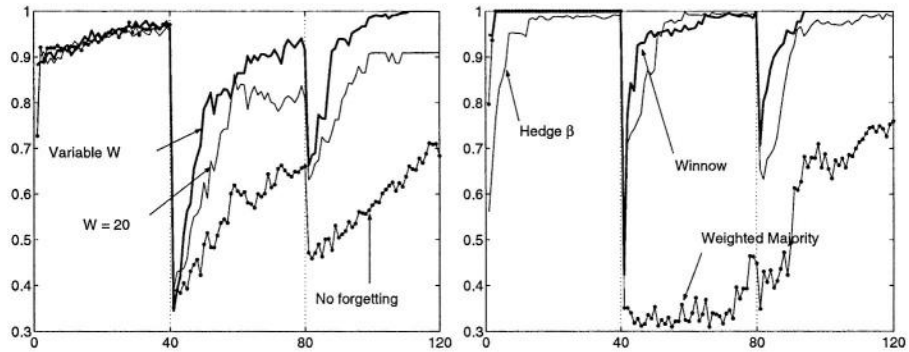
Forgetting by Ageing at a Variable Rate. If a change is detected, the window is shrunk (past examples are forgotten)[3]. For a static bout, the window is expanded to a predefined limit.

To illustrate the concepts being introduced we will keep along a synthetic example taken from [23]. There are 3 categorical features with three categories each: size $\in$ {small, medium, large}, colour $\in$ {red, green, blue} and shape $\in$ {square, circular, triangular}. There are three classification tasks to be learned. The first class to be distinguished is (size = small AND colour = red). 40 examples are generated randomly (with a uniform distribution over the possible values) and labeled according to the class description. These are fed to the classifier sequentially. An independent set of 100 objects labeled according to the current class description is generated for each of the 40 objects. The classifier is tested after each submission on the respective 100 examples. The class description is changed at step 40, so that objects 41 to 80 are labeled according to class (colour = green OR shape = circular). The testing objects are generated again from a uniform random distribution and labeled according to the current class description. Finally, a last set of 40 objects is generated (from 81 to 120) and

[3] FLORA 3 is an example of this group of methods [23].

labeled according to class (size = small OR size = large). The largest of the two prior probabilities for the first stage is 0.89, for the second stage is 0.56, and for the third stage, 0.67.

The importance of using a valid forgetting strategy is demonstrated in Figure 1(a). The Naive Bayes classifier is trained incrementally by updating the probabilities for the classes with each new data point (each example). The plot shows the accuracies of three classifiers versus the number of examples processed. The graph is the average of 10 runs. The classifier based on windows of variable size appears to be more versatile and able to recover from an abrupt change of class concept than the classifier with a constant window and the "no forgetting" classifier.



(a) Naive Bayes classifier without and with forgetting (fixed and variable window)

(b) Hedge β, Winnow and Weighted Majority for an ensemble of 3 ideal classifiers

Fig. 1. Classification accuracy for online algorithms.

Density-Based Forgetting. Sometimes older data points may be more useful than more recent points. Deleting training data based on the distribution is considered in adaptive nearest neighbors models [1, 20]. A weight is attached to each data point. The weight may decay with time or may be modified depending on how often the object has been found as a nearest neighbor.

3 Online Learning

Online learning is becoming increasingly a center stage methodology in the current information era. Massive streams of data are being processed every day, exceeding by far the time and memory capacity of the ever improving modern computational technology. Examples of large data streams can be found in telecommunications, credit card transactions, Internet searches, etc. [6,7,21,22].

Online learning, also termed incremental learning, is primarily focused on processing the data in a sequential way so that in the end the classifier is no worse than a (hypothetical) classifier trained on the batch data. A static environment

is typically assumed, so forgetting of the learned knowledge is not envisaged. Each submitted data point is classified and at some later stage its true label is recovered. The data point and its label are added to the training set of the classifier. The classifier is updated, minimally if possible, to accommodate the new training point. In the simplest case the true label is known immediately. Adapting to the changing environment may come as an effortless bonus in online learning as long as some forgetting mechanism is put into place.

3.1 Desiderata for Online Classifiers

A good online classifier should have the following qualities [7, 21]

- *Single pass through the data.* The classifier must be able to learn from each example without revisiting it.
- *Limited memory and processing time.* Each example should be processed in a (small) constant time regardless of number of the examples processed in the past.
- *Any-time-learning.* If stopped at time t the algorithm should provide the best answer. The trained classifier should ideally be equivalent to a classifier trained on the batch data up to time t. An algorithm which produces a classifier functionally equivalent to the corresponding classifier trained on the batch data is termed a *lossless* online algorithm [19].

In fact, this list applies also for classifiers able to learn in changing environments. Any-time-learning accounts for the need for adapting the training in case of a concept drift.

3.2 Online Classifier Models and Their Extensions for Changing Environments

One of the oldest online training algorithms is the Rosenblatt's perceptron even though it only possesses the second one of the desired qualities.

Learning Vector Quantization (LVQ). LVQ is a simple and successful online learning algorithm originated also from the neural network literature [14]. The principle is as follows. We initialize (randomly or otherwise) a set of points labeled in the classes of the problem. The points are called prototypes. At the presentation of a new object, $\mathbf{x} \in \Re^n$, the prototypes' locations in the feature space are updated. The prototypes of the same class as $\mathbf{x}$ are pulled toward $\mathbf{x}$ while the prototypes from different classes are pushed away. In the simplest version only the nearest to $\mathbf{x}$ prototype is updated. The magnitude of the movement is controlled by a parameter of the algorithm, α, called the learning rate. Let $\mathbf{v} \in \Re^n$ be a prototype. The new value of the prototype is

$$\mathbf{v}^{\text{new}} = \begin{cases} \mathbf{v} + \alpha(\mathbf{x} - \mathbf{v}), & \text{if } \mathbf{v} \text{ has the same class label as } \mathbf{x} \\ \mathbf{v} - \alpha(\mathbf{x} - \mathbf{v}), & \text{otherwise} \end{cases}$$

For training a classifier in stationary environment the learning rate α is typically chosen as a decreasing function of the number of iterations so that at the end of

the algorithm only small changes take place. We can regard α as a function of the accuracy of the classifier and enforce more aggressive training when a change in the environment is detected.

Decision Trees. To understand how the updating works, recall how a decision tree is built. Starting with a root node, at each node we decide whether or not the tree should be split further. If yes, a feature and its threshold value are selected for that node so as to give the best split according to a specified criterion. Upon receiving a new object $\mathbf{x}$, it is propagated down the tree to a leaf forming a path $P_{\mathbf{x}}$. The counts of training objects reaching the nodes on the path $P_{\mathbf{x}}$ are updated. For every internal node, the feature chosen for the split is confirmed along with the threshold value for the split. All necessary alterations are made so that the current tree is equivalent to a tree built upon the whole data set of all past $\mathbf{x}$'s.

Updating a tree may be an intricate job and may take longer than may be acceptable for an online system processing millions of records a day. Domingos and Hulten [6] propose a system for this case called VFDT (Very Fast Decision Trees). They use Hoeffding bound to guarantee that the VFDT will be asymptotically equivalent to the batch tree. The Hoeffding bound allows to calculate the sample size needed to estimate a confidence interval for the mean of the variable of interest, regardless of the distribution of the variable[4]. The importance of the bound is that after a certain number of input points has been processed, there is no need to update the tree further. The VFDT system is guaranteed to be asymptotically optimal for static distributions [6,7]. Hulten et al. [11] extend VFDT to Concept-adapting VFDT to cope with changing environment.

Naive Bayes. In the Naive Bayes model, the class-conditional probabilities are updated with each new $\mathbf{x}$. For simplicity, suppose that $\mathbf{x}$ consists of n categorical variables $x_1, x_2, \ldots, x_n$. For each variable, we keep a probability mass function over the possible categories for each of the classes, $P(x_k|\omega_i)$. When a new $\mathbf{x}$ is received, labeled in class ω_j, the probabilities for ω_j are updated. For example, let $\mathbf{x}$ = (small,blue,circular) and let the label of $\mathbf{x}$ be ω_1. Suppose that 99 objects from class ω_1 have been processed hitherto. Let $P(x_1 = \mathrm{S}\,|\omega_1) = 0.2$, $P(x_1 = \mathrm{M}\,|\omega_1) = 0.7$, and $P(x_1 = \mathrm{L}\,|\omega_1) = 0.1$. The updated values for x_1 are

$$P(x_1 = \mathrm{S}|\omega_1) = \frac{0.2 \times 99 + 1}{100} = 0.208 \quad P(x_1 = \mathrm{M}|\omega_1) = \frac{0.7 \times 99}{100} = 0.693$$

$$P(x_1 = \mathrm{L}|\omega_1) = \frac{0.1 \times 99}{100} = 0.099.$$

[4] The Hoeffding bound is also known as additive Chernoff bound [6]. It states that for any $\delta \in (0,1)$, with probability at least $1-\delta$ and irrespective of the true distribution of the random variable y with range R, the true mean of y is within ϵ of the mean $\bar{y}$ calculated from an i.i.d. sample of size n taken from the distribution of y where

$$\epsilon = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}}.$$

Naive Bayes classifier is a lossless classifier. A forgetting mechanism can be incorporated by keeping a window and "unlearning" the oldest example in the window while learning **x**.

Neural Networks. The training of neural networks is carried out by submitting the training set in a sequential manner a specified number of times (number of epochs). To train a NN online we abandon the epoch protocol and use the continuous stream of data instead. NN classifier is not a lossless model [19]. If we keep the training on-going with the data stream, the NN parameters will follow the concept drift. The responsiveness of the NN to changes will depend on the learning rate used in the backpropagation algorithm. A neural architecture called PFAM (Probabilistic Fuzzy ARTMAP neural network) is used as the base classifier for an ensemble suitable for online learning in [15].

Nearest Neighbor. Nearest neighbour classifier is both intuitive and accurate. We can build the training set (the reference set from which the neighbors are found) by storing each labeled **x** as it comes. This model is called IB1 (instance-based learning, model 1) in [1]. IB1 is a lossless algorithm but it fails on the criteria for time and memory. IB2 accepts in the reference set only the **x**'s which are misclassified using the reference set at the time they arrive. This is the online version of the Hart's *editing algorithm* [10]. Editing algorithms look for the minimal possible reference set with as high as possible generalization accuracy. IB3 introduces forgetting based on the usefulness of the **x**'s in the reference set. There have been many studies on editing for the nearest neighbour algorithm [2, 5]. Developing efficient online editing algorithms could be one of the important future directions.

4 Ensemble Strategies for Changing Environments

When and why would an ensemble be better than a single classifier in a changing environment?

In massive data streams we are interested in simple models because there might not be time for running and updating an ensemble. On the other hand, Wang et al. argue that a simple ensemble might be easier to use than a decision tree as a single adaptive classifier [22].

When time is not of primary importance but very high accuracy is required, an ensemble would be the natural solution. An example is scanning mammograms for tissue lesions or cervical smear images for cell abnormalities. In these cases taking several minutes per image will be acceptable.

Various online ensemble solutions have been proposed for changing environments. We can group the approaches as follows

- Dynamic combiners (or "horse racing" algorithms). It this group we put the ensemble methods where the individual classifiers (experts) are trained in advance and the changes in the environment are tracked by changing in the combination rule.
- Updated training data. The algorithms in this group rely on using fresh data to make online updates of the team members. The combination rule may or may not change in the process.

- Updating the ensemble members. Classifiers in the online ensemble can be updated online or retrained in a batch mode if blocks of data are available.
- Structural changes of the ensemble. "Replace the loser" is one possible strategy from this group. In the case of a change in the environment, the individual classifiers are re-evaluated and the worst classifier is replaced by a new classifier trained on the most recent data.
- Adding new features. The features will naturally change their importance along the life of the ensemble. There might be a need for incorporating new features without going through the loop of re-designing the entire ensemble system.

Some of the approaches are detailed below.

4.1 "Horse Racing" Ensemble Algorithms

We will follow the horse racing analogy aptly used in [8] to introduce the Hedge-β algorithm and AdaBoost. Assume that you want to predict the outcome of a horse race and you have L expert-friends whose prediction you can take or ignore. Each $\mathbf{x}$ is a particular race and the class label is its outcome. You note which of the experts have been wrong in their prediction and update a set of weights to keep track on whose prediction is currently most trustworthy. The most famous representative of this group is the **Weighted Majority** algorithm by Littlestone and Warmuth [17], shown below.

1. Given is a classifier ensemble $\mathcal{D} = \{D_1, \ldots, D_L\}$.
   Initialize all L weights as $w_j = 1$.
2. *Operation.* For a new $\mathbf{x}$, calculate the support for each class ω_i as the sum of the weights of all classifiers D_i that suggest class label ω_i for $\mathbf{x}$.
   Label $\mathbf{x}$ to the class with the largest support.
3. *Training.* Observe the true label of $\mathbf{x}$. Using a pre-defined $\beta \in [0, 1]$, update the weights for all experts whose prediction was incorrect as $w_i \leftarrow \beta w_i$.
4. Continue from Step 2.

Hedge β operates in the same way as the Weighted Majority algorithm but instead of taking the weighted majority, one classifier is selected from the ensemble and its prediction is taken as the ensemble decision. The classifier is selected according to the probability distribution defined by the normalized weights.

Winnow is another algorithm under the horse race heading [16]. The algorithm was originally designed for predicting the value of a (static) Boolean function, called target function, $f : \{0, 1\}^n \rightarrow \{0, 1\}$, by getting rid of the irrelevant variables in the Boolean input. Translated into our ensemble terminology, Winnow follows the weighted majority algorithm but uses a different updating Step 3. The weights are reconsidered only if the ensemble gives a wrong prediction for the current input $\mathbf{x}$. If classifier D_i gives the correct label for $\mathbf{x}$, its weight is increased as $w_i \leftarrow \alpha w_i$, where $\alpha > 1$ is a parameter of the algorithm. This is called a *promotion step,* rewarding the expert not so much for predicting correctly but for predicting correctly in times when the global algorithm should

have listened to them more carefully [3]. If classifier D_i gives an incorrect label for $\mathbf{x}$, the *demotion step* takes place as $w_i \leftarrow w_i/\alpha$, ensuring that D_i takes a share of the blame for the ensemble error. Good results have been obtained for $\alpha = 2$ [16]. Machine learning literature abounds in proofs of theoretical bounds and further variants of Winnow and Weighted Majority [24].

Figure 1(b) shows the average of 10 runs of Hedge β, The Weighted Majority and the Winnow algorithms for the artificial data described above. Here we are "cheating" in a way because we suppose that the ensemble consists of three perfect classifiers, one for each stage. For example, if classifier D_1 was always picked to make the decision for objects 1 to 40, classifier D_2 for objects from 41 to 80, and classifier D_3 for objects from 81 to 120, the classification accuracy would be 100% for all objects. As the plot shows, the Weighted Majority cannot recover from changing the class description at object 40 even though the ensemble consists of the three optimal experts, Hedge β would follow the same pattern if applied in its standard form. Here we modified it a little. Instead of updating all the weights for each $\mathbf{x}$, we update only the weight of the classifier selected to label $\mathbf{x}$. Winnow is the undisputed winner between the three algorithms in the plot. It quickly discovers the right classifier for each part. Obviously, involving the ensemble accuracy in the weight updating step makes all the difference.

Blum [3] describes the above algorithms as "learning simple things really well". The problem with the horse racing approach, in the context discussed here, is that the individual classifiers are not re-trained at any stage. Thus their expertise may ware off and the ensemble may be left without a single adequate expert for the new environment. Classifiers in the original algorithms are taken from the whole (finite) set of possible classifiers for the problem, so there is no danger of lack of expertise.

Mixture of experts type of training constitutes a special niche in the group of dynamic combiner methods. Originally developed as a strategy for training an ensemble of multi-layer perceptrons [12], the idea has a more generic standing as an on-line algorithm suitable for changing environments. The ensemble operates as an on-line dynamic classifier selection system and updates one classifier and the combination rule with each new example. Figure 2 shows a sketch of a training cycle upon a presentation of a new data point $\mathbf{x}$. The individual classifiers must be supplied with a forgetting mechanism so that outdated knowledge is "unlearned".

Good Old Combination Rules. Any combination rule can be applied for the online ensemble. Although there is a marked preference for majority vote [21] and weighted majority, there is no reason why we should stop here. Naive Bayes and BKS combination rules are explored in [15]. Both can be updated online.

4.2 Updated Training Data for Online Ensembles

Reusing the Data Points. Oza proposes a neat **online bagging** algorithm which converges to the batch bagging as the number of training examples and the number of classifiers tend to infinity [19]. The idea is that the training samples

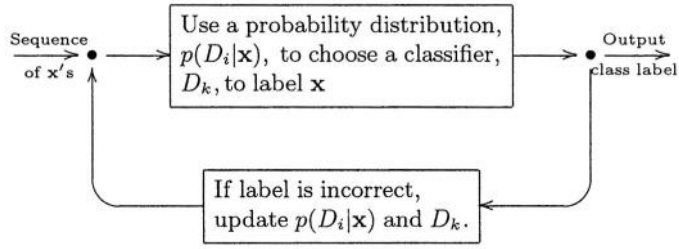


Fig. 2. A generic on-line ensemble construction method based on dynamic classifier selection.

for the L classifiers in the ensemble can be created incrementally. The base classifiers are trained using online (preferably lossless) classifier models. To form the training sample for classifier D_i, Oza observes that each training data point appears K times in a training set of size N sampled with replacement from the available data set of size N. K is a binomial random variable, so the probability that a certain $\mathbf{z}$ appears $K = k$ times in the ith training set is

$$P(K = k) = \binom{N}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{N-k}$$

When N is large the probability for selecting a particular data point is small, and the binomial distribution can be approximated by a Poisson distribution with $\lambda = 1$. Therefore the probability $P(K = k)$ is calculated as $P(K = k) = \frac{\exp(-1)}{k!}$. The probabilities for $k = 0 \ldots, 7$ are tabulated below. Practically it is very unlikely that any $\mathbf{z}$ will be present in a sample more than 7 times.

k	0	1	2	3	4	5	6	7
$P(K = k)$	0.3679	0.3679	0.1839	0.0613	0.0153	0.0031	0.0005	0.0001

The online bagging proposed by Oza is given below

1. Pick the number of classifier in the ensemble, L.
2. *Training.* For each new labeled data point $\mathbf{x}$, for each classifier D_i, $i = 1, \ldots, L$, sample from the Poisson distribution explained above to find k. Put k copies of $\mathbf{x}$ in the training set of D_i. Retrain D_i using an online training algorithm.
3. *Operation.* Take the majority vote on the ensemble to label $\mathbf{x}$.
4. Continue from Step 2.

The **online boosting** algorithm proposed next by Oza [19] draws upon a similar idea as his online bagging. The number of times that a data point appears in the training set of classifier D_i is again guided by the Poisson distribution. However, the parameter of the distribution, λ will change from one classifier to the next. We start with $\lambda = 1$ to add copies of $\mathbf{x}$ to the first sample and train classifier D_1 on it. If D_1 misclassifies $\mathbf{x}$, λ increases so that $\mathbf{x}$ is likely to feature

more often in the training set of D_2. The subsequent λs are updated in the same manner until all L classifiers are retrained.

Filtering. Breiman suggests to form the training sets for the consecutive classifiers as the data flows through the system, called **Pasting small votes.** [4]. The first N data points are taken as the training set for D_1. The points coming next are filtered so that the next classifier is trained on a sample such that the ensemble built so far (just D_1 for now) will misclassify approximately half of it. To form the sample for classifier D_{k+1}, run each new data point through the current ensemble. If misclassified, add it to the new training set. If correctly classified, add it to the training set with probability $\frac{e_k}{1-e_k}$, where e_k is an estimate of the error of the ensemble on the training set for classifier D_k. To calculate this estimate, Breiman suggests to use the smoothed version

$$e_k = 0.75 \times e_{k-1} + 0.25 \times r_k,$$

where e_{k-1} is the *generalization* ensemble error[5] up to classifier D_{k-1} and r_k is the ensemble error found during building the training set for classifier D_k. The smoothing is needed because if N is small, then the estimate $r(k)$ will be noisy. In this way, L classifiers are subsequently trained on diverse data sets.

Breiman proposes to use e_k as a stopping criterion. In case of static environment, the ensemble error will comfortably level off at some L and no more ensemble members will be needed. If, however, the environment changes, e_k will start increasing again. Changes in the ensembles will be needed at this point. We can, for example, replace ensemble members selected through a certain criterion. Alternatively, we can keep an ongoing process of building classifiers for the ensemble and "forget" the oldest member each time a new member is added, thus keeping the ensemble size constant at L.

The methods in this subsection have been proposed as variants of the corresponding batch methods for stationary environments. When faced with changing environments, we have to introduce a forgetting mechanism. One possible solution would be to use a window of past examples, preferably of variable size, as discussed earlier.

Using Data Blocks or Chunks. In this model we assume that the data come as blocks of data points at a time. The ensemble can be updated using batch mode training on a "chunk" of data [9,21,22]. The blocks can be treated as single items of data in the sense that we may train the ensemble on the most recent block, on a set of past blocks or on the whole set of blocks. A forgetting strategy needs to be incorporated in the model, e.g., a window of blocks. Ganti et al. [9] propose a change detection algorithm based on difference between blocks of data. They also consider selective choices of past blocks so that a certain pattern is modelled. For example, if the blocks come once a day and contain all the data for that day, it might be interesting to build an ensemble for the Wednesday's classification problem and apply it on the subsequent Wednesdays.

Figure 3 gives an illustration of the three data handling approaches.

[5] This error may be measured on an independent testing set or on the so called "out-of-bag" data points. For details see [4].

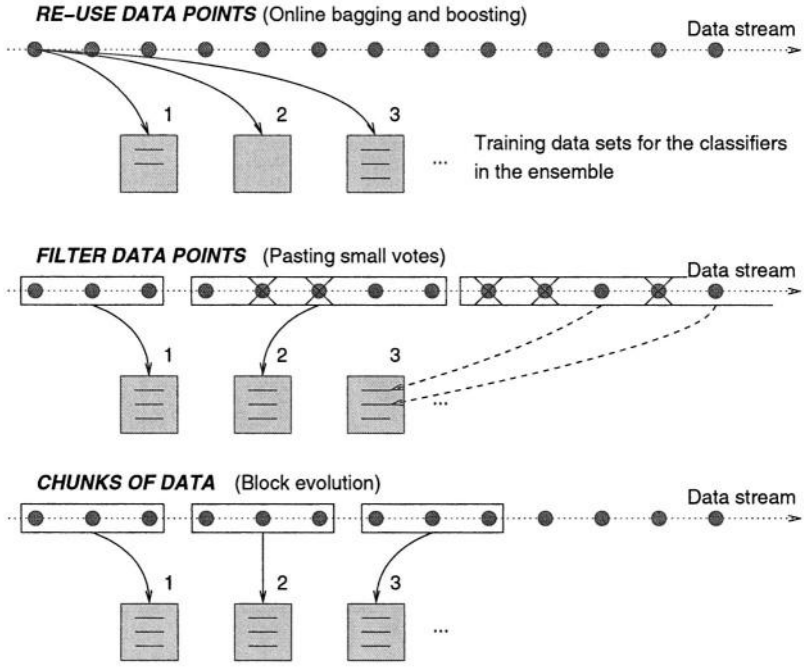


Fig. 3. Illustration of the three data handling approaches for building online classifier ensembles.

4.3 Changing the Ensemble Structure

The simplest strategy here can be named **replace the oldest.** We remove the oldest classifier in the ensemble and train a new classifier to take its place. The three methods of data handling described above can be run within this framework.

We may adopt a more sophisticated strategy for pruning of the ensemble. Wang et al. [22] propose to evaluate all classifiers using the *most recent* "chunk" of data as the testing set. The classifiers whose error exceeds a certain threshold are discarded from the ensemble. We call this strategy **replace the loser.** Street and Kim [21] consider a "quality score" for replacing a classifier in the ensemble based on its merit to the ensemble, not only on the basis of its individual accuracy. This quality score is a soft version of the Winnow approach for updating the weights of the classifiers. Suppose we are testing the ensemble members on a new data set. We start with equal weights for all ensemble members. For each $\mathbf{x}$, let P_1 be the proportion of votes for the majority class, P_2 be the proportion of votes for the second most voted for class, P_c be the proportion for the correct class of $\mathbf{x}$, and P_i be the proportion for the class suggested by classifier D_i. The score for D_i is updated as $w_i \leftarrow w_i + \delta(\mathbf{x})$, where $\delta(\mathbf{x})$ is

$$
\begin{array}{ll}
1 - |P_1 - P_2|, & \text{if both the ensemble and } D_i \text{ are correct} \\
1 - |P_1 - P_c|, & \text{if the ensemble is wrong and } D_i \text{ is correct} \\
1 - |P_i - P_c|, & \text{if } D_i \text{ is wrong (regardless of the ensemble)}
\end{array}
$$

Most of the methodologies for building ensembles considered hitherto have built-in methodologies for controlling the ensemble size. Alternatively, the ensemble size can be left as a parameter of the algorithm or a designer's choice.

5 Conclusions

Knowing that ensemble methods are accurate, flexible and sometimes more efficient than single classifiers, the purpose of this study was to explore classifier ensembles within changing classification environments. "Concept drift" has long been an ongoing theme in machine learning. Ensemble methods have been recently probed in this line. The frontier of advanced research in multiple classifier systems which we are reaching now is likely to bring new fresh solutions to the changing environment problem.

The ensemble can be perceived as a living population – expanding, shrinking, replacing and retraining classifiers, taking on new features, forgetting outdated knowledge. Ideas can be borrowed from evolutionary computation and artificial life. Even if we are still a long way from putting together a toolbox and a theoretical fundament, the road ahead is fascinating.

References

1. D. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning,* 6:37–66, 1991.
2. J.C. Bezdek and L.I. Kuncheva. Nearest prototype classifier designs: An experimental study. *International Journal of Intelligent Systems,* 16(12):1445–1473, 2001.
3. A. Blum. Empirical support for Winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In *Proc. 12th International Conference on Machine Learning,* pages 64–72, San Francisco, CA., 1995. Morgan Kaufmann.
4. L. Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning,* 36:85–103, 1999.
5. B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, Los Alamitos, California, 1990.
6. P. Domingos and G. Hulten. Mining high-speed data streams. In *Knowledge Discovery and Data Mining,* pages 71–80, 2000.
7. P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics,* 12:945–949, 2003.
8. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences,* 55(1):119–139, 1997.
9. V. Ganti, J. Gehrke, and R. Ramakrishnan. Mining data streams under block evolution. *A CM SIGKDD Explorations Newsletter,* 3:1–10, 2002.
10. P.E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory,* 16:515–516, 1968.
11. G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *In Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pages 97–106. ACM Press, 2001.

12. R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation,* 3:79–87, 1991.
13. M. G. Kelly, D. J. Hand, and N. M. Adams. The impact of changing populations on classifier performance. In *Proc 5th ACM SIGDD International Conference on Knowledge Discovery and Data Mining,* pages 367–371, San Diego, CA, 1999. ACM Press.
14. T. Kohonen. *Self-Organizing Maps.* Springer, New York, 3rd edition, 2000.
15. C. P. Lim and R. E. Harrison. Online pattern classification with multiple neural network systems: An experimental study. *IEEE Transactions on Systems, Man, and Cybernetics,* 35:235–247, 2003.
16. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear threshold algorithm. *Machine Learning,* 2:285–318, 1988.
17. N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inform. Computation,* 108:212–261, 1994.
18. M. Markou and S. Singh. Novelty detection: a review. Part 1: statistical approaches. *Signal Processing,* 83(12):2481–2497, 2003.
19. N. C. Oza. *Online Ensemble Learning.* PhD thesis, University of California, Berkeley, 2001.
20. M. Salganicoff. Density-adaptive learning and forgetting. In *Proceedings of the 10th International Conference on Machine Learning,* pages 276–283, 1993.
21. W. N. Street and Y. S. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pages 377–382. ACM Press, 2001.
22. H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pages 226–235. ACM Press, 2003.
23. G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning,* 23:69–101, 1996.
24. G. Widmer and M. Kubat. Special Issue on Context Sensitivity and Concept Drift. *Machine Learning,* 32, 1998.

A Generic Sensor Fusion Problem: Classification and Function Estimation

Nageswara S.V. Rao

Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
raons@ornl.gov

Abstract. A generic fusion problem is studied for multiple sensors whose outputs are probabilistically related to their inputs according to unknown distributions. Sensor measurements are provided as iid input-output samples, and an empirical risk minimization method is described for designing fusers with distribution-free performance bounds. The special cases of isolation and projective fusers for classifiers and function estimators, respectively, are described in terms of performance bounds. The isolation fusers for classifiers are probabilistically guaranteed to perform at least as good as the best classifier. The projective fusers for function estimators are probabilistically guaranteed to perform at least as good as the best subset of estimators.

1 Introduction

The information fusion problems have been solved for centuries in various disciplines, such as political economy, reliability, pattern recognition, forecasting, and distributed detection. In multiple sensor systems, the fusion problems arise naturally when overlapping regions are covered by the sensors. Often, the individual sensors can themselves be complex, consisting of sophisticated sensor hardware and software. Consequently, sensor outputs can be related to the actual object features in a complicated manner, and these relationships are often characterized by probability distributions. Early information fusion methods required statistical independence of sensor errors, which greatly simplified the fuser design; for example, a weighted majority rule suffices in detection problems. Such solutions are not applicable to current multiple sensor systems, since the sensors measurements can be highly correlated and consequently violate the statistical independence property. Another classical approach to fuser design is the Bayesian method that minimizes a suitable expected risk, which relies on analytical expressions for sensor distributions. Deriving the required closed-form sensor distributions is very difficult since it often requires the knowledge of areas such as device physics, electrical engineering, and statistical modeling. Particularly when only a finite number of measurements are available, the selection of a fuser from a carefully chosen function class is easier, in a fundamental information-theoretic sense, than inferring completely unknown sensor distributions [21].

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 16–30, 2004.

In operational sensor systems measurements are collected by sensing objects and environments with known parameters. Thus fusion methods that utilize such empirical observational or experimental data will be of high practical relevance. In this paper, we present a brief overview of rigorous approaches for designing such fusers based on the empirical process theory [21] to provide performance guarantees based on finite samples. We briefly describe a general fuser design approach and illustrate it using a vector space method. A more detailed account of the generic sensor fusion problem can be found in [18]. The problem of combining outputs of multiple classifiers is a special case of the generic sensor fusion problem, wherein the training sample corresponds to the measurements. We describe the isolation fuser methods for classifiers to probabilistically ensure that fuser's performance guarantees are at least as good as those of best classifier. The fusion of function estimators is another special case of the sensor fusion problem which is of practical utility. We then describe the nearest-neighbor projective fuser for function estimators that performs at least as good as the best projective combination of the estimators. Both isolation and projective fusers have been originally developed for the generic sensor fusion problem, and we sharpen the general performance results for these special cases.

This paper presents a brief account of results from other papers. We describe the classical sensor fusion methods in Section 2. We present a generic sensor fusion problem and a solution using empirical risk minimization in Section 3. We describe the problem of fusing classifiers and function estimators in Sections 4 and 5, respectively. The original notations from the respective areas are retained in the individual sections; while it results in a non-uniform notation, it makes it easier to relate these results to the individual areas.

2 Classical Fusion Problems

Fusion methods for multiple sources to achieve performances exceeding those of individual sources have been studied in political economy models in 1786 and composite methods in 1818. In the twentieth century, fusion methods have been applied in a wide spectrum of areas such as reliability, forecasting, pattern recognition, neural networks, decision fusion, and statistical estimation. A brief overview of early information fusion works can be found in [7]. The problem of fusing classifiers is relatively new and is first addressed in a probabilistic framework by Chow [1] in 1965.

When sensor distributions are known, several fusion rule estimation problems have been solved under various formulations. A simpler version of this problem is the Condorcet jury model (see [4] for an overview), where a majority rule can be used to combine 1-0 probabilistically independent decisions of a group of N members. If each member has probability p of making a correct decision, the probability that the majority makes the correct decision is $p_N = \sum_{i=N/2}^{N} \binom{N}{i} p^i (1-p)^{N-i}$. Then we have an interesting dichotomy: (a) if $p > 0.5$, then $p_N > p$ and $p_N \to 1$ as $N \to \infty$; and (b) if $p < 0.5$, then $p_N < p$ and $p_N \to 0$ as $N \to \infty$.

For the boundary case $p = 0.5$ we have $p_N = 0.5$. Interestingly, this result has been rediscovered by von Neumann in 1959 in building reliable computing devices using unreliable components by taking a majority vote of duplicated components. For multiple classifiers, a weighted majority fuser is optimal [1] under statistical independence, and the fuser weights can be derived in a closed-form using the classifier detection probabilities. Over the past few years, multiple classifier systems have witnessed an extensive interest and growth [5, 23].

The distributed detection problem [22] studied extensively in the target tracking area can be viewed as a generalization of the above two problems. The Boolean decisions from a system of detectors are combined by minimizing a suitably formulated Bayesian risk function. The risk function is derived from the detector densities and the minimization is typically carried out using analytical or deterministic optimization methods. In particular, the risk function used for classifier fusion in [1] corresponds to the misclassification probability and its minima is achieved by the weighted majority rule. In these works, the sensor distributions are assumed to be known, which is reasonable in their domains. While several of these solutions can be converted into sample-based ones [9], these are not primarily designed for measurements. As evidenced in practical multiple sensor systems and classifiers, it is more pragmatic to have the measurements rather than the error distributions.

3 A Generic Sensor Fusion Problem

We consider a multiple sensor system of N sensors, where sensor S_i, $i = 1, 2, \ldots, N$, outputs $Y^{(i)} \in \Re^d$ corresponding to input $X \in \Re^d$ according to distribution $P_{Y^{(i)}|X}$. Intuitively, input X is the "measured" quantity such as presence of a target or a value of feature vector. The *expected error* of sensor S_i is defined as

$$I(S_i) = \int C\left(X, Y^{(i)}\right) dP_{Y^{(i)},X},$$

where $C : \Re^d \times \Re^d \mapsto \Re$ is the cost function. Here, $I(S_i)$ is a measure of how good sensor S_i is in "measuring" input feature X. If S_i is a detector [22] or classifier [2], we can have $X \in \{0, 1\}$ and $Y^{(i)} \in \{0, 1\}$, where $X = 1$ (0) corresponds to presence (absence) of a target. Then $I(S_i) = \int \left[X \oplus Y^{(i)}\right] dP_{Y^{(i)},X}$ is the probability of misclassification (false alarm and missed detection) of S_i, where $\oplus$ is the exclusive-OR operation[1].

The *measurement error* corresponds to the randomness in measuring a particular value of feature X, which is distributed according to $P_{Y^{(i)}|X}$. The *systematic error* at X corresponds to $E[C(X, Y^{(i)})|X]$ which must be 0 in the case of a perfect sensor. This error is often referred to as the bias error.

We consider a fuser $f : \Re^{Nd} \mapsto \Re^d$ that combines the outputs of sensors $Y = \left(Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)}\right)$ to produce the fused output $f(Y)$. We define the *expected error* of the fuser f to be

[1] Alternatively, X can be expanded to include the usual "feature" vector and $C(.)$ can be redefined so that $I(S_i)$ is misclassification probability.

$$I_F(f) = \int C(X, f(Y))dP_{Y,X}$$

where $Y = \left(Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)}\right)$. The objective of fusion is to achieve low values of $I_F(f)$, and for this both systematic and measurement errors must be taken into account. The fuser is typically chosen from a family of fusion rules $\mathcal{F} = \{f : \Re^{Nd} \mapsto \Re^d\}$ which could be either explicitly or implicitly identified. The *expected best* fusion rule f^* satisfies $I_F(f^*) = \min_{f \in \mathcal{F}} I_F(f)$. For example, if $\mathcal{F}$ is a set of sigmoidal neural networks obtained by varying the weight vector for a fixed architecture, then $f^* = f_{w^*}$ corresponds to the weight vector w^* that minimizes $I_F(.)$ over all weight vectors.

In this formulation, since $I_F(.)$ depends on $P_{Y,X}$, f^* cannot be computed even in principle if the distribution is not known. We consider that only an independently and identically distributed (iid) l-sample (X_1, Y_1), (X_2, Y_2), ..., (X_l, Y_l) is given, where $Y_i = \left(Y_i^{(1)}, Y_i^{(2)}, \ldots, Y_i^{(N)}\right)$ and $Y_i^{(j)}$ is the output of S_j in response to input X_i. Our goal is to obtain an estimator $\hat{f}$, based *only* on a sufficiently large sample, such that

$$P_{Y,X}^l \left[I_F(\hat{f}) - I_F(f^*) > \epsilon\right] < \delta \tag{1}$$

where $\epsilon > 0$ and $0 < \delta < 1$, and $P_{Y,X}^l$ is the distribution of iid l-samples. As per this condition the "error" of $\hat{f}$ is within ϵ of optimal error (of f^*) with probability $1 - \delta$, *irrespective* of the sensor distributions. Since $\hat{f}$ is to be "chosen" from a potentially infinite set, namely $\mathcal{F}$, based only on a finite sample, this condition is a reasonable target. Strictly stronger conditions are generally not possible to achieve. For example, consider the condition $P_{Y,X}^l[I_F(\hat{f}) > \epsilon] < \delta$ for the case case of classifiers $\mathcal{F} = \{f : [0,1]^N \mapsto \{0,1\}\}$. This condition cannot be satisfied, since for any classifier $f \in \mathcal{F}$, there exists a distribution for which $I_F(f) > 1/2 - \rho$ for any $\rho \in [0,1]$ (see Theorem 7.1 of [2] for details).

Consider a simple two-sensor system such that $Y^{(1)} = a_1 X + Z$, where Z is normally distributed with zero mean, and is independent of X, i. e. a constant scaling error and a random additive error. For the second sensor, we have $Y^{(2)} = a_2 X + b_2$, which has a scaling and bias error. Let X be uniformly distributed over [0, 1], and $C[X, Y] = (X - Y)^2$. Then, we have $I(S_1) = (1 - a_1)^2$ and $I(S_2) = (1 - a_2 - b_2)^2$, which are non zero in general. For

$$f\left(Y^{(1)}, Y^{(2)}\right) = \frac{Y^{(1)}}{2a_1} + \frac{1}{2a_2}(Y^{(2)} - b_2).$$

we have $I_F(f) = 0$, since the bias b_2 is subtracted from $Y^{(2)}$ and the multipliers cancel the scaling error. Such fuser can be designed only with a significant insight into sensors, in particular with a detailed knowledge about the distributions. To illustrate the effects of finite samples, we generate three values for X given by $\{0.1, 0.5, 0.9\}$ with corresponding Z values given by $\{0.1, -0.1, -0.3\}$. The corresponding values for $Y^{(1)}$ and $Y^{(2)}$ are given by $\{0.1a_1 + 0.1, 0.5a_1 - 0.1, 0.9a_1 - 0.3\}$

and $\{0.1a_2 + b_2, 0.5a_2 + b_2, 0.9a_2 + b_2\}$ respectively. Consider a linear fuser $f\left(Y^{(1)}, Y^{(2)}\right) = w_1 Y^{(1)} + w_2 Y^{(2)} + w_3$. The following weights enable the fuser outputs to exactly match X values for each measurement:

$$w_1 = \frac{1}{0.2 - 0.4a_1}, w_2 = \frac{1}{0.4a_2} \text{ and } w_3 = \frac{0.1a_1 + 0.1}{0.4a_1 + 0.1} - \frac{0.1a_2 + b_2}{0.4a_2}.$$

While these weights achieve zero error on the measurements they do not achieve zero value for I_F (even though a fuser with zero expected error exists and can be computed if the sensor distributions are given). The idea behind the criterion in Eq 1 is to achieve performances close to optimal using only a sample. To achieve this a suitable $\mathcal{F}$ is selected first, from which a fuser is chosen to achieve small error on a sufficiently large sample, as will be illustrated subsequently.

Due to the generic nature of the sensor fusion problem described here, it is related to a number of similar problems in a wide variety of areas. A detailed discussion of these aspects can be found in [18].

3.1 Empirical Risk Minimization

Consider that the *empirical error estimate*

$$I_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} \left[X_i - f\left(Y_i^{(1)}, Y_i^{(2)}, \ldots, Y_i^{(N)}\right)\right]^2$$

is minimized by $\hat{f} \in \mathcal{F}$. Such a method corresponds to an ad hoc approach of choosing a class of fusers such as neural networks or linear fusers, and choosing a particular fuser to minimize the error within the class. Performance of such method, including the basic feasibility, depends on the fuser class and the complexity of minimizing the empirical error. For example, if $\mathcal{F}$ has finite capacity [21], then under bounded error, or bounded relative error for sufficiently large sample, we have $P^l_{Y,X}\left[I_F(\hat{f}) - I_F(f^*) > \epsilon\right] < \delta$ for arbitrarily specified $\epsilon > 0$ and δ, $0 < \delta < 1$. Typically, the required sample size is expressed in terms of ϵ and δ and the parameters of $\mathcal{F}$. The most general result [13] that ensures this condition is based on the scale-sensitive dimension, which establishes the basic tractability of this problem. But this general method often results in very loose bounds for the sample size, and tighter estimates are possible by utilizing specific properties of $\mathcal{F}$.

If $\mathcal{F}$ is a vector space of dimensionality d_V, we have the following results [12]: (a) the sample size is a simple function of d_V, (b) $\hat{f}$ can be computed using least square methods in polynomial time, and (c) no smoothness conditions are required on the functions or distributions. For simplicity consider that $X \in [0, 1]$ and $Y \in [0, 1]^N$. Let f^* and $\hat{f}$ denote the expected best and empirical best fusion functions chosen from a vector space $\mathcal{F}$ of dimension d_V and range [0, 1]. Given an iid sample of size

$$\frac{512}{\epsilon^2}\left[d_V \ln\left(\frac{64e}{\epsilon} + \ln\frac{64e}{\epsilon}\right) + \ln(8/\delta)\right],$$

we have $P\left[I_F(\hat{f}) - I_F(f^*) > \epsilon\right] < \delta$ (see [12] for details). This method subsumes two very important cases [12]:

(a) *Potential Functions:* The potential functions where $f_i(y)$ is of the form $exp((y-\alpha)^2/\beta)$ for suitably chosen constants α and β, constitute an example of the vector space method.

(b) *Special Neural Networks:* In two-layer sigmoidal networks of [6], the unknown weights are only in the output layer, which enables us to express each network in the form $\sum_{k=1}^{d_V} a_i \eta_i(y)$ with universal $\eta_i(.)$'s.

Similar sample size estimates have been derived for fusers based on feedforward neural networks in [10]. Also non-linear statistical estimators can be employed to estimate the fuser based on the sample, such as the Nadaraya-Watson estimator [12]. The main limitation of empirical risk minimization approach is that $\hat{f}$ is only guaranteed to be close to f^* but there are no guarantees that the latter is any good. While it is generally true that if $\mathcal{F}$ is large enough, f^* would perform better than best sensor, it is indeed possible that it performs worse than worst sensor. Systematic approaches such as isolation fusers [17] and projective fusers [15] would be useful to ensure the fuser performance. We will subsequently discuss the special cases of isolation and projective fusers for classifiers [11] and function estimators [19], respectively. We note that projective fusers have also been applied to classifiers [11] and isolation fusers have also been applied to function estimators [16].

3.2 Example

We consider 5 classifiers such that $Y \in \{0,1\}^5$ such that $X \in \{0,1\}$ corresponds to "correct" class, which is generated with equal probabilities, i. e., $P(X=0) = P(X=1) = 1/2$ [20]. The error of classifier C_i, $i = 1, 2, \ldots, 5$, is described as follows: the output $Y^{(i)}$ is correct decision with probability of $1 - i/10$, and is the opposite with probability $i/10$. The task is to combine the outputs of classifiers to predict the correct class. The percentage error of the individual classifiers and the fused system based on the Nadaraya-Watson estimator is presented in Table 1. Note that the fuser is consistently better than the best classifier C_1 beyond the sample size of 1000. The performance results of Nadaraya-Watson estimator, empirical decision rule, nearest neighbor rule, and Bayesian rule based on the

Table 1. Percentage error of Nadaraya-Watson estimator and individual classifiers.

Sample Size	Test set	C_1	C_2	C_3	C_4	C_5	Nadaraya-Watson
100	100	7.0	20.0	33.0	35.0	55.0	12.0
1000	1000	11.3	18.5	29.8	38.7	51.6	10.6
10000	10000	9.5	20.1	30.3	39.8	49.6	8.58
50000	50000	10.0	20.1	29.8	39.9	50.1	8.860

Table 2. Correct classification percentage of fusers.

Sample Size	Test Size	Bayesian Fuser	Empirical Decision	Nearest Neighbor	Nadaraya-Watson
100	100	91.91	23.00	82.83	88.00
1000	1000	91.99	82.58	90.39	89.40
10000	10000	91.11	90.15	90.81	91.42
50000	50000	91.19	90.99	91.13	91.14

analytical formulas are presented in Table 2. The Bayesian rule is computed based on the formulas used in the data generation and is provided for comparison only.

4 Isolation Fusers for Classifiers

Over the past decades several methods, such as nearest neighbor rules, neural networks, tree methods, and kernel rules, have been developed for designing classifiers. Often, the classifiers are quite varied and their performances are characterized by various smoothness and/or combinatorial parameters [2]. The designer is thus faced with a wide variety of choices which are not easily comparable. It is generally known that a good fuser outperforms the best classifier, and at the same time, a bad fuser choice can result in a performance worse than the worst classifier. Thus it is very important to employ fusion methods that provide concrete performance guarantees – in particular, for the fuser to be reasonable it must perform at least as well as the best classifier.

We are given an independently and identically distributed (iid) sample (X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n), according to an unknown distribution $P_{X,Y}$, where $X_i \in \Re^d$ and $Y_i \in \{0, 1\}$. The problem is to design a classifier $\phi : \Re^d \mapsto \{0, 1\}$ based on the sample that ensures a small value for the *probability of misclassification*

$$L(\phi) = \int_X I_{\{\phi(X) \neq Y\}} dP_{X,Y},$$

where $I_D(x)$ is the *indicator function* of the set $D \subseteq \Re^d$ such that $I_C(x) = 1$ if $x \in C$ and $I_C(x) = 0$ otherwise. We often suppress the operand x when it is clear from the context.

For $\phi \in \mathcal{H}$, the *empirical error of misclassification* is given by

$$\hat{L}(\phi) = \frac{1}{n} \sum_{i=1}^{n} I_{\{\phi(X_i) \neq Y_i\}}.$$

Let $\hat{\phi}$ minimize $\hat{L}(.)$ over $\mathcal{H}$. If $\mathcal{H}$ has finite Vapnik-Chervonenkis dimension $V_{\mathcal{H}}$, we have [2]

$$P_{X,Y}^n \left[L(\hat{\phi}) - \min_{\phi \in \mathcal{H}} L(\phi) > \epsilon \right] \leq \delta$$

for sufficiently large n, irrespective of the distribution $P_{X,Y}$. We are given N such classifiers corresponding to the classes $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_N$ such that

$$P^n_{X,Y}\left[L(\hat{\phi}_i) - \min_{\phi \in \mathcal{H}_i} L(\phi) > \epsilon\right] \leq \delta_i$$

where $\hat{\phi}_i$ minimizes $\hat{L}(.)$ over $\mathcal{H}_i$. Our objective is to "fuse" the classifier outputs so that the fused system performs at least as well as the best individual classifier based on the sample only. We next describe a method based on the isolation property that enables us to compare the fused system with the best individual classifier [11]. This method is simple to apply and requires easily satisfiable criteria.

4.1 Single Classifier

The lowest possible error achievable by any deterministic classifier is given by the *Bayes error* $L(\phi^*)$, where $\phi^* : \Re^d \mapsto \{0, 1\}$ is defined as

$$\phi^*(x) = \begin{cases} 1 \text{ if } P_{X,Y}[Y = 1 | X = x] \geq P_{X,Y}[Y = 0 | X = x] \\ 0 \text{ otherwise} \end{cases}$$

Since the distribution is not known, ϕ^* cannot be computed. The performance of $\hat{\phi}$ that minimizes $\hat{L}(.)$ can be characterized using the properties of $\mathcal{H}$.

Let $\mathcal{A}$ be a collection of measurable sets of R^d. For $(z_1, z_2, \ldots, z_n) \in \{\Re^d\}^n$, let $\mathcal{N}_{\mathcal{A}}(z_1, z_2, \ldots, z_n)$ denote the number of different sets in $\{\{z_1, z_2, \ldots, z_n\} \cap A : A \in \mathcal{A}\}$. The nth *shatter coefficient* of $\mathcal{A}$ is

$$s(\mathcal{A}, n) = \max_{(z_1, z_2, \ldots, z_n) \in \{\Re^d\}^n} \mathcal{N}_{\mathcal{A}}(z_1, z_2, \ldots, z_n).$$

Then, the *Vapnik-Chervonenkis* (VC) dimension of $\mathcal{A}$, denoted by $V_{\mathcal{A}}$, is the largest integer $k \geq 1$ such that $s(\mathcal{A}, k) = 2^k$. The following important identity [21] relates the shatter coefficient to VC dimension:

$$s(\mathcal{A}, n) = \begin{cases} 2^n & \text{if } n \leq V_{\mathcal{A}} \\ 2\frac{n^{V_{\mathcal{A}}}}{V_{\mathcal{A}}!} & \text{if } n > V_{\mathcal{A}} \end{cases}$$

Then we have $P^n_{X,Y}\left[\sup_{\phi \in \mathcal{A}} |\hat{L}(\phi) - L(\phi)| > \epsilon\right] \leq 8s(\mathcal{A}, n)e^{-n\epsilon^2/32}$. which in turn implies $P^n_{X,Y}\left[L(\hat{\phi}) - \min_{\phi \in H} L(\phi)| > \epsilon\right] \leq 8s(\mathcal{H}, n)e^{-n\epsilon^2/128}$. Thus, given a sample of size $n = \frac{128}{\epsilon^2}(\ln s(\mathcal{H}, n) + \ln(8/\delta))$ we have

$$P^n_{X,Y}\left[L(\hat{\phi}) - \min_{\phi \in \mathcal{H}} L(\phi) > \epsilon\right] < \delta,$$

irrespective of the distribution $P_{X,Y}$.

4.2 Isolation Fusers

We consider a family of fuser functions $\mathcal{F} : \{f : \{0,1\}^N \mapsto \{0,1\}\}$ such that the fused output is given by $f[\hat{\phi}_1(X), \hat{\phi}_2(X), \ldots, \hat{\phi}_N(X)]$, denoted by $f(Z)$, where $Z = (\hat{\phi}_1(X), \hat{\phi}_2(X), \ldots, \hat{\phi}_N(X))$. The error probability of the fused system is

$$L_F(f) = \int I_{\{f(Z) \neq Y\}} dP_{X,Y}.$$

Note that Z is a deterministic function of X given the sample. For computational convenience, we utilize the following alternative formula

$$L_F(f) = \int [f(Z) - Y]^2 dP_{X,Y}.$$

Note that $|\mathcal{F}| \leq 2^{2^N}$ since $\mathcal{F}$ consists of at most all Boolean functions on N variables. Consider the function class

$$\mathcal{G} = \{f(\phi_1(X), \phi_2(X), \ldots, \phi_N(X)) : \phi_1 \in \mathcal{H}_1, \phi_2 \in \mathcal{H}_2, \ldots, \phi_N \in \mathcal{H}_N\}.$$

Here $f(\phi_1(.), \phi_2(.), \ldots, \phi_N(.))$ specifies a subset of $\Re^d$, and hence $\mathcal{G}$ specifies a family of sets of $\Re^d$.

The fuser is obtained in two steps: (a) a training set (Z_1, Y_1), (Z_2, Y_2), ..., (Z_n, Y_n), where $Z_i = (\hat{\phi}_1(X_i), \hat{\phi}_2(X_i), \ldots, \hat{\phi}_N(X_i))$, is derived from the classifiers and the original sample, and (b) the fuser is derived by minimizing empirical error over $\mathcal{F}$. Let f^* minimize $L_F(.)$ over $\mathcal{F}$. Consider the *empirical error*

$$\hat{L}_F(f) = \frac{1}{n} \sum_{i=1}^{n} [f(Z_i) - Y_i]^2.$$

Let $\hat{f}$ minimize $\hat{L}_F(.)$ over $\mathcal{F}$.

If one of the classifier is to be chosen, the lowest achievable error is given by $\min_{i=1}^{N} L(\phi_i^*)$. Since the classifiers can be correlated in an arbitrary manner, the empirically best classifier $\hat{\phi}_{\min} = \arg\min_i \hat{L}(\hat{\phi}_i)$ yields the following guarantee

$$P_{X,Y}^n \left[L(\hat{\phi}_{\min}) - \min_{i=1}^{N} L(\phi_i^*) > \epsilon \right] < \delta_1 + \delta_2 + \ldots + \delta_N.$$

The fuser, thus, provides a *better guarantee* if $\delta_F < \delta_1 + \delta_2 + \ldots + \delta_N$ where

$$P_{X,Y}^n \left[L_F(\hat{f}) - \min_{i=1}^{N} L(\phi_i^*) > \epsilon \right] < \delta_F.$$

The fuser class $\mathcal{F}$ satisfies the *isolation property* [17] if it contains the following N functions: for all $i = 1, 2, \ldots, N$ we have $f_i(z_1, z_2, \ldots, z_N) = z_i$. This property is trivially satisfied if $\mathcal{F}$ consists of all Boolean functions of N variables.

Although it is sufficient to include N functions in $\mathcal{F}$ to satisfy this property, in general a richer class performs better in practice [11].

If the fuser class $\mathcal{F}$ satisfies the isolation property, then fuser $\hat{f}$ provides better guarantee than the best classifier under the condition $|\mathcal{F}| \leq \frac{1}{2}\sum_{i=1}^{N} \delta_i e^{\epsilon^2 n/2}$ (see [11] for the proof). A minimal realization of this result can be based on $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ as per the isolation property. We wish to emphasize that this fusion method can be easily applied without identifying the best classifier, while still ensuring its performance in the fused system. The above condition can also be expressed in terms of the VC dimensions as follows

$$|\mathcal{F}| \leq 4\sum_{i=1}^{N} \frac{(n)^{V_{\mathcal{H}_i}}}{V_{\mathcal{H}_i}!} e^{-\epsilon^2 63n/128}.$$

by noting that $\delta_i = \frac{8(n)^{V_{\mathcal{H}_i}}}{V_{\mathcal{H}_i}!} e^{-\epsilon^2 n/128}$ for $n > \max(V_{\mathcal{H}_1}, V_{\mathcal{H}_2}, \ldots, V_{\mathcal{H}_N})$ [21].

5 Projective Fusers for Function Estimation

The problem of function estimation based on empirical data arises in a number of disciplines such as statistics, systems theory, and computer science. As a result, there has been a profusion of function estimators, whose performance conditions could be quite involved and beyond the expertise of an average practitioner. Nevertheless, several of these estimators are based on considerable practical and theoretical insights, and it would be most desirable to retain their strengths.

We are required to estimate a function $f : [0,1]^d \mapsto [0,1]$, based on a finite *sample* $(X_1, f(X_1)), (X_2, f(X_2)), \ldots, (X_l, f(X_l))$ where $X_1, X_2, \ldots, X_l$, for $l < \infty$, are iid according to an *unknown* distribution P_X on $[0,1]^d$. For an estimator $\hat{f}$ of f we consider the *expected square error* given by

$$I(\hat{f}) = \int (f(X) - \hat{f}(X))^2 dP_X.$$

We are given N previously computed function estimators (as in [14]) each obtained by using an existing method. The individual estimator $\hat{f}_i$ could be a potential function estimator, radial basis function, k-nearest neighbor estimator, regressogram, kernel estimator, regression tree or another estimator.

Given the estimators $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_N$, we consider that the *fuser* is a function $f_F : [0,1]^N \mapsto [0,1]$ such that $f_F(X, \hat{f}_1(X), \hat{f}_2(X), \ldots, \hat{f}_N(X))$ is the *fused estimate* of $f(X)$. The *expected* and *empirical errors* of the fuser are respectively given by

$$I(f_F) = \int [f(X) - f_F(X, \hat{f}_1(X), \hat{f}_2(X), \ldots, \hat{f}_N(X))]^2 dP_X$$

$$\hat{I}(f_F) = \frac{1}{l}\sum_{i=1}^{l} [f(X_i) - f_F(X_i, \hat{f}_1(X_i), \hat{f}_2(X_i), \ldots, \hat{f}_N(X_i))]^2.$$

5.1 Class of Projective Fusers

A *projective fuser* [15], f_P, corresponding to a *partition* $P = \{\pi_1, \pi_2, \ldots, \pi_k\}$, $k \leq N$, of input space $[0,1]^d$ of X ($\pi_i \subseteq [0,1]^d$, $\bigcup_{i=1}^{k} \pi_i = [0,1]^d$, and $\pi_i \cap \pi_j = \phi$ for $i \neq j$), assigns to each block π_i to an estimator $\hat{f}_j$ such that

$$f_P(X, \hat{f}_1, \ldots, \hat{f}_N) = \hat{f}_j(X)$$

for all $X \in \pi_i$. For simplicity, we denote $f_P(X, \hat{f}_1, \ldots, \hat{f}_N)$ by $f_P(X)$. An *optimal projective fuser,* denoted by f_{P^*}, minimizes $I(.)$ over all projective fusers corresponding to all partitions of $[0,1]^d$ and assignments of blocks to estimators.

We define the *error curve* of the estimator $\hat{f}$ for f as $\mathcal{E}(X, \hat{f}) = (f(X) - \hat{f}(X))^2$. The projective fuser based on *lower envelope of error curves* is defined by

$$f_{LE}(X, \hat{f}_1, \ldots, \hat{f}_N) = \hat{f}_{i_{LE}(X)}(X)$$

where $i_{LE}(X) = \arg \min_{i=1,2,\ldots,N} \mathcal{E}(X, \hat{f}_i)$. In other words, $f_{LE}(X, \hat{f}_1, \ldots, \hat{f}_N)$ simply outputs the estimator with the lowest error at X. Thus, we have $\mathcal{E}(X, f_{LE}) = \min_{i=1}^{N} \mathcal{E}(X, \hat{f}_i)$, or equivalently the error curve of f_{LE} is the lower envelope with respect to X of the set of error curves $\{\mathcal{E}(X, \hat{f}_1), \ldots, \mathcal{E}(X, \hat{f}_N)\}$.

5.2 Nearest Neighbor Projective Fuser

We partition the space of X into Voronoi regions $V(X_1), V(X_2), \ldots, V(X_l)$ such that

$$V(X_j) = \{X : \| X - X_j \| < \| X - X_k \| \text{ for all } k = 1, 2, \ldots, l; k \neq j\}$$

where $\| \, . \, \|$ is the Euclidean metric. The points equidistant from more than one sample point are arbitrarily assigned to one of the regions. We assume that all X_i's are distinct without the loss of generality. $V(X_j)$ is simply the set of all points that are at least as close to X_j as to any other X_k. Let $NN(X) = k$ such that $X \in V(X_k)$ for some k, which is the Voronoi cell that X belongs to. For the cell $V(X_{NN(X)})$ that contains X, we identity the estimator that achieves the lowest empirical error at the sample point $X_{NN(X)}$ by defining the *estimator index* of X as follows

$$i_{NN}(X) = \arg \min_{i=1,2,\ldots,N} [f(X_{NN(X)}) - \hat{f}_i(X_{NN(X)})]^2.$$

That is, $i_{NN}(X)$ is the index of the estimator that achieves least empirical error at the sample point $X_{NN(X)}$ nearest to X. Then the *nearest neighbor projective fuser* [19] is defined as

$$\hat{f}_{NN}(X, \hat{f}_1(X), \ldots, \hat{f}_N(X)) = \hat{f}_{i_{NN}(X)}(X).$$

Despite the notational complexity, the idea of $\hat{f}_{NN}$ is quite simple: $\hat{f}_{NN}(X)$ is $\hat{f}_i(X)$ that achieves least empirical error at the nearest sample point to X.

5.3 Sample-Based Projective Fusers

The computation of f_{LE} in general requires a complete knowledge of the distribution P_X. To address the case where such knowledge is not available, a method was proposed in [15] that utilizes regression estimation methods to compute an estimator $\hat{\mathcal{E}}(X, \hat{f}_i)$ of $\mathcal{E}(X, \hat{f}_i)$, and utilizes the lower envelope of these estimators in the computation of fuser. We now briefly outline the basic approach using the cubic partitions with data-dependent offsets for $d = 1$. For a sequence $\{h_l\}$ of positive numbers, consider the partition of $\Re$ given by $\theta_l = \{[(r-1)h_l, rh_l) | r \in \mathcal{Z}\}$. Let $\psi_l[X]$ denote the unique cell of θ_l that contains X. Then, the estimator of $\mathcal{E}(X, \hat{f}_i)$ is given by

$$\hat{\mathcal{E}}(X, \hat{f}_i) = \frac{\sum_{j=1}^{l} (X_j - \hat{f}_i(X_j))^2 1_{\psi_l[X]}(X_j)}{\sum_{j=1}^{n} 1_{\psi_l[X]}(X_j)}.$$

In other words, the estimator simply computes the mean of the error of $\hat{f}_i$ within the cell of θ_l that contains X. Consider the conditions: (i) $((X - f(Y))^2 < K$ for some $K > 0$; (ii) $\lim_{l \to \infty} h_l \to 0$; and (iii) $nh_l \to \infty$ as $l \to \infty$. Then, we have $\int |\mathcal{E}(X, \hat{f}_i) - \hat{\mathcal{E}}(X, \hat{f}_i)|^2 dP_X \to 0$ with probability 1 [8], regardless of the distribution P_X. The fuser $\hat{f}_{LE}$ is computed using $\hat{\mathcal{E}}(X, \hat{f}_i)$ in place of $\mathcal{E}(X, \hat{f}_i)$. The strong consistency of $\hat{f}_{LE}$ method is shown under the boundedness of $(X - f(X))^2$, namely $I(\hat{f}_{LE}) \to I(f_{LE})$ as $l \to \infty$ with probability 1 for any distribution P_X [15]. This result specifies the performance of $\hat{f}_{LE}$ for sample sizes approaching infinity and does not tell much when sample sizes are finite. The implementation of $\hat{f}_{LE}$ itself is tricky in that the choice of h_l is not evident if finite-sample performance is needed.

The individual function estimators $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_N$ could be quite varied, but several of them satisfy certain smoothness or non-smoothness conditions. For any function $g : [-A, A]^d \mapsto \Re$, let $\| g(r) \|_\infty = \sup_{r \in [-A,A]^d} |g(r)|$. A function $g(y) : [-A, A]^d \mapsto \Re^a$ is *Lipschitz* with constant k_g if for all $y_1, y_2 \in [-A, A]^d$, we have $\| g(y_1) - g(y_2) \|_\infty \leq k_g \| y_1 - y_2 \|_\infty$. The examples of smooth function estimators include potential functions, sigmoid neural networks, smooth kernel estimates, radial basis functions, linear and polynomial estimators.

Several function estimators are not Lipschitz with popular examples including nearest neighbor and Nadaraya-Watson estimators. To address such cases we consider the class of functions with bounded variation, which allows for discontinuities and includes Lipschitz functions as a subclass. Consider one-dimensional function $h : [-A, A] \mapsto \Re$. For $A < \infty$, a set of points $P = \{y_0, y_1, \ldots, y_n\}$ such that $-A = y_0 < y_1 < \ldots < y_n = A$ is called a *partition* of [–*A*, *A*]. The collection of all possible partitions of [–*A*, *A*] is denoted by $\mathcal{P}[-A, A]$. A function $g : [-A, A] \mapsto \Re$ is of *bounded variation,* if there exists the total variation M such that for any partition $P = \{y_0, y_1, \ldots, y_n\}$, we have $\sum_{k=1}^{n} |g(y_k) - g(y_{k-1})| \leq M$. A

multivariate function $g : [-A, A]^d \mapsto \Re$ is of bounded variation if it is so in each of its input variable for every value of the other input variables. The following are useful facts about the functions of bounded variation: (i) not all continuous functions are of bounded variation, e.g. $g(y) = y\cos(\pi/(2y))$ for $y \neq 0$ and $g(0) = 0$; (ii) differentiable functions on compact domains are of bounded variation; and (iii) absolutely continuous functions, which include Lipschitz functions, are of bounded variation.

The function estimators such as k-nearest neighbor, Haar wavelet estimators, regression tree, regressogram and Nadaraya-Watson estimator (which all could involve discrete jumps) satisfy the bounded variation property. Since Lipschitz estimators over compact domains also have bounded variation, the latter is a fairly general property satisfied by most of the widely-used estimators.

We consider that the function estimators $\hat{f}_1, \ldots, \hat{f}_N$ are of bounded variation. Let each function estimator $\hat{f}_i$ be of total variation V_i. For $V = \sum_{i=1}^{N} V_i$, it is shown in [19] that $P\left[I(\hat{f}_{NN}) - I(f_{LE}) > \epsilon\right] < \delta$ for sample size

$$\frac{256}{\epsilon^2}\left[18\left(1 + \frac{128V}{\epsilon}\right)\ln^2(128/\epsilon) + \ln(16/\delta)\right].$$

Furthermore, $I(\hat{f}_{NN}) \to I(f_{LE})$ as $l \to \infty$. This result establishes the analytical viability of $\hat{f}_{NN}$ for finite samples. While the sample size estimate is not necessarily within practical limits, the overall result itself is stronger than the asymptotic consistency.

5.4 Computational Example

We consider the problem of estimating

$$f(X) = 0.02(12 + 3X + 7.2x^2)(1.0 + \cos(4\pi X))(1.0 + 0.8\sin(3\pi X/7))$$

based on a sample. Two samples each of size 200 (Fig. 1(a)) are used in training the neural networks and fuser. Five feedforward neural networks are trained using the backpropagation algorithm with different starting weights and different learning rates as shown in Fig. 1(b). The performance of the estimators and fuser is measured by the empirical error on the sample. The estimator 1 approximated well only in the vicinity of $X = 1$, whereas estimator 2 is close to the function in the vicinity of $X = 0$. Estimator 3 provided a good approximation at both ends of the interval [0,1] and is the best of the estimators. However, this estimator is insensitive to the variations of $f(X)$ in the middle of the interval [0,1]. Estimator 4 performs the worst staying close to 0 for entire [0,1]. The performance of $\hat{f}_{NN}$ is shown in Figure 1(c) which is uniformly as good as any of the estimators across the entire interval. The best estimator 3 is used by the fuser for the most of the interval [0,1] except in the middle. It is interesting to note that the worst estimator, namely, estimator 4, is used in the lowest portions of $f(X)$, and indeed is responsible for the better performance achieved by the fuser.

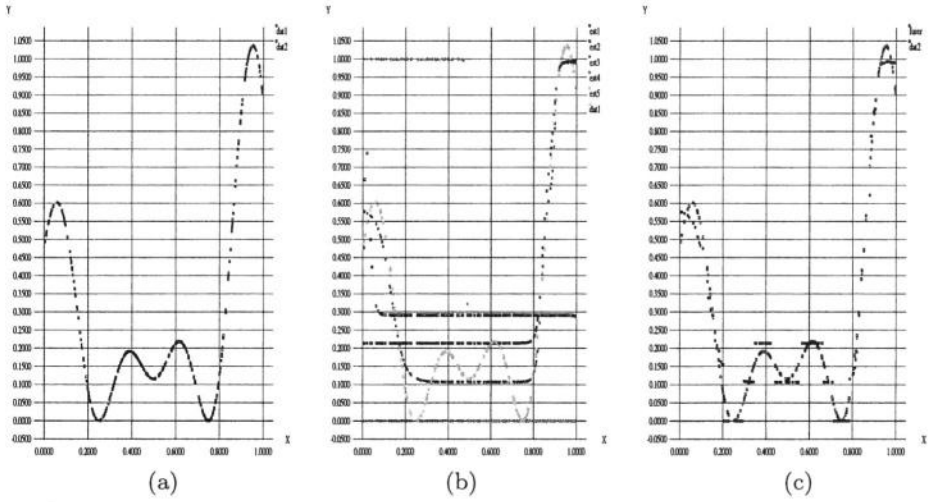


Fig. 1. Nearest neighbor projective fuser for function estimators.

6 Conclusions

A generic sensor fusion problem is formulated for sensors whose measurements are subject to unknown probability distributions. A brief overview of fuser design methods is presented with a focus on finite sample performance guarantees. The classes of isolation and projective fusers are described for the special cases of classification and function estimation. Similar concepts have been studied in multiple classifier systems [3,23]. The methods described in this paper have been applied in practice for combining ultrasonic and infrared sensor measurements for robot navigation, prediction of embrittlement levels in light water reactors, combining sensor readings of well data in methane hydrate explorations, and combining radar measurements for target detection.

Several open problems remain in the generic sensor fusion problem as well as in classification and function estimation. Often the sample bounds are too large to be practical, and the performance equations do not provide uniform precision in that the sensor with best bound is not necessarily the best. It would be interesting to develop principles that bridge the gap between performance bounds and actual performance. Also there has been a profusion of fusion concepts of significant diversity, and it would be interesting to identify unifying principles behind these developments.

Acknowledgments

This research is sponsored by the Engineering Research Program of the Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

References

1. C. K. Chow. Statistical independence and threshold functions. *IEEE Trans. Electronic Computers,* EC-16:66–68, 1965.
2. L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, New York, 1996.
3. G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behavior. *Pattern Recognition,* 34:1879–1881, 2001.
4. B. Grofman and G. Owen, editors. *Information Pooling and Group Decision Making.* Jai Press Inc., Greenwich, Connecticut, 1986.
5. J. Kittler and F. Roli, editors. *Multiple Classifier Systems,* volume 1857. Springer-Verlag, Berlin, 2000.
6. V. Kurkova. Kolmogorov's theorem and multilayer neural networks. *Neural Networks,* 5:501–506, 1992.
7. R. N. Madan and N. S. V. Rao. Guest editorial on information/decision fusion with engineering applications. *Journal of Franklin Institute,* 336B(2), 1999. 199-204.
8. A. Nobel. Histogram regression estimation using data-dependent partitions. *Annals of Statistics,* 24(3): 1084–1105, 1996.
9. N. S. V. Rao. Distributed decision fusion using empirical estimation. *IEEE Transactions on Aerospace and Electronic Systems,* 33(4):1106–1114, 1996.
10. N. S. V. Rao. Fusion methods in multiple sensor systems using feedforward neural networks. *Intelligent Automation and Soft Computing,* 5(1):21–30, 1998.
11. N. S. V. Rao. To fuse or not to fuse: Fuser versus best classifier. In *SPIE Conference on Sensor Fusion: Architectures, Algorithms, and Applications II,* pages 25–34. 1998.
12. N. S. V. Rao. Vector space methods for sensor fusion problems. *Optical Engineering,* 37(2):499–504, 1998.
13. N. S. V. Rao. Multiple sensor fusion under unknown distributions. *Journal of Franklin Institute,* 336(2) :285–299, 1999.
14. N. S. V. Rao. On optimal projective fusers for function estimators. In *Second International Conference on Information Fusion,* pages 296–301. 1999.
15. N. S. V. Rao. Projective method for generic sensor fusion problem. In *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems,* pages 1–6. 1999.
16. N. S. V. Rao. Finite sample performance guarantees of fusers for function estimators. *Information Fusion,* 1(1):35–44, 2000.
17. N. S. V. Rao. On fusers that perform better than best sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 23(8):904–909, 2001.
18. N. S. V. Rao. Multisensor fusion under unknown distributions: Finite sample performance guarantees. In A. K. Hyder, editor, *Multisensor Fusion.* Kluwer Academic Pub., 2002.
19. N. S. V. Rao. Nearest neighbor projective fuser for fucntion estimation. In *Proceedings of International Conference on Information Fusion,* 2002.
20. N. S. V. Rao and S. S. Iyengar. Distributed decision fusion under unknown distributions. *Optical Engineering,* 35(3):617–624, 1996.
21. V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.
22. P. K. Varshney. *Distributed Detection and Data Fusion.* Springer-Verlag, 1997.
23. T. Windeatt and F. Roli, editors. *Multiple Classifier Systems.* Springer-Verlag, Berlin, 2003.

AveBoost2: Boosting for Noisy Data

Nikunj C. Oza

Computational Sciences Division
NASA Ames Research Center
Mail Stop 269-1
Moffett Field, CA 94035-1000, USA
oza@email.arc.nasa.gov

Abstract. AdaBoost [4] is a well-known ensemble learning algorithm that constructs its *base* models in sequence. AdaBoost constructs a distribution over the training examples to create each base model. This distribution, represented as a vector, is constructed with the goal of making the next base model's mistakes uncorrelated with those of the previous base model [5]. We previously [7] developed an algorithm, AveBoost, that first constructed a distribution the same way as AdaBoost but then averaged it with the previous models' distributions to create the next base model's distribution. Our experiments demonstrated the superior accuracy of this approach. In this paper, we slightly revise our algorithm to obtain non-trivial theoretical results: bounds on the training error and generalization error (difference between training and test error). Our averaging process has a regularizing effect which leads us to a worse training error bound for our algorithm than for AdaBoost but a better generalization error bound. This leads us to suspect that our new algorithm works better than AdaBoost on noisy data. For this paper, we experimented with the data that we used in [7] both as originally supplied and with added label noise – some of the data has its original label changed randomly. Our algorithm's experimental performance improvement over AdaBoost is even greater on the noisy data than the original data.

1 Introduction

AdaBoost [4] is one of the most well-known and highest-performing ensemble classifier learning algorithms [3]. It constructs a sequence of base models, where each model is constructed based on the performance of the previous model on the training set. In particular, AdaBoost calls the base model learning algorithm with a training set weighted by a distribution[1]. After the base model is created, it is tested on the training set to see how well it learned. We assume that the base model learning algorithm is a *weak learning algorithm;* that is, with high probability, it produces a model whose probability of misclassifying an example

[1] If the base model learning algorithm cannot take a weighted training set as input, then one can create a sample with replacement from the original training set according to the distribution and call the algorithm with that sample.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 31–40, 2004.

is less than 0.5 when that example is drawn from the same distribution that generated the training set. The point is that such a model performs better than random guessing[2]. The weights of the correctly classified examples and misclassified examples are scaled down and up, respectively, so that the two groups' total weights are 0.5 each. The next base model is generated by calling the learning algorithm with this new weight distribution and the training set. The idea is that, because of the weak learning assumption, at least some of the previously misclassified examples will be correctly classified by the new base model. Previously misclassified examples are more likely to be classified correctly because of their higher weights, which focus more attention on them. AdaBoost scales the distribution with the goal of making the next base model's mistakes uncorrelated with those of the previous base model [5].

AdaBoost is notorious for performing poorly on noisy datasets [3], such as those with label noise – that is, some examples were randomly assigned the wrong class label. Because these examples are inconsistent with the majority of examples, they tend to be harder for the base model learning algorithm to learn. AdaBoost increases the weights of examples that the base model learning algorithm did not learn correctly. Noisy examples are likely to be incorrectly learned by many of the base models so that eventually these examples' weights will dominate those of the remaining examples. This causes AdaBoost to focus too much on the noisy examples at the expense of the majority of the training examples, leading to poor performance on new examples.

We previously [7] presented an algorithm, called AveBoost, which calculates the next base model's distribution by first calculating a distribution the same way as in AdaBoost, but then averaging it elementwise with those calculated for the previous base models. This averaging mitigates AdaBoost's tendency to increase the weights of noisy examples to excess. In our previous work we presented promising experimental results. However, we did not present theoretical results. In our subsequent research, we were unable to derive a non-trivial training error bound for the algorithm presented in [7]. In this paper, we present a slight modification to AveBoost which allows us to obtain both a non-trivial training error bound and a generalization error bound (difference between training error and test error). We call this algorithm AveBoost2. In Section 2, we review AdaBoost. In Section 3, we describe the AveBoost2 algorithm and state how it is different from AveBoost. In Section 4, we present our training error bound and generalization error bound. The averaging in our algorithm has a regularizing effect; therefore, as expected, our training error bound is worse than that of AdaBoost but our generalization error bound is better than AdaBoost's. In Section 5, we present an experimental comparison of our new AveBoost2 with AdaBoost on some UCI datasets [1] both in original form and with 10% label noise added. Section 6 summarizes this paper and describes ongoing and future work.

[2] The version of AdaBoost that we use was designed for two-class classification problems. However, it is often used for a larger number of classes when the base model learning algorithm is strong enough to have an error less than 0.5 in spite of the larger number of classes.

AdaBoost$(\{(x_1, y_1), \ldots, (x_m, y_m)\}, L_b, T)$
Initialize $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$.
For $t = 1, 2, \ldots, T$:
$h_t = L_b(\{(x_1, y_1), \ldots, (x_m, y_m)\}, \mathbf{d}_t)$
Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} d_{t,i}$.
If $\epsilon_t \geq 1/2$ then,
set $T = t - 1$ and abort this loop.
$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
Calculate distribution $\mathbf{d}_{t+1}$:

$$w_i = d_{t,i} \times \begin{cases} \beta_t \text{ if } h_t(x_i) = y_i \\ 1 \quad \text{otherwise} \end{cases}$$

$$d_{t+1,i} = \frac{w_i}{\sum_{i=1}^{m} w_i}$$

Output the final hypothesis:
$h_{fin}(x) = \text{argmax}_{y \in Y} \sum_{t:h_t(x)=y} log \frac{1}{\beta_t}$

Fig. 1. AdaBoost algorithm: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ is the training set, L_b is the base model learning algorithm, and T is the maximum allowed number of base models.

2 AdaBoost

Figure 1 shows AdaBoost's pseudocode. AdaBoost constructs a sequence of base models h_t for $t \in \{1, 2, \ldots, T\}$, where each model is constructed based on the performance of the previous base model on the training set. In particular, AdaBoost maintains a distribution over the m training examples. The distribution $\mathbf{d}_1$ used in creating the first base model gives equal weight to each example ($d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$). AdaBoost now enters the loop, where the base model learning algorithm L_b is called with the training set and $\mathbf{d}_1$. The returned model h_1 is then tested on the training set to see how well it learned. The total weight of the misclassified examples (ϵ_1) is calculated. The weights of the correctly-classified examples are multiplied by $\epsilon_1/(1 - \epsilon_1)$ so that they have the same total weight as the misclassified examples. The weights of all the examples are then normalized so that they sum to 1 instead of $2\epsilon_1$. AdaBoost assumes that L_b is a *weak learner,* i.e., $\epsilon_t < \frac{1}{2}$ with high probability. Under this assumption, the total weight of the misclassified examples $\epsilon_t < 1/2$ is increased to 1/2 and the total weight of the correctly classified examples $1 - \epsilon_t > 1/2$ is decreased to 1/2. This is done so that, by the weak learning assumption, the next model h_{t+1} will classify at least some of the previously misclassified examples correctly. Returning to the algorithm, the loop continues, creating the T base models in the ensemble. The final ensemble returns, for a new example, the one class in the set of classes Y that gets the highest weighted vote from the base models. Each base model's vote is proportional to its accuracy on the weighted training set used to train it.

AveBoost2$(\{(x_1, y_1), \ldots, (x_m, y_m)\}, L_b, T)$
Initialize $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$.
For $t = 1, 2, \ldots, T$:
$h_t = L_b(\{(x_1, y_1), \ldots, (x_m, y_m)\}, \mathbf{d}_t)$
Calculate the error of $h_t : \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} d_{t,i}$.
If $\epsilon_t \geq 1/2$ then,
set $T = t - 1$ and abort this loop.
$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
$\gamma_t = \frac{2(1-\epsilon_t)t+1}{2\epsilon_t t+1}$
Calculate distribution $\mathbf{d}_{t+1}$:
For $i = 1, 2, \ldots, m$:

$$w_i = d_{t,i} \times \begin{cases} \beta_t \text{ if } h_t(x_i) = y_i \\ 1 \quad \text{otherwise} \end{cases}$$

$$c_{t,i} = \frac{w_i}{\sum_{i=1}^{m} w_i}$$

$$d_{t+1,i} = \frac{t d_{t,i} + c_{t,i}}{t+1}$$

Fig. 2. AveBoost2 algorithm: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ is the training set, L_b is the base model learning algorithm, and T is the maximum allowed number of base models.

3 AveBoost2 Algorithm

Figure 2 shows our new algorithm, AveBoost2. Just as in AdaBoost, AveBoost2 initializes $d_{1,i} = 1/m$ for all $i \in \{1, 2, \ldots, m\}$. Then it goes inside the loop, where it calls the base model learning algorithm L_b with the training set and distribution $\mathbf{d_1}$ and calculates the error ϵ_1 of the resulting base model h_1. It then calculates $\mathbf{c_1}$, which is the distribution that AdaBoost would use to construct the next base model. However, AveBoost2 averages this with $\mathbf{d_1}$ to get $\mathbf{d_2}$, and uses this $\mathbf{d_2}$ instead. Showing that the $\mathbf{d_t}$'s in AveBoost2 are distributions is a trivial proof by induction. For the base case, $\mathbf{d_1}$ is constructed to be a distribution. For the inductive part, if $\mathbf{d_t}$ is a distribution, then $\mathbf{d_{t+1}}$ is a distribution because it is a convex combination of $\mathbf{d_t}$ and $\mathbf{c_t}$, both of which are distributions. The vector $\mathbf{d_{t+1}}$ is a running average of $\mathbf{d_1}$ and the vectors $\mathbf{c_q}$ for $q \in \{1, 2, \ldots, t\}$.

Returning to the algorithm, the loop continues for a total of T iterations. Then the base models are combined using a weighted voting scheme slightly different from that of AdaBoost and the original AveBoost from [7]: each model's weight is $log(1/(\beta_t \gamma_t))$ instead of $log(1/\beta_t)$. AveBoost2 is actually AdaBoost with β_t replaced by $\beta_t \gamma_t$. However, we wrote the AveBoost2 pseudocode as we did to make the running average calculation of the distribution explicit.

AveBoost2 can be seen as a relaxed version of AdaBoost. When training examples are noisy and therefore difficult to fit, AdaBoost is known to increase the

weights of those examples to excess and overfit them [3] because many consecutive base models may not learn them properly. AveBoost2's averaging does not allow the weights of noisy examples to increase rapidly, thereby mitigating the overfitting problem. We therefore expect AveBoost2 to outperform AdaBoost on the noisy datasets to a greater extent than on the original datasets. We also expect AveBoost2's advantage to be greater for smaller numbers of base models. When AveBoost2 creates a large ensemble, later training set distributions (and therefore later base models) cannot be too different from each other because they are prepared by averaging over many previous distributions. Therefore, we expect that later models will not have as much of an impact on performance.

4 Theory

In this section, we give bounds on the training error and generalization error (difference between training and test error). Not surprisingly, the relaxed nature of AveBoost2 relative to AdaBoost caused us to obtain a worse training error bound but superior generalization error bound for AveBoost2 relative to AdaBoost. Due to space limitations, we defer the proofs and more intuition on the theoretical frameworks that we use to a longer version of this paper. AveBoost2's training error bound is stated in the following theorem.

Theorem 1. *In AveBoost2, suppose the weak learning algorithm L_b generates hypotheses with errors $\epsilon_1, \epsilon_2, \ldots, \epsilon_T$ where each $\epsilon_t < 1/2$. Then the ensemble's error $\epsilon = \sum_{i:h_{fin}(x_i) \neq y_i} d_{1,i}$ is bounded as follows:*

$$\epsilon \leq \prod_{t=1}^{T} \frac{t+1}{\sqrt{t^2 + \frac{t}{2\epsilon_t(1-\epsilon_t)} + \frac{1}{4\epsilon_t(1-\epsilon_t)}}}.$$

This bound is non-trivial ($\epsilon \leq 1$); but greater than that of AdaBoost [4]:

$$\epsilon \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}.$$

To derive our generalization error bound, we use the algorithmic stability framework of [6]. Intuitively, algorithmic stability is similar to Breiman's notion of stability [2] – the more stable a learning algorithm is, the less of an effect changes to the training set have on the model returned. The more stable the learning algorithm is, the smaller the difference between the training and test errors tends to be, assuming that the training and test sets are drawn from the same distribution. We show that AveBoost2 is more stable than AdaBoost; therefore, the difference between the training and test errors is lower. We first give some preliminaries from [6] and then state our new result.

For the following, $\mathcal{X}$ is the space of possible inputs, $\mathcal{Y} = \{0, 1\}$ is the set of possible labels, and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 1 (Definition 2.5 from [6]). *A* learning algorithm *is a process which takes as input a distribution $\mathbf{p}$ on $\mathcal{Z}$ with finite support and outputs a function $f_p : \mathcal{X} \to [0,1]$. For $S \in \mathcal{Z}^m$ for some positive integer m, f_S means f_p where $\mathbf{p}$ is the uniform distribution on S.*

In the following, the error of f on an example (x, y) is $c(f,(x,y)) = |f(x) - y|$.

Definition 2 (Definition 2.11 from [6]). *A learning algorithm has L_1-stability λ if, for any two distributions $\mathbf{p}$ and $\mathbf{q}$ on $\mathcal{X}$ with finite support,*

$$\forall z \in \mathcal{Z}, |c(f_p, z) - c(f_q, z)| \leq \lambda \|p - q\|_1.$$

In the following, D is a distribution on $\mathcal{Z}$, $S \sim D^m$ is a set of m examples drawn from $\mathcal{Z}$ according to D, and $S^{i,u}$ is S with example $i \in \{1, 2, \ldots, m\}$ removed (each i is chosen with probability $1/m$) and example $u \sim D$ added.

Definition 3 (Definition 2.14 from [6]). *A learning algorithm is (β, δ)-stable if*

$$P_{S \sim D^m}(|c(f_S, z) - c(f_{S^{i,u}}, z)| \leq \beta) \geq 1 - \delta.$$

Intuitively, f_S and $f_{S^{i,u}}$ are models that result from running the learning algorithm on two slightly different training sets. As β and δ decrease, the probability of having smaller differences in errors between these two models increases, which means that the learning algorithm is more stable. Greater stability implies lower generalization error according to the following theorem. In the following, $Err_S(f_S)$ is the training error (error on the training set S) and $Err_D(f_S)$ is the test error, i.e., the error on an example (x, y) chosen at random according to distribution D.

Theorem 2 (Theorem 3.4 from [6]). *Suppose a (β, δ)-stable learning algorithm returns a hypothesis f_S for any training set $S \sim D^m$. Then for all $\tau > 0$ and $m \geq 1$,*

$$P_{S \sim D^m}(|Err_S(f_S) - Err_D(f_S)| > \tau + \beta + \delta) \leq 2exp\left(\frac{-\tau^2 m}{2(2m\beta + 1)^2}\right) + \frac{4m^2\delta}{2m\beta + 1}.$$

Intuitively, this theorem shows that lower values of β and δ lead to lower probabilities of large differences between the training and test errors. We can finally state our theorem on AveBoost2's stability.

Theorem 3. *Suppose the base model learning algorithm has L_1-stability λ and $\min_{t \in \{1,2,\ldots,T\}} \epsilon_t > \epsilon_* > 0$. Then, for sufficiently large m and for all T, AveBoost2 is (β, δ)-stable, where*

$$\beta = \frac{2}{m} \sum_{t=1}^{T} 2^{3t-2} \left(\frac{(\lambda + 1)t}{\epsilon_*}\right)^t$$

$$\delta = exp\left(\frac{-m\epsilon_*^2}{2}\right).$$

Table 1. The datasets used in the experiments.

Data Set	Training Set	Test Set	Inputs	Classes
Promoters	84	22	57	2
Balance	500	125	4	3
Breast Cancer	559	140	9	2
German Credit	800	200	20	2
Car Evaluation	1382	346	6	4
Chess	2556	640	36	2
Mushroom	6499	1625	22	2
Nursery	10368	2592	8	5
Connect4	54045	13512	42	3

Table 2. Performance of AveBoost2 compared to AdaBoost.

ORIGINAL	Num. Base Models			10% NOISE	Num. Base Models		
Base Model	10	50	100	Base Model	10	50	100
Naive Bayes	+4=4-1	+4=4-1	+4=4-1	Naive Bayes	+8=1-0	+8=1-0	+7=2-0
Decision Trees	+2=6-1	+1=6-2	+2=6-1	Decision Trees	+6=2-1	+5=3-1	+6=2-1
Decision Stumps	+1=6-2	+1=6-2	+1=6-2	Decision Stumps	+0=7-2	+1=7-1	+1=7-1

For AdaBoost, the theorem is the same except that [6]

$$\beta = \frac{2}{m} \sum_{t=1}^{T} \frac{2^{t^2+1}(\lambda+1)^t}{\epsilon_*^{2t-1}},$$

which is larger than the corresponding β for AveBoost2. This means that AveBoost2's generalization error is less than that of AdaBoost by Theorem 2.

Note that as ϵ_* decreases toward zero, the β and δ for both AdaBoost and AveBoost increase, which means both algorithms are less stable. This makes sense because, as ϵ_* decreases, the upper bounds on the weights assigned to each base model ($log\frac{1}{\beta_t}$ and $log\frac{1}{\beta_t \gamma_t}$) increase. This means that each individual base model's potential influence on the ensemble's result is higher, so that changes in the individual base models lead to greater changes in the ensemble's predictions.

5 Experimental Results

In this section, we compare AdaBoost and AveBoost2 on the nine UCI datasets [1] described in Table 1. We ran both algorithms with three different values of T, which is the maximum number of base models that the algorithm is allowed to construct: 10, 50, and 100. Each result reported is the average over 50 results obtained by performing 10 runs of 5-fold cross-validation. Table 1 shows the sizes of the training and test sets for the cross-validation runs. We also repeated these runs after adding 10% label noise. That is, we randomly chose 10% of the

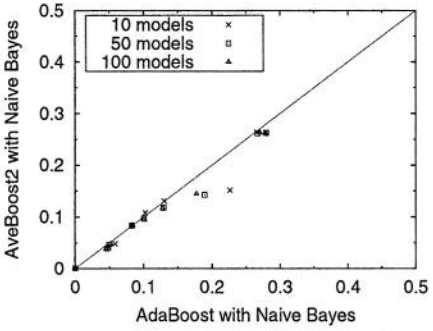


Fig. 3. Test set error rates of AdaBoost vs. AveBoost2 (Naive Bayes, original datasets).

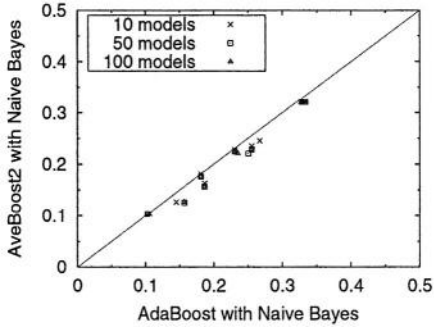


Fig. 4. Test set error rates of AdaBoost vs. AveBoost2 (Naive Bayes, noisy datasets).

examples in each dataset and changed their labels to one of the remaining labels with equal probability.

Table 2 shows how often AveBoost2 significantly outperformed, performed comparably with, and significantly underperformed AdaBoost. For example, on the original datasets and with 10 Naive Bayes base models, AveBoost2 significantly outperformed[3] AdaBoost on four datasets, performed comparably on four datasets, and performed significantly worse on one, which is written as "+4=4-1." Figures 3 and 4 compare the error rates of AdaBoost and AveBoost2 with Naive Bayes base models on the original and noisy datasets, respectively. In all the plots presented in this paper, each point marks the error rates of two algorithms when run with the number of base models indicated in the legend and a particular dataset. The diagonal lines in the plots contain points at which the two algorithms have equal error. Therefore, points below/above the line correspond to the error of the algorithm indicated on the y-axis being less than/greater than the error of the algorithm indicated on the x-axis, respectively. For Naive Bayes base models, AveBoost2 performs much better than AdaBoost overall, especially on the noisy datasets.

We compare AdaBoost and AveBoost2 using decision tree base models in figure 5 (original datasets) and figure 6 (noisy datasets). On the original datasets, the performances of the two algorithms are comparable. However, on the noisy datasets, AveBoost2 is superior for all except the Balance dataset. On the Balance dataset, AdaBoost actually performed as much as 10% better on the noisy data than the original data, which is strange, and needs to be investigated further. On the other hand, AveBoost2 performed worse on the noisy Balance data than on the original Balance data. Figure 7 gives the error rate comparison between AdaBoost and AveBoost2 with decision stump base models on the original datasets. Figure 8 gives the same comparison on the noisy datasets. With decision stumps, the two algorithms always seem to perform comparably. We suspect

[3] We use a t-test with $\alpha = 0.05$ to compare all the classifiers in this paper.

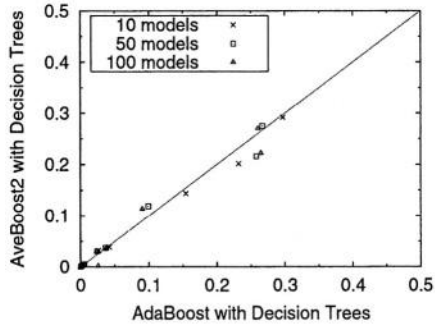


Fig. 5. Test set error rates of AdaBoost vs. AveBoost2 (Decision Trees, original datasets).

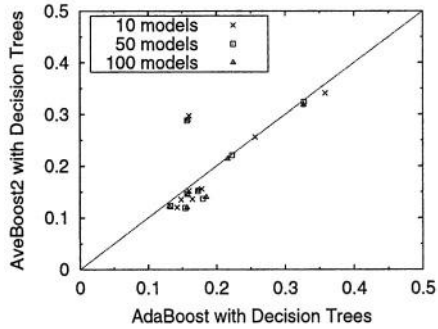


Fig. 6. Test set error rates of AdaBoost vs. AveBoost2 (Decision Trees, noisy datasets).

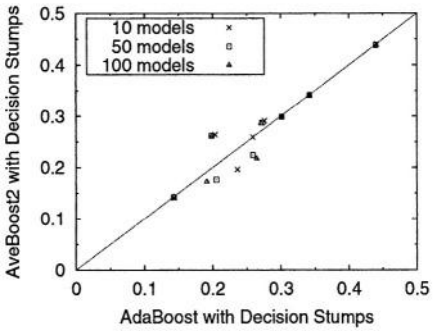


Fig. 7. Test set error rates of AdaBoost vs. AveBoost2 (Decision Stumps, original datasets).

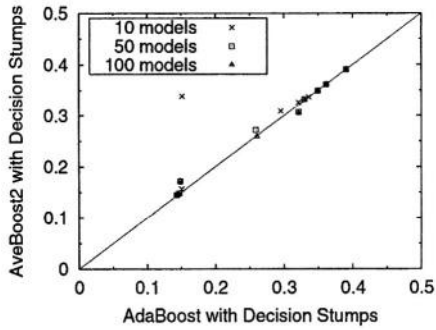


Fig. 8. Test set error rates of AdaBoost vs. AveBoost2 (Decision Stumps, noisy datasets).

that decision stumps are too stable to allow the different distribution calculation methods of AdaBoost and AveBoost2 to yield significant differences. In all the results, we did not see the hypothesized differences in performance as a function of the number of base models.

6 Conclusions

We presented AveBoost2, a boosting algorithm that trains each base model using a training example weight vector that is based on the performances of all the previous base models rather than just the previous one. We discussed our theoretical results and demonstrated empirical results that are superior overall to AdaBoost; especially on datasets with label noise.

Our theoretical and empirical results do not account for what happens as the amount of noise in the data changes. We plan to derive such results. In

a longer version of this paper, we plan to perform a more detailed empirical analysis including the performances of the base models and ensembles on the training and test sets, correlations among the base models, ranges of the weights of regular and noisy examples, etc. In [7], we performed such an analysis to a limited extent for the original AveBoost and were able to confirm some of what we hypothesized there and in this paper: the base model accuracies tend to be higher than for AdaBoost, the correlations among the base models also tend to be higher, and the ranges of the weights of the training examples tends to be lower. We were unable to repeat this analysis here due to a lack of space. We also plan to compare our algorithm to other efforts to make boosting work better with noisy data, such as identifying examples that have consistently high weight as noisy and; therefore, untrustworthy [8].

References

1. C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1999. (URL: http://www.ics.uci.edu/~mlearn/MLRepository.html).
2. L. Breiman. Bagging predictors. *Machine Learning,* 24(2):123–140, 1996.
3. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning,* 40:139–158, Aug. 2000.
4. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning,* pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.
5. Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory,* pages 134–144, 1999.
6. Samuel Kutin and Partha Niyogi. The interaction of stability and weakness in adaboost. Technical Report TR-2001-30, University of Chicago, October 2001.
7. Nikunj C. Oza. Boosting with averaged weight vectors. In T. Windeatt and F. Roli, editors, *Proceedings of the Fourth International Workshop on Multiple Classifier Systems,* pages 15–24. Springer, Berlin, 2003.
8. G. Rätsch, T. Onoda, and K.R. Müller. Soft margins for adaboost. *Machine Learning,* 42:287–320, 2001.

Bagging Decision Multi-trees*

Vicent Estruch, César Ferri,
José Hernández-Orallo, and Maria José Ramírez-Quintana

DSIC, Univ. Politècnica de València , Camì de Vera s/n, 46020 València, Spain
{vestruch.cferri,jorallo,mramirez}@dsic.upv.es

Abstract. Ensemble methods improve accuracy by combining the predictions of a set of different hypotheses. A well-known method for generating hypothesis ensembles is Bagging. One of the main drawbacks of ensemble methods in general, and Bagging in particular, is the huge amount of computational resources required to learn, store, and apply the set of models. Another problem is that even using the bootstrap technique, many simple models are similar, so limiting the ensemble diversity. In this work, we investigate an optimization technique based on sharing the common parts of the models from an ensemble formed by decision trees in order to minimize both problems. Concretely, we employ a structure called decision multi-tree which can contain simultaneously a set of decision trees and hence consider just once the "repeated" parts. A thorough experimental evaluation is included to show that the proposed optimisation technique pays off in practice.

Keywords: Ensemble Methods, Decision Trees, Bagging.

1 Introduction

With the goal of improving model accuracy, there has been an increasing interest in defining methods that combine hypotheses. These methods construct a set of hypotheses (ensemble), and then combine the components of the ensemble in some way (typically by a weighted or unweighted voting) in order to classify examples. The accuracy obtained will be often better than that of the individual components of the ensemble. This technique is known as *Ensemble Methods* [3].

This accuracy improvement of ensemble methods can be intuitively justified because the combined model represents an increase in expressiveness over the single components of the ensemble, and the fact that the combination of uncorrelated errors avoids over-fitting. The quality of the generated ensemble highly depends on the accuracy and diversity of its individual components [9].

Many methods have been proposed to construct a set of classifiers from a single evidence. These techniques have been applied in many different learning algorithms. Dietterich [3] distinguishes different kinds of ensemble construction methods, being probably the methods based on the manipulation of the training

* This work has been partially supported by CICYT under grant TIC2001-2705-C03-01 and MCyT Acción Integrada HU 2003-0003.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 41–51, 2004.

examples the most frequently used. The common idea of this kind of methods boils down to several times the same learning algorithm, each time with a different subset or weighting of the training examples, thus generating a different classifier for each set. The relevant issue in these methods is to define a good mechanism to generate subsets from the set of training examples. For instance, *Bagging* [2,16], *Boosting* [8,16] and *Cross-validated committees* [13] are ensemble methods of this family.

Bagging is derived from the technique known as *bootstrap aggregation.* This method constructs subsets by generating a sample of m training examples, selected randomly (and with replacement) from the original training set of m instances. The new subsets of training examples are called *bootstrap replicates.*

There have been some works that compare the performance of ensemble methods. Dietterich has compared *Bagging, Boosting* and *Randomisation* ensemble methods experimentally in [4]. The conclusions are that in problems without noise *Boosting* gets the best results, while the results of *Bagging* and *Randomisation* are quite similar. With respect to noisy datasets, *Bagging* is the best method, followed by *Randomisation,* and, finally, *Boosting.*

Ensemble methods have also important drawbacks. Probably, the most important problem is that they require the generation, storage, and application of a set of models in order to predict future cases. This represents an important consumption of resources, in both scenarios: learning process and predicting new cases. This important hindrances frequently limit the application of ensemble methods to real problems.

Let us consider the following scenario, given the classical *playtennis* [11] problem; we construct an ensemble of four decision trees by applying *Bagging,* over the C4.5 decision-tree learning algorithm, i.e., we learn four decision trees with C4.5 from four different bootstrap replicates. The four trees learned are shown in Figure 1.

If we observe the four decision trees, we can appreciate that there are many similarities among them. For instance, Decision Tree 1 and Decision Tree 2, as well as Decision Tree 3 and Decision Tree 4, have the same condition at the root. More concretely, Decision Tree 1 and Decision Tree 2 are almost identical, the only difference between both trees is that Decision Tree 1 has an additional split in a node considered as a leaf in Decision Tree 2.

Furthermore, the first consequence of this phenomenon is that most of the solutions are similar, and hence, the errors can be correlated. All these patterns and regularities are not considered when learning ensembles of decision trees, and therefore, this process is also usually expensive in terms of computational cost. Motivated by these two problems, we present in this work an algorithm that exploits these regularities, and therefore, it allows a better dealing of resources when learning ensembles of decision trees. We employ a structure called decision multi-tree that can contain simultaneously a set of decision trees sharing their common parts. In previous works [6], we developed the idea of options trees [10] into the multi-structure, using a beam-search method to populate the multi-tree, mostly based on the randomisation technique, and several fusion techniques to

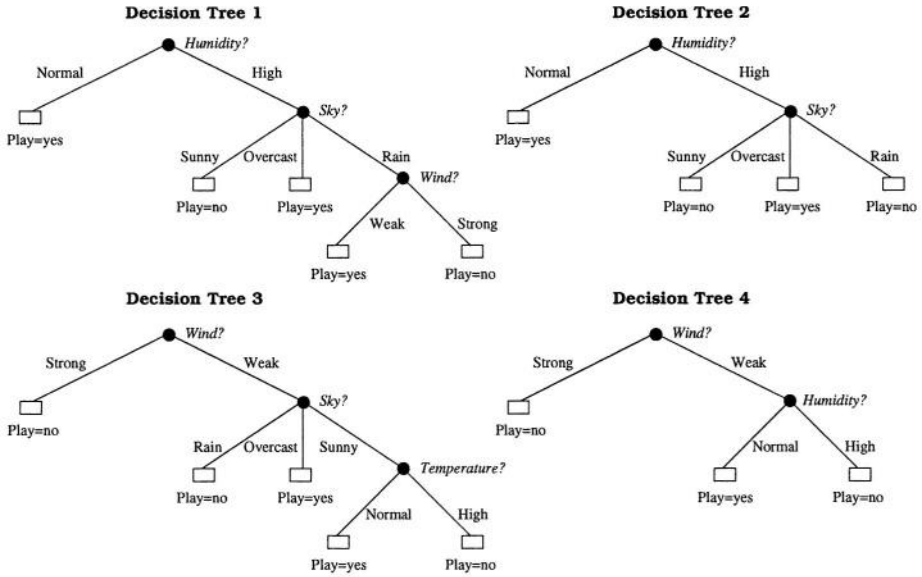


Fig. 1. Four different decision trees for the playtennis problem.

merge the solutions in the multi-tree. One of the most delicate things of our previous approach was the choice of alternate splits. The use of *Bagging* in multi-trees solves this problem and, furthermore, it can allow a fairer comparison between the multi-tree and other ensemble methods. Although the presented algorithm is based on *Bagging,* the same idea could be easily applied to other ensembles methods such as *Boosting.*

The paper is organised as follows. First, in Section 2, we introduce the decision multi-tree structure. Section 3 introduces the algorithm that allows a decision multi-tree to be constructed by employing bootstrap replicates. A thorough experimental evaluation is included in Section 4. Finally, the last section presents the conclusions and proposes some future work.

2 Decision Multi-trees

In this section we present the decision multi-tree structure. This structure can be seen as a generalisation of a classical decision tree. Basically, a decision multi-tree is a tree in which the rejected splits are not removed, but stored as *suspended* nodes. The further exploration of these nodes after the first solution is built permits the generation of new models. For this reason, we call this structure decision multi-tree, rather than a decision tree. Since each new model is obtained by continuing the construction of the multi-tree, these models share their common parts.

Likewise, a decision multi-tree can also be seen as an AND/OR tree [12, 14], if one considers the split nodes as being OR-nodes and considers the nodes generated by an exploited OR-node as being AND-nodes.

Formally, a decision multi-tree is formed by an AND-node on the root, with a set of children which are OR-nodes (each one represents the split considered). Each OR-node can be *active* or *suspended.* The active OR-nodes have a set of children which are AND-nodes corresponding to a descendant of the split. This schema is repeated for each new descendant node (AND-node), until an AND-node that is not further explored and it is assigned a class (a leaf).

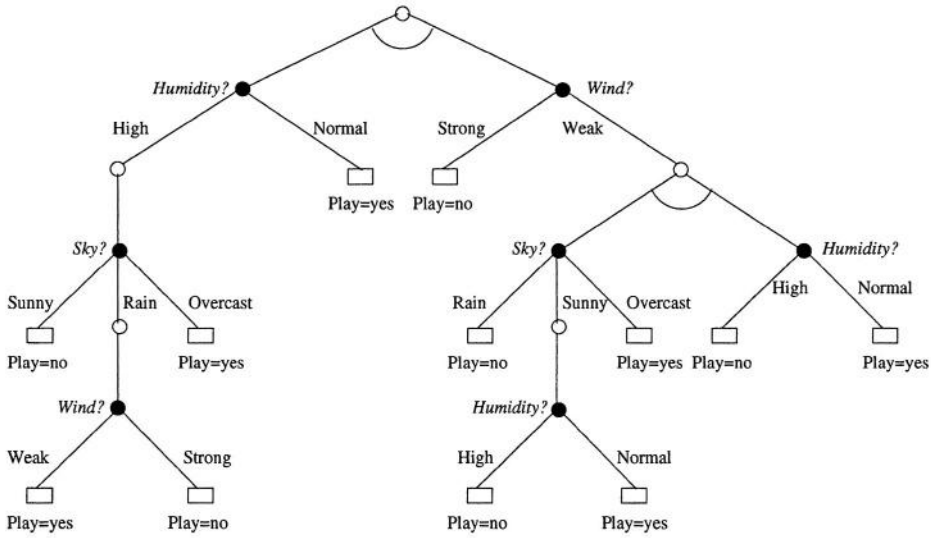


Fig. 2. The multi-tree structure.

Figure 2 shows a decision multi-tree that contains the four decision trees of Figure 1. AND-nodes are represented with no filled circles and they have an arc under the node. Leaves are represented by rectangles. OR-nodes are expressed by black-filled circles. In a decision multi-tree, as we can see, there are alternatively levels of AND-nodes and OR-nodes. Note that a classic decision tree can be seen as a decision multi-tree where only one OR-node is explored at each OR-node level.

In previous work [7], we have introduced an ensemble method that employs the decision multi-tree structure. In that work, the ensemble method performs a beam-search population of the decision multi-tree based on a random selection of the suspended nodes to be explored. However, one of the critical issues about this technique is the criterion for choosing the OR-node to "wake" from its suspended state.

3 Bagging Construction of a Decision Multi-tree

In this section we present our method to construct a decision multi-tree by using different training sets. As in *Bagging,* each training set is a bootstrap replicate

of the original training set. However, as we have mentioned in the introduction, our approach differs from *Bagging* in that we construct a single structure (a multi-tree) that includes the ensemble of decision trees obtained by *Bagging* but without repeating their common parts.

In order to implement this, we use the first bootstrap replicate for filling the multi-tree with a single decision tree. Then, for each new bootstrap replicate we continue the construction of the multi-tree, but only exploiting those nodes which have not been considered in the previous bootstrap replicates (i.e. they are suspended OR-nodes).

3.1 Algorithm

The following algorithm formalises this process:

Algorithm Bagging-Multi-tree (INPUT E**:dataset,** n**:integer; OUTPUT** M**:multi-tree)** {n is the number of iterations}

```
    M=Initialize_multi_tree(); {M only contains an empty AND-node}
    for i=1 to n do
        D=Bootstrap_replicate(E); {a bootstrap replicate is generated}
        if i=1 then LearnM(M.root, D)
        else LearnMBagg(M.root, D)
    end for
end
```

As we can see, the algorithm begins by generating an empty multi-tree, that is, a multi-tree that only contains one AND-node. Then, a bootstrap replicate is obtained and processed in each step of the main loop. At the first iteration, a multi-tree is constructed (procedure *LearnM*) and, for the following iterations, this structure is populated using a new bootstrap replicate for selecting the suspended OR-node that must be exploited (procedure *LearnMBagg*). In what follows we describe both processes.

Procedure *LearnM* generates a multi-tree by selecting in each OR-level the best OR-node to be exploited (using, e.g. GainRatio or other splitting criterion). The process is repeated for the descendant of the active OR-node until a leaf is reached.

Procedure *LearnM* **(INPUT** X**:AND-node,** D**:training dataset; OUTPUT** M**:multi-tree)**

```
    if X=leaf then exit;
    List_of_OR_nodes=Create_OR_nodes(X, D); {generate a list with one OR-
    node for each possible split and their descendants (AND-nodes)}
    B=Select_Best_OR_node(L, D); {the best node according to the split opti-
    mality criterion is selected}
    Activate(B); {the selected OR-node is activated}
    for Y ∈ children_of(B) do
        D'=filter(D, Y); {the examples of D that fall in node Y are selected}
        LearnM(Y, D')
    end for
end procedure
```

As in the above process, Procedure *LearnMBagg* also selects in each OR-level the best OR-node to be exploited. But in this case the selected OR-node can be active or suspended. If it is an active node (which means that it has been exploited previously) then the procedure continues exploring their children. However, if it is an OR-node suspended, it is activated and their children are added to the multi-tree.

Procedure *LearnMBagg* **(INPUT** X**:AND-node,** D**:training dataset; OUTPUT** M**:multi-tree)**

```
if X=leaf then exit;
List_of_OR_nodes=Update_OR_nodes(X, D);{update the split optimality of
OR-nodes according to the training set D}
B=Select_Best_OR_node(L, D);
if B is active then
    for Y ∈ children_of(B) do
        D'=filter(D, Y);
        LearnMBagg(Y, D')
    end for
else
    Activate(B);
    for Y ∈ children_of(B) do
        D'=filter(D, Y);
        LearnM(Y, D') {the multi-tree is expanded from node Y}
    end for
end procedure
```

As regards the combination of predictions, there is an important difference with respect to classical ensemble methods: fusion points are distributed allover the multi-tree structure. Concretely, we combine the votes at the AND-nodes using the maximum fusion strategy. This strategy obtains the best results according to the experiments of [6].

3.2 Bagging Decision Trees versus Bagging Decision Multi-trees

Although the algorithm presented in this section is inspired by the *Bagging* method over decision trees, there are some differences between these two techniques and there can be differences in the errors performed by both methods. The most significant differences are how the ensemble is used for predicting new cases.

In *Bagging* decision trees, there is a significant probability (as we will see in the experiments) of learning similar trees. In the prediction phase, the repeated decision trees will be more determinant in the final decision. In the decision multi-tree, since we avoid duplicated trees, all the leaves have identical weight in the final decision. Note that this mechanism of ignoring repeated leaves for the prediction can be intuitively justified by the fact of having a set of models semantically different can help to improve the accuracy of the predictions.

Additionally, in a decision multi-tree the fusion of the predictions is performed internally at the OR-nodes, while in an ensemble of decision trees the

voting is performed using each independent decision tree. Performing the fusion in the internal nodes of the multi-tree can alter the colour of the final decision. Furthermore, it represents an important improvement in the response time of the ensemble.

4 Experiments

In this section, we present an experimental evaluation of our approach, as is implemented in the SMILES system [5]. SMILES is a multi-purpose machine learning system which (among many other features) includes the implementation of a multiple decision tree learner.

Table 1. Datasets used in the experiments.

#	Datasets	Size	Classes	Nom. Attr.	Num. Attr.
1	Balance Scale	325	3	0	4
2	Breast Cancer	699	2	0	9
3	Breast Cancer Wisconsin	569	2	1	30
4	Chess	3196	2	36	0
5	Dermatology	366	6	33	1
6	Hayes-Roth	106	3	5	0
7	Heart Disease	920	5	8	5
8	Hepatitis	155	2	14	5
9	Horse-colic-outcome	366	3	14	8
10	Horse-colic-surgical	366	2	14	8
11	House Congressional Voting	435	2	16	0
12	Iris Plan	158	3	0	4
13	MONK's1	566	2	6	0
14	MONK's2	601	2	6	0
15	MONK's3	554	2	6	0
16	New Thyroid	215	3	0	5
17	Postoperative Patient	90	3	7	1
18	Segmentation Image Database	2310	7	0	14
19	Teaching Assistant Evaluation	151	3	2	3
20	Thyroid ANN	7200	3	15	0
21	Tic-Tac-Toe Endgame	958	2	8	0
22	Wine Recognition	178	3	0	13

For the experimental evaluation, we have employed 22 datasets from the UCI dataset repository [1]. Some details of the datasets are included in Table 1.

For the experiments, we used GainRatio [15] as a splitting criterion. Pruning is not enabled. The experiments were performed on a Pentium III-800 Mhz with 180MB of memory running Linux 2.4.2.

First, let us examine how the multi-tree grows with respect to the number of iterations of *Bagging*. Table 2 shows the mean of the number of nodes (AND nodes and OR nodes) in the multi-tree of all the datasets. The results are rather surprising since the number of nodes does not increase as one would expect. This reflects the facts that *Bagging* tends to repeat frequently the same decision trees. We also must remark that we do not remove the suspended OR nodes. This is the reason for the relatively high number of nodes of the multi-tree with just one iteration.

Table 2. Mean size of multi-trees in number of nodes (OR-nodes and And-nodes).

Number of iterations	1	20	40	60
Mean size	4021	9287	12927	15493

Table 3. Mean accuracy.

	1		20		40		60	
#	BagMDT	BagDT	BagMDT	BagDT	BagMDT	BagDT	BagMDT	BagDT
1	77,94	79,40	79,26	81,92	80,13	82,85	80,63	82,92
2	93,81	94,12	94,28	96,07	94,64	96,11	94,81	96,28
3	92,43	94,22	93,48	95,61	93,13	95,91	93,29	95,91
4	99,61	99,40	99,37	99,43	99,42	99,40	99,28	99,40
5	91,58	93,80	94,00	96,43	94,31	96,89	94,75	97,05
6	72,13	72,50	73,44	47,31	73,44	54,25	73,19	52,56
7	51,64	47,26	54,58	46,21	54,72	46,45	55,98	46,52
8	76,20	78,98	77,27	83,23	77,47	82,38	77,80	82,76
9	62,72	65,82	64,14	65,66	64,81	65,77	65,33	65,79
10	78,53	83,15	81,92	82,86	82,19	83,12	82,50	83,06
11	94,74	95,47	94,95	96,45	94,77	96,55	94,81	96,58
12	94,13	94,60	95,13	94,57	94,13	94,33	94,47	94,20
13	96,73	95,11	95,93	99,84	96,42	100,00	95,73	100,00
14	70,97	62,66	67,62	66,38	67,45	66,91	66,17	67,19
15	97,47	98,67	98,00	98,74	97,69	98,90	97,93	98,88
16	92,81	92,95	92,76	94,24	93,29	94,56	93,10	94,76
17	66,00	59,89	66,25	63,23	65,88	64,44	65,50	65,00
18	95,91	96,90	95,29	97,67	95,23	97,69	95,23	97,62
19	61,93	56,46	56,93	59,35	57,33	61,50	55,60	60,51
20	99,23	99,72	99,15	99,65	99,06	99,65	98,98	99,65
21	77,16	79,16	79,54	83,17	80,32	83,82	80,34	83,80
22	93,00	93,49	93,12	95,26	93,06	95,79	92,94	95,90
Geomean	82.18	81.66	82.60	81.67	82.73	82.55	82.69	82.46
Average	83.49	83.35	83.93	83.79	84.04	84.42	84.02	84.38

Table 3 shows the accuracy comparison (10 × 10-fold cross-validation) between classical *Bagging* as it is implemented in Weka (we call it *BagDT*), and the proposed algorithm as it is implemented in Smiles (we call it *BagMDT*) depending on the number of iterations of the ensemble method. We have employed the version 3.2.3 of Weka[1]. We use *J* 48 as base classifier (the Weka version of C4.5), with the default settings, except from pruning, which is not enabled.

The results of both algorithms are similar. Initially, there is a slight advantage for the *BagMDT* due to some small differences between the two systems in the implementation of the C4.5 algorithm. When the number of iterations is increased, both learning methods improve the accuracy, and the difference between them is partially reduced (even *BagDT* obtain better results if we consider the arithmetic average). Nonetheless, it seems that *BagMDT* reaches the saturation point earlier, probably because the repeated "good" branches are not weighted more for higher number of iterations.

Table 4 contains the average learning time for each classifier and dataset, and the geometric and arithmetic mean of all the datasets. From a practical point of view, it is resource consumption where we see the advantages of using decision multi-trees when learning ensembles of decision trees, since the training

[1] http://www.cs.waikato.ac.nz/~ml/weka/

Table 4. Mean training time.

	1		20		40		60	
#	BagMDT	BagDT	BagMDT	BagDT	BagMDT	BagDT	BagMDT	BagDT
1	0,04	0,08	0,31	1,07	0,59	2,29	0,85	3,24
2	0,08	0,06	0,67	1,07	1,29	2,32	1,87	3,19
3	0,32	0,29	2,50	4,24	4,81	8,73	6,94	13,02
4	0,26	0,53	2,28	6,39	4,43	12,44	6,51	18,72
5	0,08	0,04	0,60	0,65	1,11	1,28	1,60	1,93
6	0,01	0,01	0,01	0,18	0,03	0,35	0,04	0,57
7	0,27	0,18	2,53	5,41	5,96	10,43	21,95	15,88
8	0,04	0,03	0,26	0,36	0,48	0,75	0,72	1,13
90	0,15	0,03	1,01	0,51	2,30	1,10	2,82	1,36
10	0,12	0,04	0,87	1,16	1,61	2,06	2,36	3,13
11	0,02	0,03	0,15	0,39	0,39	0,78	0,43	1,18
12	0,01	0,01	0,04	0,09	0,08	0,18	0,12	0,27
13	0,01	0,02	0,06	0,25	0,16	0,47	0,17	0,74
14	0,01	0,04	0,13	0,50	0,33	0,91	0,35	1,50
15	0,01	0,01	0,04	0,15	0,12	0,29	0,13	0,44
16	0,02	0,02	0,15	0,24	0,28	0,49	0,40	0,75
17	0,01	0,01	0,03	0,10	0,07	0,19	0,08	0,31
18	0,85	1,17	8,93	16,90	19,41	33,56	40,00	51,31
19	0,02	0,02	0,15	0,28	0,28	0,55	0,40	0,85
20	1,13	0,69	9,84	9,53	20,59	18,95	31,99	28,54
21	0,03	0,06	0,28	0,74	0,72	1,43	0,78	2,35
22	0,04	0,03	0,27	0,44	0,52	0,88	0,74	1,33
Geomean	0.04	0.05	0.34	0.73	0.72	1.44	1.03	2.19
Average	0.16	0.15	1.41	2.30	2.98	4.57	5.51	6.90

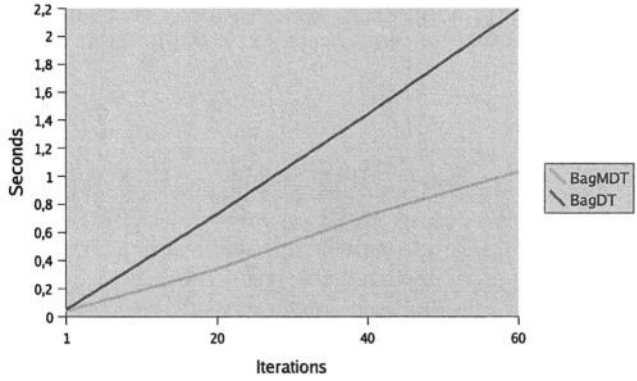


Fig. 3. Training time comparison.

time is significantly reduced. To appreciate better this feature, Figure 3 shows the geometric average training time of *Bagging,* using Weka, and Smiles depending on the size of the ensemble. While *Bagging* decision trees shows a linear increase in time, *Bagging* decision multi-trees shows also a linear increase with a significant lower slope.

5 Conclusions

In this work, we present an algorithm that reduce the high computational cost characteristic of *Bagging* method. The technique is based on the use of the multi-

tree structure. This structure allows trees to share the common parts. Therefore, the more structurally similar the trees are the better the improvement of the computational resources made by the multi-tree. Additionally, the multi-tree also makes it possible to enhance a key parameter in ensemble techniques: diversity. Note that by applying *Bagging* over c.4.5, the set of decision trees obtained presents many structural similarities (low diversity), so that misclassification errors can be easily correlated. But if these decision trees are organised into a multi-tree the redundancy is reduced and theoretically the accuracy should be better. In fact, *Bagging* multi-tree reaches the saturation point earlier than classical *Bagging*. However, a collateral effect arises when multi-tree is used, because this structure does not take into account how many times a branch tree appears. Therefore, the frequent branches and unusual ones have the same weight in the classification stage. The experimental evaluation makes clear that it would be feasible to get a trade-off between the redundancy and the diversity, by using the multi-tree structure.

Summing up, the multi-tree structure can be viewed as a feasible and elegant way to overcome the main inherent drawbacks (huge amount of the computational resources and redundancy) of *Bagging*.

As future work, it would be interesting to investigate how we can improve accuracy by an adequate adjustment of the diversity and redundancy parameters in *Bagging* multi-trees. Additionally, we also plan to study whether the multi-tree is able to enhance other well-known ensemble methods, such as boosting.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. L. Breiman. Bagging predictors. *Machine Learning,* 24(2): 123–140, 1996.
3. T. G Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems,* pages 1–15, 2000.
4. T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning,* 40(2): 139–157, 2000.
5. V. Estruch, C. Ferri, J. Hernández, and M. J. Ramírez. SMILES: A multi-purpose learning system. In *Logics in Artificial Intelligence, European Conference, JELIA,* volume 2424 of *Lecture Notes in Computer Science,* pages 529–532, 2002.
6. V. Estruch, C. Ferri, J. Hernández, and M.J. Ramírez. Shared Ensembles using Multi-trees. In *the 8th Iberoamerican Conference on Artificial. Intelligence, Iberamia'02,* volume 2527 of *LNCS,* pages 204–213, 2002.
7. V. Estruch, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez. Beam Search Extraction and Forgetting Strategies on Shared Ensembles. In *4th Workshop on Multiple Classifier Systems, MCS2003,* LNCS, 2003.
8. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning,* pages 148–146. Morgan Kaufmann, 1996.
9. Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-12(10):993–1001, October 1990.

10. Ron Kohavi and Clayton Kunz. Option decision trees with majority votes. In *Proc. 14th International Conference on Machine Learning,* pages 161–169. Morgan Kaufmann, 1997.
11. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
12. N.J. Nilsson. *Artificial Intelligence: a new synthesis.* Morgan Kaufmann, 1998.
13. Bambang Parmanto, Paul W. Munro, and Howard R. Doyle. Improving committee diagnosis with resampling techniques. In *Advances in Neural Information Processing Systems,* volume 8, pages 882–888. The MIT Press, 1996.
14. J. Pearl. *Heuristics: Intelligence search strategies for computer problem solving.* Addison Wesley, 1985.
15. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.
16. J. R. Quinlan. Bagging, Boosting, and C4.5. In *Proc. of the 30th Nat. Conf. on A.I. and the 8th Innovative Applications of A.I. Conf.,* pages 725–730. AAAI Press / MIT Press, 1996.

Learn++.MT: A New Approach to Incremental Learning

Michael Muhlbaier, Apostolos Topalis, and Robi Polikar

Rowan University, Electrical and Computer Engineering Department
201 Mullica Hill Rd., Glassboro, NJ 08028, USA
{muhl1565,topa4536}@students.rowan.edu, polikar@rowan.edu

Abstract. An ensemble of classifiers based algorithm, Learn++, was recently introduced that is capable of incrementally learning new information from datasets that consecutively become available, even if the new data introduce additional classes that were not formerly seen. The algorithm does not require access to previously used datasets, yet it is capable of largely retaining the previously acquired knowledge. However, Learn++ suffers from the inherent "out-voting" problem when asked to learn new classes, which causes it to generate an unnecessarily large number of classifiers. This paper proposes a modified version of this algorithm, called Learn++.MT that not only reduces the number of classifiers generated, but also provides performance improvements. The out-voting problem, the new algorithm and its promising results on two benchmark datasets as well as on one real world application are presented.

1 Introduction

It is well known that the amount of training data available and how well the data represent the underlying distribution are of paramount importance for an automated classifier's satisfactory performance. For many applications of practical interest, obtaining such adequate and representative data is often expensive, tedious, and time consuming. Consequently, it is not uncommon for the entire data to be obtained in installments, over a period of time. Such scenarios require a classifier to be trained and incrementally updated – as new data become available – where the classifier needs to learn the novel information provided by the new data without forgetting the knowledge previously acquired from the data seen earlier. This raises the so-called stability-plasticity dilemma [1]: a completely stable classifier can retain knowledge, but cannot learn new information, whereas a completely plastic classifier can instantly learn new information, but cannot retain previous knowledge. Many popular classifiers, such as the ubiquitous multilayer perceptron (MLP) or the radial basis function networks, are not structurally suitable for incremental learning, since they are "completely stable" classifiers. The approach generally followed for learning from new data involves discarding the existing classifier, combining the old and the new data and training a new classifier from scratch using the aggregate data. This causes the previously learned information to be lost, a phenomenon known as catastrophic forgetting [2]. Furthermore, training with the combined data may not even be feasible, if the previously used data are lost, corrupted, prohibitively large, or otherwise unavailable.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 52–61, 2004.

We have recently introduced an algorithm, called Learn++, capable of learning incrementally, even under hostile learning conditions: not only does Learn++ assume the previous data to be no longer available, but it also allows additional classes to be introduced with new data, while retaining the previously acquired knowledge.

Learn++ is an ensemble approach, inspired primarily by the AdaBoost algorithm. Similar to AdaBoost, Learn++ also creates an ensemble of (weak) classifiers, each trained on a subset of the current training dataset, and later combined through weighted majority voting. Training instances for each classifier are drawn from an iteratively updated distribution. The main difference is that the distribution update rule in AdaBoost is based on the performance of the previous hypothesis [3], which focuses the algorithm on difficult instances, whereas that of Learn++ is based on the performance of the entire ensemble [4], which focuses this algorithm on instances that carry novel information. This distinction gives Learn++ the ability to learn new data, even when previously unseen classes are introduced. As new data arrive, Learn++ generates additional classifiers, until the ensemble learns the novel information. Since no classifier is discarded, previously acquired knowledge is retained. Other approaches suggested for incremental learning, a bibliography of ensemble systems and their applications can be found in and within the references of [4 ~9].

As reported in [4,5], Learn++ works rather well on a variety of real world problems, though there is much room for improvement. An issue of concern is the relatively large number of classifiers required for learning instances coming from a new class. This is because, when a new dataset introduces a previously unseen class, new classifiers are trained to learn the new class; however, the existing classifiers continue to misclassify instances from the new class. Therefore, the decisions of latter classifiers that recognize the new class are out-voted by the previous classifiers that do not recognize the new class, until a sufficient number of new classifiers are generated that recognize the new class. This leads to classifier proliferation.

In this contribution, we first describe the out-voting problem associated with the original Learn++, propose a modified version of the algorithm to address this issue, and present some preliminary simulation results on three benchmark datasets.

2 Learn++.MT

In ensemble approaches that use a voting mechanism for combining classifier outputs, each classifier votes on the class it predicts [10, 11]. The final classification is then determined as the class that receives the highest total vote from all classifiers. Learn++ uses weighted majority voting [12], where each classifier receives a voting weight based on its training performance. This works well in practice for most applications. However, for incremental learning problems that involve introduction of new classes, the voting scheme proves to be unfair towards the newly introduced class: since none of the previously generated classifiers can pick the new class, a relatively large number of new classifiers that recognize the new class are needed, so that their total weight can out-vote the first batch of classifiers on instances of the new class. This in return populates the ensemble with an unnecessarily large number of classifiers. Learn++.MT is specifically designed to address the classifier proliferation issue. The novelty in Learn++.MT is the way by which the voting weights are determined. Learn++.MT also uses a set of voting weights based on the classifiers' performances,

however, these weights are then adjusted based on the classification of the specific instance at the time of testing, through dynamic weight voting (DWV).

For any given test instance, Learn++.MT compares the class predictions of each classifier and cross-references them against the classes on which they were trained. If a subsequent ensemble overwhelmingly chooses a class it has seen before, then the voting weights of those classifiers not trained with that class are proportionally reduced. As an example, assume that an ensemble has seen classes 1 and 2, and a second ensemble has seen classes 1, 2 and 3. For a given instance, if the second ensemble (trained on class 3) picks class 3, the classifiers in the first ensemble (which has not seen class 3) reduce their voting weights in proportion to the confidence of the second ensemble. In other words, when the algorithm detects that the new classifiers overwhelmingly choose a new class on which they were trained, the weights of the other classifiers which have not seen this new class are reduced. The Learn++.MT algorithm is given in Figures 1 and 2, and explained in detail below.

For each dataset ($\mathcal{D}_k$) that becomes available to Learn++.MT, the inputs to the algorithm are (i) a sequence of m_k training data instances x_i and their correct labels y_i, (ii) a classification algorithm **BaseClassifier,** and (iii) an integer T_k specifying the maximum number of classifiers to be generated using that database. If the algorithm is seeing its first database (k=1), a data distribution (D_t) – from which training instances will be drawn – is initialized to be uniform, making the probability of any instance being selected equal. If k>l then the distribution is updated from the previous step based on the performance of the existing ensemble on the new data. The algorithm then adds T_k classifiers to the ensemble starting at $t=eT_k+1$ where eT_k denotes the number of classifiers that currently exist in the ensemble.

For each iteration t, the instance weights, w_t, from the previous iteration are first normalized (step 1) to create a weight distribution D_t. A hypothesis, h_t, is generated from a subset of $\mathcal{D}_k$ that is drawn from D_t (step 2). The error, ε_t, of h_t is then calculated; if $\varepsilon_t > ½$, the algorithm deems the current classifier, h_t, to be too weak, discards it, and returns to step 2, otherwise, calculates the normalized error β_t (step 3). The class labels of the training instances used to generate this hypothesis are then stored as CTr_t (step 4). The dynamic weight voting (DWV) algorithm is called to obtain the composite hypothesis, H_t, of the ensemble (step 5). H_t represents the ensemble decision of the first t hypotheses generated thus far. The error of the composite hypothesis, E_t is then computed and normalized (step 6). The instance weights w_t are finally updated according to the performance of H_t (step7) such that the weights of instances correctly classified by H_t are reduced (and those that are misclassified are effectively increased). This ensures that the ensemble focus on those regions of the feature space that are not yet learned, paving the way for incremental learning.

The inputs to the dynamic weight voting algorithm are (i) the current training data (during training) or any test instance, (ii) classifiers h_t, (iii) β_t, normalized error for each h_t, and (iv) the vector CTr_t containing the classes on which h_t has been trained. Classifier weights are first initialized (step 1), where each classifier receives a standard weight that is inversely proportional to its normalized error β_t so that those classifiers that performed well on their training data are given higher voting weights. A normalization factor Z_c is then created as the sum of the weights of all classifiers trained on instances from class c (step 2).

Input: For each dataset $\mathcal{D}_k$ $k=1,2,...,K$

- Sequence of i=1,…,m_k instances x_i with labels $y_i \in Y_k = \{1,...,c\}$
- Weak learning algorithm **BaseClassifier**.
- Integer T_k, specifying the number of iterations.

Do for $k=1,2,...,K$

If k=1 **Initialize** $w_1 = D_1(i) = 1/m$, $eT_1 = 0$ for all i.

Else Go to Step 5 to evaluate the current ensemble on new dataset $\mathcal{D}_k$, update weights, and recall current number of classifiers → $eT_k = \sum_{j=1}^{k-1} T_j$

Do for $t=eT_k+1$, eT_k+2,..., eT_k+T_k:

1. Set $D_t = w_t \Big/ \sum_{i=1}^{m} w_t(i)$ so that D_t is a distribution.
2. **Call BaseClassifier** with a subset of $\mathcal{D}_k$ randomly chosen using D_t.
3. Obtain h_t : X → Y, and calculate its error: $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

   If $\varepsilon_t > ½$, discard h_t and go to step 2. Otherwise, compute normalized error as $\beta_t = \varepsilon_t/(1-\varepsilon_t)$.
4. $CTr_t = Y_k$, to save labels of classes used in training h_t.
5. **Call DWV** to obtain the composite hypothesis H_t.
6. Compute the error of the composite hypothesis $E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i)$
7. Set $B_t=E_t/(1-E_t)$, and update the instance weights:

$$w_{t+1}(i) = w_t \times \begin{cases} B_t, & \text{if } H_t(x_i) = y_i \\ 1, & \text{otherwise} \end{cases}$$

Call DWV to obtain the final hypothesis, H_{final}.

Fig. 1. Learn++.MT Algorithm.

For each instance, a preliminary per-class confidence factor $0<P_c<1$ is generated (step 3). P_c is the sum of weights of all the classifiers that choose class c divided by the sum of the weights of all classifiers trained with class c (which is Z_c). In effect, this can be considered as the ensemble assigned confidence of the instance for belonging to each of the c classes. Then, again for each class, the weights are adjusted for classifiers that have not been trained with that class, that is, the weights are lowered proportional to the ensemble's preliminary confidence on that class (step 4). The final / composite hypothesis is then calculated as the maximum sum of the weights that chose a particular class (step 5).

Input:

- Sequence of i=1,…, n training instances or any test instance x_i
- Classifiers h_t.
- Corresponding error values, β_t.
- Classes, CTr_t used in training h_t.

For t=1,2,…,T where T is the total number classifiers

1. **Initialize** classifier weights $W_t = \log(1/\beta_t)$
2. **Create** normalization factor, Z, for each class
$$Z_c = \sum_{t:c\in CTr_t} W_t \text{, for } c=1,2,...,C \text{ classes}$$
3. **Obtain** preliminary decision $P_c = \sum_{t:h_t(x_i)=c} W_t \Big/ Z_c$, for $c=1,2,...,C$
4. **Update** voting weights $W_{t:c\notin CTr_t} = W_{t:c\notin CTr_t}(1-P_c)$, for $c=1,2,...,C$
5. **Compute** final / composite hypothesis
$$H_{final}(x_i) = \arg\max_c \sum_{t:h_t(x_i)=c} W_t$$

Fig. 2. Dynamic Weight Voting Algorithm for Learn++.MT.

3 Learn++.MT Simulation Results

Learn++.MT has been tested on several databases. For brevity, we present results on two benchmark databases and one real-world application. The benchmark databases are the Wine database and the Optical Character Recognition database from UCI [13], and the real world application is a gas identification problem for determining one of five volatile organic compounds based on chemical sensor data. MLPs – normally incapable of incremental learning – were used as base classifiers on all three cases. Base classifiers were all single layer MLPs with 20~50 nodes and a rather generous error goal of 0.1 ~ 0.01 to ensure weak classifiers with respect to the difficulty of the underlying problem.

3.1 Wine Recognition Database

The Wine Recognition database features 3 classes with 13 attributes. The database was split into two training, a validation, and a test dataset. The data distribution is given in Table 1. In order to test the algorithms' ability to incrementally learn a new class, instances from class 3 are only included in the second dataset. Each algorithm was allowed to create a set number of classifiers (30) on each dataset. The optimal number of classifiers to retain for each dataset was automatically determined based on the maximum performance on the validation data. This process was applied 30 times on Learn++ and Learn++.MT to compare their generalization performance on the test data, the mean results of which are shown in Tables 2 and 3. Each row shows class-

by-class generalization performance of the ensemble on the test data after being trained with dataset $\mathcal{D}_k$, k=1,2. The last two columns are the average overall generalization performance over 30 simulation trials (on the entire test data which includes instances from all three classes), and the standard deviation of the generalization performances. The number of classifiers in the ensemble after each training session is given in parentheses.

Table 1. Wine Recognition database distribution.

Class→	1	2	3
$\mathcal{D}_1$	26	31	0
$\mathcal{D}_2$	13	16	32
Valid.	7	8	5
Test	13	16	11

Table 2. Learn++ performance results on Wine Recognition database.

Class→	1	2	3	Gen.	Std.
$\mathcal{D}_1$ (6)	99%	96%	-	71%	5.8
$\mathcal{D}_2$ (26)	100%	94%	24%	**77%**	**12.1**

Table 3. Learn++.MT performance results on Wine recognition database.

Class→	1	2	3	Gen.	Std.
$\mathcal{D}_1$ (5)	96%	95%	-	70%	6.0
$\mathcal{D}_2$ (6)	99%	87%	90%	**92%**	**5.0**

Tables 2 and 3 show that Learn++.MT not only incrementally learns the new class, but also outperforms its predecessor by 15% using a significantly fewer number of classifiers. The poor performance of Learn++ in the new class (class 3) is explained below within the context of larger database simulations.

3.2 Optical Character Recognition Database

The optical character recognition (OCR) database features 10 classes (digits 0 ~ 9) with 64 attributes. The database was split into four to create three training and a test subset, whose distribution can be seen in Table 4. In this case, we wanted to evaluate the performance of each algorithm on a fixed number of classifiers (rather than determining the number of classifiers via a validation set) so that they can be compared on equal number of classifiers. Each algorithm was allowed to create five classifiers with the addition of each dataset (total of 15 classifiers in three training sessions). The data distribution was deliberately made rather challenging, specifically designed to test the algorithms' ability to learn multiple new classes at once with each additional dataset while retaining the knowledge of previously learned classes. In this incremental learning problem, instances from only six of the ten classes are present in each subsequent dataset resulting in a rather difficult problem. Results previously

obtained using Learn++ on this data using less challenging data distributions was in the order of lower to mid 90% range [4,5]. Results from this test are shown in Tables 5 and 6, which is formatted similar to the previous tables.

Table 4. OCR data distribution.

Class→	0	1	2	3	4	5	6	7	8	9
$\mathcal{D}_1$	250	250	250	0	0	250	250	250	0	0
$\mathcal{D}_2$	150	0	150	250	0	150	0	150	250	0
$\mathcal{D}_3$	0	150	0	150	400	0	150	0	150	400
Test	110	114	111	114	113	111	111	113	110	112

Table 5. Learn++ performance results on OCR database.

Class→	0	1	2	3	4	5	6	7	8	9	Gen.	Std.
$\mathcal{D}_1$	99%	98%	99%	-	-	96%	99%	100%	-	-	59%	0.6%
$\mathcal{D}_2$	98%	98%	99%	32%	-	96%	99%	100%	60%	-	68%	1.8%
$\mathcal{D}_3$	98%	96%	99%	94%	22%	96%	99%	100%	90%	13%	**81%**	**4.0%**

Table 6. Learn++.MT performance results on OCR database.

Class→	0	1	2	3	4	5	6	7	8	9	Gen.	Std.
$\mathcal{D}_1$	95%	98%	98%	-	-	95%	99%	100%	-	-	58%	0.8%
$\mathcal{D}_2$	96%	95%	99%	95%	-	95%	98%	100%	98%	-	69%	0.6%
$\mathcal{D}_3$	67%	95%	92%	98%	83%	63%	98%	100%	95%	96%	**89%**	**0.7%**

Interesting observations can be made from these tables. First, we note that Learn++ was able to learn the new classes, 3 and 8, only poorly after they were first introduced in $\mathcal{D}_2$ but able to learn them rather well, when further trained with these classes in $\mathcal{D}_3$. Similarly, it performs rather poorly on classes 4 and 9 after they are first introduced in $\mathcal{D}_3$, though it is reasonable to expect that it would do well on these classes with additional training. More importantly however, Learn++.MT was able to learn new classes quite well in its first attempt. Finally, recall that the generalization performance of the algorithm is computed on the entire test data which included instances from all classes. This is why the generalization performance is only around 60% after the first training session, since the algorithms have seen only six of the 10 classes in the test data. Both Learn++ and Learn++.MT exhibit an overall increase of generalization performance as new datasets are introduced – and hence the ability of incremental learning. Learn++.MT, however, is able to learn not only faster, but better than Learn++, as demonstrated by the significant jump in generalization performance (81% to 89%).

3.3 Volatile Organic Compound Recognition Database

The Volatile Organic Compound (VOC) database is a real world dataset that consist of 5 classes (toluene, xylene, hectane, octane and ketone) with 6 attributes coming

from six (quartz crystal microbalance type) chemical gas sensors. The dataset was divided into three training and a test dataset. The distribution of the data is given in Table 7, where a new class was introduced with each dataset.

Table 7. Volatile Organic Compounds database.

Class→	1	2	3	4	5
$\mathcal{D}_1$	20	0	20	0	40
$\mathcal{D}_2$	10	25	10	0	10
$\mathcal{D}_3$	10	15	10	40	10
Test	24	24	24	40	52

Again both algorithms were incrementally trained with three subsequent training datasets. In this experiment, both algorithms were allowed to generate as many classifiers as necessary to obtain their maximum performance. Learn++ generated a total of 36 classifiers to achieve its best performance. Learn++.MT, however, not only generated only 16 classifiers, but it also provided significant improvement in generalization performance.

Table 8. Learn++ performance results on VOC database.

Class	1	2	3	4	5	Gen	Std.
$\mathcal{D}_1$ (6)	90%	-	91%	-	88%	55%	1.3%
$\mathcal{D}_2$ (12)	89%	61%	95%	-	87%	64%	4.3%
$\mathcal{D}_3$ (18)	82%	95%	94%	12%	83%	**69%**	**6.6%**

Table 9. Learn++.MT performance results on VOC database.

Class	1	2	3	4	5	Gen	Std.
$\mathcal{D}_1$ (6)	90%	-	89%	-	89%	54%	2.0%
$\mathcal{D}_2$ (4)	86%	85%	93%	-	75%	63%	3.5%
$\mathcal{D}_3$ (6)	81%	96%	91%	83%	67%	**81%**	**2.3%**

The results averaged over 20 trials are given in Tables 8 and 9. We note that both algorithms provide a performance characteristic that is similar to those obtained with the previous databases. Specifically, Tables 8 and 9 show a significant increase from Learn++ to Learn++.MT on the average generalization performance. Furthermore, Learn++.MT was able to accomplish its performance using 20 fewer classifier, and learning each new class faster than its predecessor.

4 Conclusions and Discussions

In this paper we presented Learn++.MT, a modified version of our previously introduced incremental learning algorithm, Learn++. The novelty of the new algorithm is its use of preliminary confidence factors in assigning voting weights, based on a

cross-reference of the classes that have been seen by each classifier during training. Specifically, if a majority of the classifiers that have seen a class votes on that class, the voting weights of those classifiers who have not seen that class are reduced in proportion to the preliminary confidence. This allows the algorithm to dynamically adjust the voting weights for each test instance. The approach overcomes the out-voting problem inherent in the original version of Learn++ and prevents proliferation of unnecessary classifiers. The new algorithm also provided substantial improvements on the generalization performance on all datasets we have tried so far. We note that these improvements are more significant in those cases where one or several new classes are introduced with subsequent datasets.

It is also worth noting that, Learn++.MT is more robust than its predecessor. One of the reasons why Learn++ is having difficulty in learning a new class when first presented is due to difficulty in choosing the strength of the base classifiers. If we choose too weak classifiers, the algorithm is unable to learn. If we choose too strong classifiers, the training data are learned very well, resulting in very low β values which then causes very high voting weights, and hence even a more difficult out-voting problem. This explains why Learn++ requires larger number of classifiers or repeated training to learn the new classes. Learn++.MT, by significantly reducing the effect of the out-voting problem, improves the robustness of the algorithm, as the new algorithm is substantially more resistant to more drastic variations in the classifier architecture and parameters (error goal, number of hidden layer nodes, etc.).

We should also note however, while we have used MLPs as base classifiers, both algorithms are in fact independent of the type of the base classifier used, and can learn incrementally with any supervised classifier that lacks this ability. In fact, the classifier independence of Learn++ was demonstrated and reported in [5].

Further optimization of the distribution update rule, the selection of voting weights, as well as validation of the techniques on a broader spectrum of applications are currently underway.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. ECS-0239090, "CAREER: An Ensemble of Classifiers Approach for Incremental Learning."

References

1. S. Grossberg, "Nonlinear neural networks: principles, mechanisms and architectures," Neural Networks, vol. 1, no. 1, pp. 17-61, 1988.
2. R. French, "Catastrophic forgetting in connectionist networks," Trends in Cognitive Sciences, vol. 3, no.4, pp. 128-135,1999.
3. Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," Computer and System Sci., vol. 57, no. 1, pp. 119-139,1997.
4. R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," IEEE Trans. on Sys., Man and Cyber. (C), vol. 31, no. 4, pp. 497-508, 2001.

5. R. Polikar, J. Byorick, et al., "Learn++: a classifier independent incremental learning algorithm for supervised Neural Networks," Proc. of Int. Joint Conference on Neural Networks (IJCNN 2002), vol.2, pp. 1742-1747, Honolulu, HI, 2002.
6. M. Lewitt and R. Polikar, "An ensemble approach for data fusion with Learn++," 4th Int. Work. on Mult. Classifier Sys. LNCS (T. Windeatt and F. Roli, eds), vol. 2709, pp. 176-185, Springer: New York, NY, 2002.
7. J. Ghosh, "Multiclassifier systems: back to the future," 3rd Int. Work. on Mult. Classifier Sys., LNCS (J. Kitler & F. Roli, eds), vol. 2364, p. 1-15, Springer: New York, NY, 2002.
8. L.I. Kuncheva, "Switching between selection and fusion in combining classifiers: an experiment," IEEE Trans. on Sys., Man and Cyber., vol. 32(B), no. 2, pp. 146-156, 2002.
9. T. Windeatt and F. Roli (eds), In Proc. 4th Int. Workshop on Multiple Classifier Systems (MCS2003), LNCS, vol. 2709, Springer: New York, NY, 2002.
10. J. Kittler, M. Hatef, R.P. Duin, J. Matas, "On combining classifiers," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no.3, pp. 226-239,1998.
11. L.I. Kuncheva, "A theoretical study on six classifier fusion strategies, " IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 281-286, 2002.
12. N. Littlestone and M. Warmuth, "Weighted majority algorithm," Information and Computation, vol. 108, pp. 212-261, 1994.
13. C.L. Blake and C.J. Merz, UCI Repository of Machine Learning Databases at Irvine, CA: Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

Beyond Boosting: Recursive ECOC Learning Machines

Elizabeth Tapia[1], José C. González[2], Alexander Hütermann[2], and Javier García[3]

[1] Department of Electronic Engineering, National University of Rosario, Argentina
etapia@eie.fceia.unr.edu.ar
[2] Department of Telematic Engineering - Technical University of Madrid, Spain
{jgonzalez,ahutermann}@gsi.dit.upm.es
[3] Department of Informatic Systems, Complutense University of Madrid, Spain
javiergv@sip.ucm.es

Abstract. We present a wide experimental work evaluating the behaviour of Recursive ECOC (RECOC) [1] learning machines based on Low Density Parity Check (LDPC) coding structures. We show that owing to the iterative decoding algorithms behind LDPC codes, RECOC multiclass learning is progressively achieved. This learning behaviour confirms the existence of new boosting dimension, the one provided by the coding space. We present a method for searching potential good RECOC codes from LDPC ones. Starting from a properly selected LDPC code, we assess the effect of boosting in both weak and strong binary learners. For nearly all domains, we find that boosting a strong learner like a Decision Tree is as effective as boosting a weak one like a Decision Stump. This surprising result substantiates the hypothesis that weakening strong classifiers by boosting has a decorrelation effect, which can be used to improve RECOC learning.

1 Introduction

Standard ECOC [2] codes are the result of exhaustive searches on sets of random codes. No particular combinatorial constraint is assumed for such sets. Both dense and sparse random ECOC codes have explored. To some extend, this brute force approach is justified considering that the design of optimal ECOC codes is NP hard [3]. ECOC codes of this type can be only defined by the enumeration of valid codewords. The enumeration yields to their matrix representation. It should be noted, however, that ECOC matrices do not convey a true coding structure. As a result, ECOC hypotheses' assembling resorts, tightly linked to decoding algorithms, are strongly limited. In fact, they are constrained to loss variants [4] of the trivial Minimum Hamming distance-decoding algorithm.

In previous work [1][5], we explored ECOC multiclass learning limitations arising from this customary approach and proposed an alternative strategy for overcoming them. Main observed limitations are the exhaustive search requirement, and the disagreement between experimental results and the predicted behaviour from coding theory. Let us consider experimental results shown in [4]. They do not provide a guide for the choice of an ECOC strategy. Ideally, we would like to bound the search space with simple but effective constraints so that learning results are at least coherent. In view of the empirical evidence that random ECOC matrices induce almost uncorrelated learners and that randomness is in the core of the families of recursive

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 62–71, 2004.

error correcting codes [6], we proposed the design of ECOC codes using output *coding transformations defined by recursive error correcting codes.* These codes are constructed from component subcodes with some imprinting of randomness. Despite of their random flavour, a proper coding structure is still preserved. This structure is present in each component subcode and is cleverly used by *bitwise, soft iterative decoding algorithms* [7]. In ECOC learning terms, it implies that overall learning is spread between binary learners' working at multiple *local* multiclass learning contexts, as many as component subcodes.

The remainder of this paper is organized as follows. In section 2, we briefly review RECOC learning models based on LDPC codes [8][9]. In section 3, we analyze the setting of parameters required by RECOC LDPC learning. In section 4, we present experimental results. Finally, in section 5, we present conclusions and further work.

2 Recursive ECOC Learning

The effectiveness of general error correcting codes strongly depends on the memoryless channel assumption i.e. errors must be independent. ECOC matrices arising from classic error correcting codes induce correlated binary learners, i.e. channels with memory, so that training error degrades [10][11]. As one might anticipate, the correlation effect might be diluted if we allow a divide and conquer approach, i.e. ECOC learning machines constructed from and a set of component ones. On each component machine only a subset of binary learners could work. Subsets could be constructed randomly but keeping their size small so that the probability of detrimental cooperation between correlated binary learners remains small. These ideas are in the core of RECOC LDPC learning machines. LDPC codes are linear codes belonging to the family of recursive error correcting codes. Binary random sparse parity check matrices define them. For sufficient large codewords block lengths n, they perform near the Shannon limit when decoded with the iterative Sum-Product (SP) [9] algorithm over a Gaussian channel.

Definition (LDPC codes [8]): **A** Low-Density Parity Check (LDPC) code is specified by a pseudorandom parity check matrix **H** containing mostly 0's and a small number of ones. A binary (n,j,v)[1] LDPC code has block length n and a parity-check matrix **H** with exact j ones per column and v ones in each row, assuming $j \geq 3$ and $v \geq j$. Thus, every code bit is checked by exactly j parity checks and every parity check involves v codeword bits. The typical minimum distance of these codes increases linearly with n for a fixed j and v.

Let $c(x) \in C : X \rightarrow Y, |Y| = M > 2$, be a target multiclass concept explained by a training sample S. Let us assume a binary linear code Θ with parameters (n,k,d), i.e. codewords of length n conveying k information bits with Minimum Hamming distance d between them. Thus, we say that the channel rate is $r = k/n$. By setting $k = \lceil log_2 M \rceil$ we can perform output encoding on S by means of a transformation

[1] Please note that the (n,j,v) LDPC codes notation characterizing a particular parity check matrix is different from the (n,k,d) notation characterizing general linear block codes.

$\Gamma: Y \to \Theta$, which uses any subset of M different codewords from Θ. As a result, an ECOC learning machine can be constructed. Such a learning machine is defined by binary learners L_i trained with binary samples S_i, $0 \le i \le n-1$. The set of S_i, $0 \le i \le n-1$, is obtained by output enconding S with Θ. The expectation is that the greater redundancy $m=n-k$ in Θ, the stronger protection against errors from binary learners L_i, $0 \le i \le n-1$. Let us assume binary learners' errors satisfy a symmetric, additive, discrete, memoryless channel. Therefore, we admit that the binary hypothesis h_i issued by learner L_i upon seeing the input vector $\boldsymbol{x} \in X$ can be modelled as follows:

$$h_i = c_i + e_i \qquad 0 \le i \le n-1 \tag{1}$$

Assuming modulo 2 arithmetic, we are expressing that the true binary hypothesis c_i is hidden by an additive residual (noise) hypothesis e_i. For the sake of clearness, we have omitted the hypotheses's dependence on the input vector $\boldsymbol{x} \in X$. Probability of binary learners' errors, $p_i = P(e_i = 1) = \prec 0.5,\ 0 \le i \le n-1$, characterizing the learning channel can be estimated from training errors. The assembling of RECOC hypotheses, i.e. multiclassification, assumes the computation of symbol (bitwise) Maximum A Posteriori (MAP) estimates c_i^* from the vector $\boldsymbol{h}$ of binary noisy hypotheses, the estimated vector of binary residual hypotheses $\boldsymbol{p}$ and the coding structure Θ:

$$c_i^* = \arg \max_{c_i \in \{0,1\}} P\left[c_i \mid \boldsymbol{h}, \boldsymbol{p}, \Theta\right] \quad i = 0,\ldots,\ n-1 \tag{2}$$

For LDPC codes the iterative SP decoding algorithm does this work. The process involves a fixed number of iterations **I**. The concluding non-binary hypothesis $h_f(\boldsymbol{x})$ is reconstructed by means of $\Gamma^{-1}(\boldsymbol{c}^*)$ being $\boldsymbol{c}^* = [c_i^*]$. RECOC LDPC learning admits a further parameter concerning multiclass learning improvement by binary transduction. Binary learners can be boosted versions of some fixed weak binary learner [12]. The aim is that binary boosting can induce a better learning channel. We are aware that constructive boosting effects are limited to the point where overfitting and thus dependence between binary learners starts. On the other hand, it is well known that boosting strong binary learners produce the counterpart effect: learners become weaker. Taking into account that suitable weakened strong learners might provide independence between errors and a roughly noise-free learning channel, we explored RECOC LDPC learning with boosted Decision Trees. Boosting binary learners before output LDPC coding is comparable to a concatenated coding scheme with an inner repetition code [13]. A concatenated coding scheme provides further protection against noise. Experimental results confirm this interpretation. In addition, considering that for a fixed boosting level multiclass learning is improved using iterative decoding, it follows that parameter **I** defines a new boosting dimension, the one provided by suitable selected coding spaces.

3 The Setting of Parameters

The purpose of this section is to explain the setting of parameters in RECOC LDPC algorithms. Let us start with the choice of the $\Gamma: Y \to \Theta$. There are many of such

mappings and we do not find significant improvements when varying them. Setting $k = \lceil log_2 M \rceil$ might lead to unexpected results when $M \prec 2^k$. The iterative decoding algorithm might supply a codeword disregarded by $\Gamma : Y \to \Theta$. In these cases, we find significant improvements by a further stage of Minimum Hamming Distance decoding on the wrongly estimated codeword. The choice of the channel rate $r = k/n$ is closely related to the behaviour of LDPC codes over the assumed channel. Higher channel rates impose stronger constraints on quality of the (learning) channel. Lower ones increases the probability of generating repeated columns in ECOC matrices, a fact that might explain the unexpected[2] learning degradation at lower channel rates i.e. with greater redundancy. In our experimental work, we set $r = 0.1$ in order to allow comparison with recent work in ECOC multiclassification.

Recoc LDPC Algorithm

Input

Sample S with $|S| = m_S$, $Y = \{1,...,M\}$, $k = \lceil log_2 M \rceil$

Θ -LDPC code with parameters (n, j, v) at rate $r = \frac{k}{n}$

$\Gamma : Y \to \Theta$, Binary Learner L

Iterations **I** for the iterative **SP** algorithm

Processing

Compute training samples S_i, $i = 0,...,n-1$ using Γ and S

Train L with samples S_i to obtain $\boldsymbol{L} = [L_i]$, $i = 0,...,n-1$

Compute $\boldsymbol{p}$ from $\boldsymbol{L}$ and S_i, $i = 0,...,n-1$

Output

$h_f(\boldsymbol{x}) = \Gamma^{-1}[SP(\boldsymbol{x}, \boldsymbol{L}, \boldsymbol{p}, I)]$

End Recoc LDPC

Picking a good LDPC code in the learning sense is the main matter regarding the implementation of RECOC LDPC learning. Let us consider $k = 5$, which covers domains with $M \in [17, 32]$. Choosing $r = 0.1$ yields to $n = 50$ and $m = 45$. LDPC parity check matrices $H_{45\times50}$ can be constructed randomly assuming $j = 3$ ones per column and minimum Hamming distance $h = 2$ between them. From $H_{45\times50}$, the generator matrix $G_{5\times50}$ must be obtained. This step requires a matrix inversion operation over a square submatrix $P_{45\times45}$ in $H_{45\times50}$. Random construction of LDPC codes does not guarantee such inverse exists. However, we can still modify $H_{45\times50}$ by inserting ones

---

[2] In 4, sparse codes working at rate $r = 0.0625$ sometimes behave worse than the dense codes working at rate $r = 0.1$. The greater redundancy the lower protection?

to allow the inversion. Using G_{5x50} yields to the desired ECOC matrix. We only require the encoding of source vectors $s = (s_0,...,s_{k-1})$ representing multiclass labels. A challenging coding-learning problem now arises: how can we characterize those LDPC codes leading to good ECOC matrices? We find that scoring ECOC matrices with their mean Hamming distance between columns and their standard deviation can be used to pick some of them. Specifically, we simulated a sufficient large set of LDPC codes and compute the average Column Hamming Distances (CHD) and standard deviation in the resulting ECOC matrices. Intuitively, we must look for LDPC codes generating ECOC matrices with high average CHD and small standard deviation. Results for the case $M = 26$ are shown in Fig. 1.

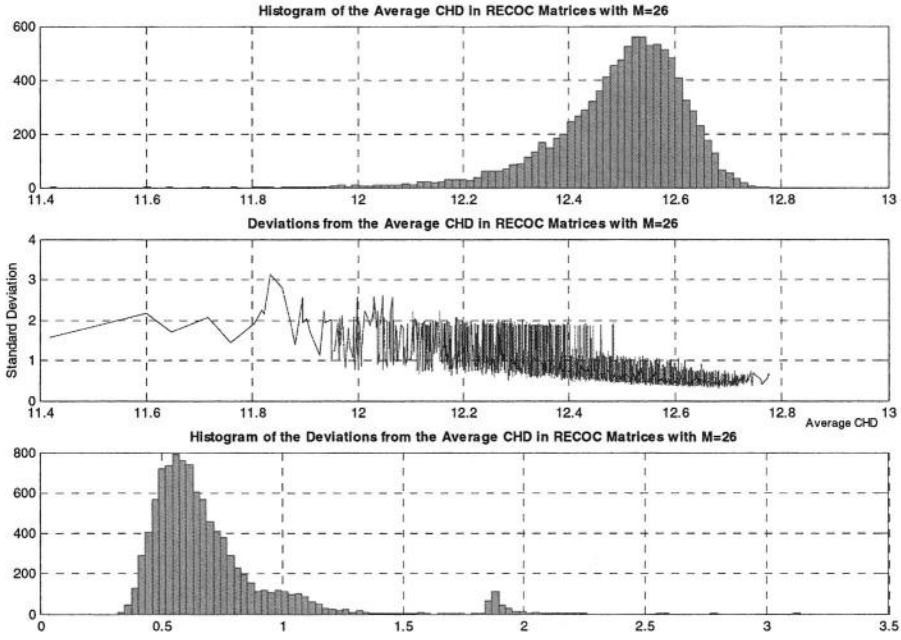


Fig. 1. Searching good RECOC codes from LDPC ones. Top and bottom figures show histograms of the average and standard deviation of CHD in RECOC matrices. Middle figure shows the relation between both.

Fig. 1 suggests that average CHD threshold values of 0.35 for maximum standard deviation and 12.65 for the minimum average might provide good results as indeed happened. An inspection to the whole set of ECOC matrices showed that those with too many repeated columns produced higher values of standard deviation. In fact, a limited number repeated ECOC columns are allowed in our framework as well as null ones. We expect that undesirable effects of correlated (equal!) binary learners are hopefully diluted in the divide and conquer approach. On the other hand, introducing a reduced amount of labelling noise solved the null column problem.

4 Experimental Results

Learning algorithms were developed using public domain Java WEKA library [14]. For LDPC coding, we developed a WEKA extension based on public domain D. J. MacKay software [9]. The experimental version is available upon request to the authors. We evaluated RECOC LDPC performance on 15 UCI data sets. For the sake of brevity we present a subset comprising only 9 data sets (see Table 1). Evaluation was done with the Test error when a partition was available. Otherwise, we used 10-Fold cross validation. Both strong (Decision Trees) and weak (Decision Stumps) were subject of study. We recall here that Decision Stumps are basically one level Decision Trees. Experimental results showed that about ten iterative boosting steps were enough for observing iterative learning improvements with Decision Trees. Convincing convergence results happened after a hundred of iterative decoding steps. We considered LDPC codes at $r = 0.1$, $j = 3$, $h = 2$. We first simulated 10000 LDPC codes with their corresponding ECOC matrices. These matrices were then ranked by their average CHDs. We then observed the top five matrices and picked by inspection the one exhibiting a good compromise between maximum average CHD and minimum variance. Regarding questions of stability in the SP algorithm, a threshold value of 1 .e-4 was assumed for all binary-training errors. Results with Decision Tree learners are depicted from Fig. 2 to Fig. 4, with individual runs manually clustered, and the boosting level shown in the upper right. Representative statistics are shown in Table 2. For each learning domain, we selected the run with the T value yielding to the minimum test or 10-fold CV error along the iterative decoding process. Alternatevely, we might fix a number of decoding steps and select T by crossvalidation. Parameter selection issues will be matter of future research. For a boosting level T, a channel rate r and number of classes M, the number of classifiers involved in a RECOC learning machine turns to be $r^{-1} \times \lceil \log_2 M \rceil \times T$. Representative results with Decision Stumps learners are included in Table 2.

Table 1. Learning Domains - UCI Data Sets.

	#Examples			
Domain	Train	Test	#Attributes	#Classes
Dermatology (DE)	366	-	34	6
Satimage (SA)	4435	2000	36	6
Glass (GL)	214	-	9	7
Pendigits (PE)	7494	3498	16	10
Vowel (VO)	528	462	10	11
Soybean (SO)	307	376	35	19
Primary Tumor (PT)	339	-	17	21
Audiology (AU)	226	-	69	24
Letter (LE)	16000	4000	16	26

Results shown in Fig. 2 to Fig. 4 and Table 2 confirm our claim of learning-coding consistency. RECOC codes are not constrained by naive dense or sparse combinatorial constraints or by artefacts like the use of pseudo ternary codes. Actually they are supported in a powerful coding structure.

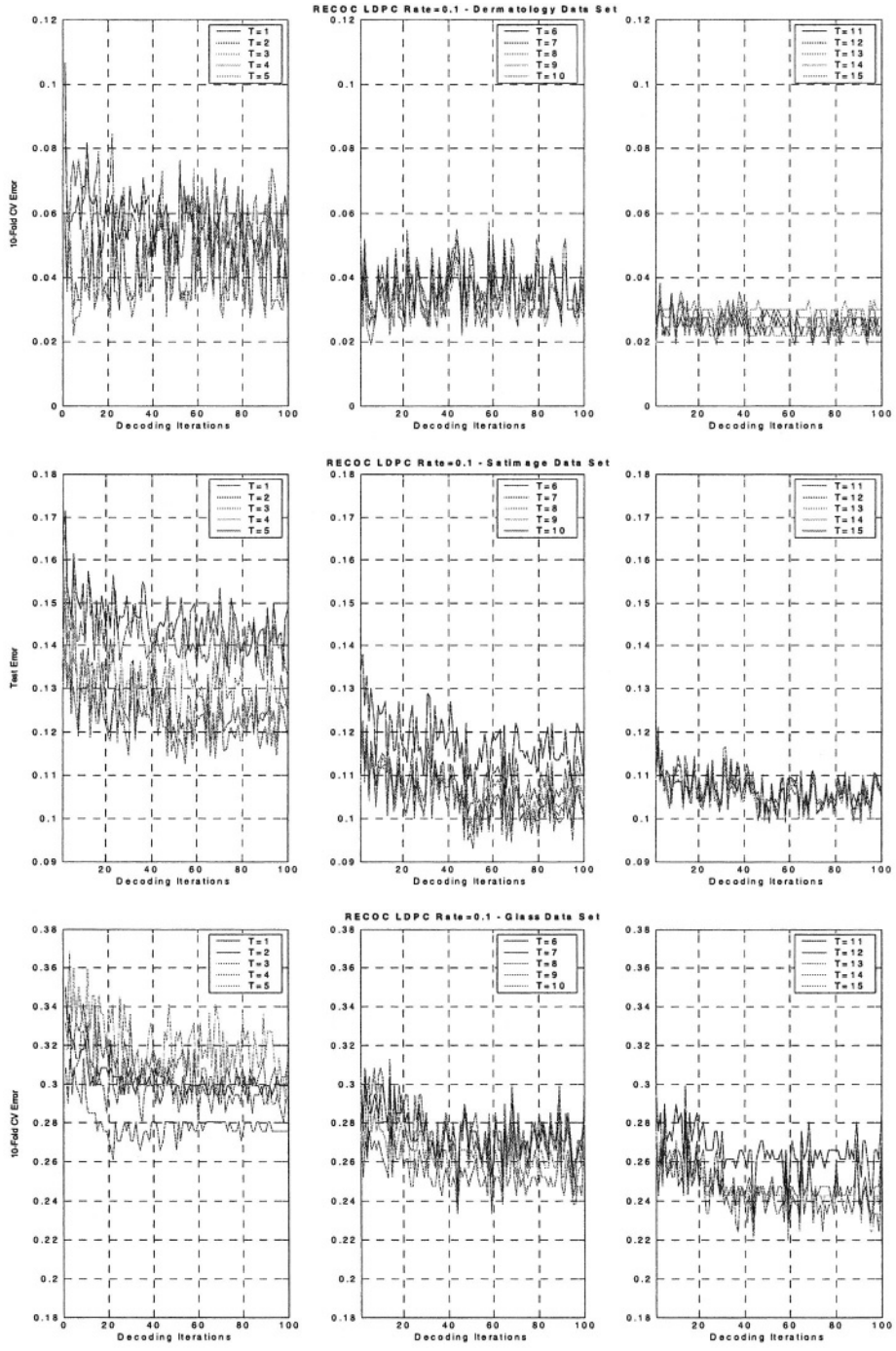


Fig. 2. Recoc LDPC $r = 0.1$ with Adaboost Decision Tree binary learners on Dermatology, Satimage and Glass data sets. The associated boosting level T is shown in the upper right.

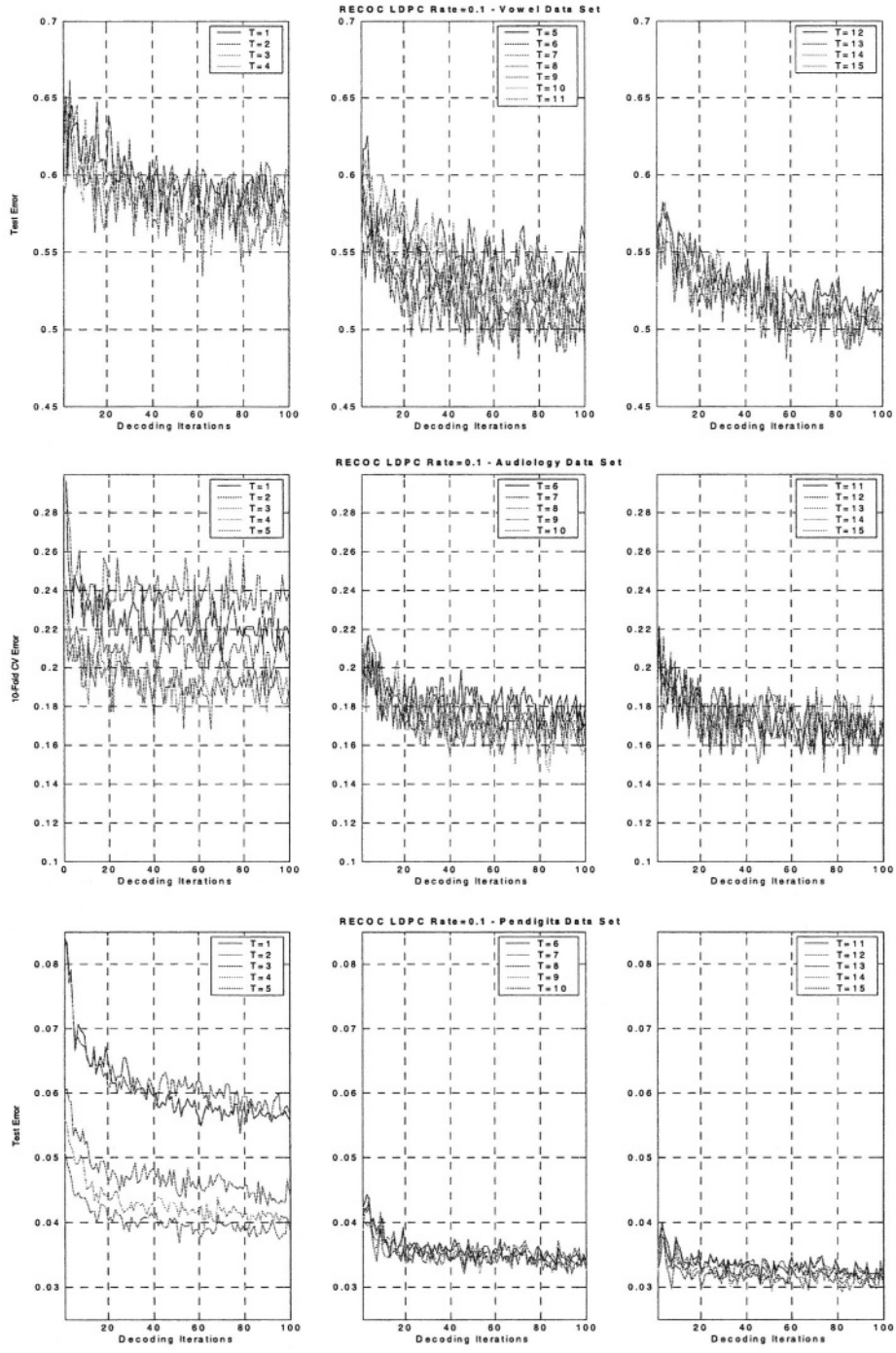


Fig. 3. Recoc LDPC $r = 0.1$ with Adaboost Decision Tree binary learners on Pendigits, Vowel and Audiology data sets. The associated boosting level T is shown in the upper right.

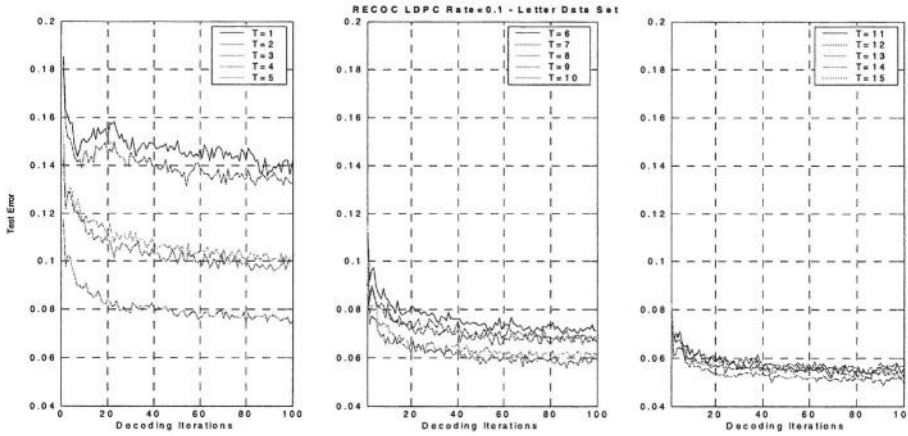


Fig. 4. Recoc LDPC $r = 0.1$ with Adaboost Decision Tree binary learners on Letter data set. The associated boosting level T is shown in the upper right.

Table 2. Error % achieved by RECOC LDPC $r = 0.1$, $j = 3$ using Decision Trees (DT) with boosting steps T∈ [1,15] and Decision Stumps (DS) with boosting steps T ∈ [1, 100] . Mean* and Std* are computed for I ∈ [75, 100], Min and Max over I ∈ [1, 100].

Boosted DT		Statistics				% Error				
Domain	T	Min	Max	Mean*	Std*	I=1	I=40	I=60	I=80	I=100
DE	15	1.91	3.83	2.22	0.1641	1.91	2.19	2.19	2.19	2.19
SA	8	9.30	12.25	10.10	0.3138	11.4	10.2	10.4	9.60	10.0
GL	14	21.96	28.04	24.37	0.9044	27.5	24.3	24.3	24.8	26.7
PE	14	2.92	3.80	3.07	0.0799	3.26	3.15	3.20	3.03	3.09
VO	11	48.2	56.1	49.81	0.8610	50.2	48.7	51.3	48.5	48.3
AU	13	14.6	21.7	16.47	0.5251	19.5	17.2	17.2	16.3	16.3
LE	15	4.85	7.62	5.10	0.0940	7.62	5.35	5.15	5.17	5.00
Boosted DS		Statistics				% Error				
PT	71	52.5	61.1	54.05	0.5860	61.1	55.7	53.1	54.5	54.2
DE	15	1.91	4.37	2.35	0.2060	4.37	2.19	2.19	2.73	2.19
SO	100	6.65	12.5	10.4	0.6486	12.5	10.9	10.6	10.9	9.57

Multiclass learning is improved when boosting binary learners. This effect is observed no matter the improvement is due to the leveraging of weak learners or improved independence in strong learners.

We remark that improvements persist even when binary and multiclass training errors attain perfect classification. Obtained results improve many instances shown in [4], and are closed to those in [15]. In addition, they are consistent with the thesis of Rifkin [16]. We note in pass, that in this study we have constrained ourselves to $r = 0.1$. Higher channel rates might yield to better results considering that the probability of repeated columns might reduced as well. We conjecture that the good performance of the simple One Against All strategy in certain learning domains is due to the effective ECOC structure with a set of orthogonal columns.

5 Conclusions and Further Work

We have experimentally shown that RECOC learning is a coding theory complaint ECOC learning framework. Knowledge is acquired in distributed an iterative way trusting in a well-bounded set of ECOC codes. A methodology for finding such sets has been presented. However, we feel that we have only scratch the problem. Clearly, the design of suitable scoring systems for qualifying ECOC matrices constructed from LDPC codes is an interesting line of research. In addition, it remains an extensive study about the behaviour of RECOC LDPC learning in a broad spectrum of channel rates.

Acknowledgments

The authors wish to thank anonymous reviewers for their valuable comments.

References

1. Tapia, E., González, J.C., García Villalba, J., Villena, J.: Recursive Adaptive ECOC models. In Proceedings EPIA 2001. Springer LNAI 2258, Oporto, Portugal (2001)
2. Dietterich, T., Bakiri, G.: Error-correcting output codes: A general method for improving multiclass inductive learning programs. Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), pp. 572-577, Anaheim, CA: AAAI Press (1991)
3. Crammer, K., Singer, Y.: On the Learnability and Design of Output Codes for Multiclass Problems. 35-46. N. Cesa-Bianchi, S. A. Goldman (Eds.): Proceedings COLT 2000, pp. 35-46, Palo Alto, California. Morgan Kaufmann (2000)
4. Allwein, E., Schapire, R., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. Journal of Machine Learning Research 1, pp. 113-141 (2000)
5. Tapia, E., González J.C., García, J.: Good Error Correcting Output Codes For Adaptive Multiclass Learning. In Proceedings of the MCS 2003. Springer LNCS, Surrey, UK (2003)
6. Tanner, M.: A recursive approach to Low Complexity Error Correcting Codes. IEEE Trans. Inf. Theory, Vol. 27, pp. 533 – 547 (1981)
7. Kschischang F., Frey, B.: Iterative decoding of compound codes by probability propagation in graphical models. IEEE Journal on Sel. Areas in Communications Vol. 16, No. 2, pp. 219-230(1998)
8. Gallager, R. G.: Low Density Parity-Check Codes. Cambridge, Massachusetts, M.I.T. Press (1963)
9. MacKay, D. J.: Good Error Correcting Codes based on Very Sparse Matrices. IEEE Trans. Inf. Theory, Vol. 45, pp. 399-431 (1999)
10. Guruswami, V., Sahai, A.: Multiclass Learning, Boosting, and Error-Correcting Codes. Proceedings COLT 99, pp. 145-155, Santa Cruz, CA, USA (1999)
11. Massulli, F., Valentini, G.: Dependence among Codeword Bits Error in ECOC Learning: An Experimental Analysis. In Proceedings of the MCS 2001. Springer LNCS 2096, Cambridge, UK (2001)
12. Schapire, R. E., Singer, Y.: Improved Boosting Algorithms Using Confidence - rated Predictions. Machine Learning, Vol. 37, No. 3, pp. 277-296 (1999)
13. Tapia, E., González, J.C., García-Villalba J.: A Generalized Class of Boosting Algorithms based on Recursive Error Correcting Codes. In Proceedings of the MCS 2001. Springer LNCS 2096, Cambridge, UK (2001)
14. Witten, I., Frank E.: Data Mining, Practical Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufmann Publishers, San Francisco, California (2000)
15. Furkranz, J. Round Robin Classification. Journal of Machine Learning Research, 2:721-747 (2002)
16. Rifkin, R.: Everything Old is New Again: A Fresh Look at the Historical Approaches in Machine Learning. Ph.D. Thesis. Massachusetts Institute of Technology (2002)

Exact Bagging with k-Nearest Neighbour Classifiers

Bruno Caprile, Stefano Merler, Cesare Furlanello, and Giuseppe Jurman

ITC-irst – Centro per la Ricerca Scientifica e Tecnologica
I-38050 Povo, Trento, Italy

Abstract. A formula is derived for the exact computation of Bagging classifiers when the base model adopted is k-Nearest Neighbour (k-NN). The formula, that holds in any dimension and does not require the extraction of bootstrap replicates, proves that Bagging cannot improve 1-Nearest Neighbour. It also proves that, for $k > 1$, Bagging has a smoothing effect on k-NN. Convergence of empirically bagged k-NN predictors to the exact formula is also considered. Efficient approximations to the exact formula are derived, and their applicability to practical cases is illustrated.

1 Introduction

This note is devoted to the derivation of a formula for the *exact* computation of the *Bagging* model [1], when the base predictor adopted is the k-Nearest Neighbour (k-NN) classifier [2].

The Bagging method essentially consists in averaging over a(n infinite) series of realizations of a same base predictor - in our case, k-NN. Virtues and limitations of methods based on combination of predictors ("ensemble" methods), as Bagging, *Boosting* [3] and, specifically, the *Adaboost* algorithm [4], have been extensively investigated – primarily, from the point of view of prediction performance [5,6].

k-Nearest Neighbour is one of the most basic (and oldest) classifiers; it is therefore interesting to study how it combines with Bagging. A first question is whether Bagging can improve k-NN at all. The formula we derive proves that 1-NN is perfectly equivalent to the bagged 1-NN, while, for $k > 1$, we are able to show that Bagging has a smoothing ("regularizing") effect on k-NN. This result adds theoretical grounding for a long recognized fact, i.e., that Bagging is a variance reducing procedure that works especially well with high variance (unstable) base models [7,1]. While consistency of Adaboost is still under scrutiny [8], the formula also implies that the limit exists for empirically bagged k-NN predictors. Convergence of the latter to the exact formula can thus be gauged under a variety of conditions. The formula is straightforward to implement. Efficient approximations to it can also be derived, which we propose for applicability in practical cases.

In Sec. 2, notations are introduced and background material on k-NN and Bagging is reviewed. Main results are contained in Sec. 3, where detailed deriva-

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 72–81, 2004.

tion of the exact and approximated formulae for the bagged k-NN predictor is provided. Because of space limitations, some proofs are omitted and can be found in [9][1]. In Sec. 4, application of main results is illustrated in concrete cases.

2 Background

Let $D \equiv \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be a *data set,* where the *data points* $\mathbf{x}_i$'s belong to a given region, A, of some metric space (X, d), and let the y_i's be class labels. In the following, we restrict ourselves to the problem of binary classification, and we shall therefore assume that $y_i \in \{-1, 1\}$, $i = 1, \ldots, N$.

In this paper, a *predictor* is a function $\phi(\cdot)$, returning a class label y for any input pattern $x \in A$, that is synthesized from a given set of examples (*i.e.,* a data set D), through some specific (*learning*) process. So, a predictor depends on the data set on which it is trained, and we may find convenient to convey this dependence by adopting the explicit notation: $\phi(\cdot\,; D)$, $\phi : A \longrightarrow \{-1, 1\}$.

For the sake of notational simplicity, we will occasionally employ the same letter, D, also to denote the set of training points, $\mathbf{x}_i$'s, $D \equiv \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$.

Nearest Neighbour is a predictor over region A that assigns to point $\mathbf{x} \in A$ the label associated with the data point $\mathbf{x}_{i^*} \in D$ which is closest (with respect to the metric of A) to x. In the same spirit, a k-NN classifier model is obtained by considering the first k data points nearest to point x, and classifying point x as belonging to class 1 or –1 according to a majority voting criterion (for which it is customary to take k odd) [2].

Bagging is an *ensemble method* for building classifier predictors, which combines those resulting from training a same base predictor over a suitable collection of training sets, D_b. More specifically, let D_b be a collection of data sets, each lying in $A \times \{-1,1\}$. An aggregated predictor $\phi_{\mathcal{A}}(\cdot)$ is obtained by training the same base model over the D_b's and combining the different predictors obtained through a majority voting criterion. Let $\mathbf{x} \in A$, and let B_1 indicate the fraction of D_b's for which $\phi(\mathbf{x}; D_b) = 1$; then $\phi_{\mathcal{A}}(\mathbf{x}) = 1$ if $B_1 > B/2$, and $\phi_{\mathcal{A}}(\mathbf{x}) = -1$ otherwise.

Usually, however, only a single training set D is available. The Bagging strategy [1] consists in building a collection D_b by *Bootstrap* [10] resampling: B replicated data sets D_b, $b = 1, \ldots, B$ are obtained by random sampling with replacement exactly N cases from the original data set D. The resulting predictor, that we denote with $\phi_{\mathcal{B}}(\cdot)$, is an approximation of the aggregated predictor $\phi_{\mathcal{A}}(\cdot)$.

As it is well known, the probability, $\pi(N)$, that any given sample is contained in a bootstrap replicate of a given data set of cardinality N is given by $\pi(N) = 1 - (1 - \frac{1}{N})^N$. As N increases, $\pi(N)$ rapidly converges to a value slightly greater than 0.632. For the sake of notational simplicity, in the following we drop the explicit dependence of π from N.

3 Main Results

Here we derive the exact formula for the computation of the bagging predictor when the base predictor adopted is k–NN, for any k. In particular, for any input

[1] Available online at the following address: http://mpa.itc.it/mcs2004/bagging-knn.ps

pattern $\mathbf{x} \in A$ we will compute $P(\phi_{\mathcal{B}}(\mathbf{x}) = 1)$, the probability that the Bagging predictor map $\mathbf{x}$ to class 1. We start with the $k = 1$ case, and extend then the argument to $k > 1$.

In all that follows, the training set D is intended to be re-sorted in ascending order of distance from $\mathbf{x}$: $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where $d(\mathbf{x}_1, \mathbf{x}) \leq d(\mathbf{x}_2, \mathbf{x}) \leq \cdots \leq d(\mathbf{x}_N, \mathbf{x})$.

3.1 The 1-NN Case

In the case $k = 1$ we will compute the probability with respect to the class y_1 of the first nearest neighbor $\mathbf{x}_1$, *i.e.,* we will compute $P(\phi_{\mathcal{B}}(\mathbf{x}) = y_1)$. We start by computing the probability $P(\phi(\mathbf{x}; D_b) = y_1)$, where D_b is a generic bootstrap replicated data set of D. The following result holds:

Proposition 1. *Let $\mathbf{x} \in A$ be any given input pattern and let D_b be a generic bootstrap replicated data set. The probability of mapping $\mathbf{x}$ to the class y_1 of its first nearest neighbor by means of a $1-NN$ predictor is at least π,* i.e., $P(\phi(\mathbf{x}; D_b) = y_1) \geq \pi$.

Proof. The probability that a given data point $\mathbf{x}_i \in D$ determines the class to be assigned to x is $p_i = \pi(1-\pi)^{i-1}$, *i.e,* the probability, π, that $\mathbf{x}_i$ is extracted in the replicated data set D_b, multiplied by the probability, $(1-\pi)^{i-1}$, that no points $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}\}$ are extracted that are closer to $\mathbf{x}$ than $\mathbf{x}_i$.
The probability of mapping $\mathbf{x}$ to class y_1 is therefore given by the sum of the probabilities p_i over the points of class y_1. Let $\mathcal{I} = \{i \in \mathbb{N} \mid y_i = y_1,\ 1 \leq i \leq N\}$ be the set of indices of the points of class y_1, then

$$P(\phi(\mathbf{x}; D_b) = y_1) = \pi \sum_{i \in \mathcal{I}} (1-\pi)^{i-1}. \tag{1}$$

By isolating the contribution of the first nearest neighbor $\mathbf{x}_1$, Eq. (1) can be rewritten as

$$P(\phi(\mathbf{x}; D_b) = y_1) = \pi + \pi \sum_{\substack{i \in \mathcal{I} \\ i > 1}} (1-\pi)^{i-1} \geq \pi. \quad \square$$

We now compute the probability that the bagging predictor agrees at point $\mathbf{x}$ with the prediction of the 1-NN.

Proposition 2. *Let $\mathbf{x} \in A$ be any given input pattern. The probability of mapping $\mathbf{x}$ to the class y_1 of its first nearest neighbor by means of a bagging 1–NN predictor satisfies the following relationship:*

$$\lim_{B \to \infty} P(\phi_{\mathcal{B}}(\mathbf{x}) = y_1) = 1 \ .$$

Proof. Let us suppose that a collection of B different versions of the base predictor obtained by learning over Bootstrap-replicated data sets is available. The

evaluation of predictors at $\mathbf{x}$, $\{\phi(\mathbf{x}; D_b)\}_{b=1,\ldots,B}$, is equivalent to random labels generation (either -1 or 1) from a binomial distribution $b(n, p)$. The parameters of the distribution are $n = B$, the number of replicated data sets, and the probability p is given in Eq. (1) (considering that the outcome y_1 correspond to "success").

By the law of large numers we know that the proportion of successes (outcomes y_1) converges to p in probability as B grows to infinity. Combined with Prop. 1, this ensures that the proportion of outcomes y_1 is at least π if B is sufficiently large. As a consequence: $\lim_{B\to\infty} P(\phi_B(\mathbf{x}) = y_1) = 1$. □

The interpretation is immediate: 1-NN and (infinitely) bagged 1-NN coincide.

3.2 The k-NN Case

We now extend the result of the previous section to the k-NN case, for $k > 1$. The overall strategy is the same: first we compute the probability $P(\phi(\mathbf{x}; D_b) = 1)$, and then we exploit the law of large numbers. The derivation of the formula for $P(\phi(\mathbf{x}; D_b) = 1)$ follows the same ideas as above, only we have to consider the probability of extraction of *subsets* of data points, rather than single points. To this end, the class α_r of subsets of D is first introduced.

Definition 1. *The class α_r is the set of all the replicated data sets containing exactly $k-1$ data closer to* x *than $\mathbf{x}_r$, and containing $\mathbf{x}_r$ as the k-th closest point to* x;

$$\alpha_r = \{D_b \subset D \mid D_b = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{k-1}}, \mathbf{x}_r, \ldots\},\ i_j < r,\ 1 \leq j \leq k-1\}.$$

Remark 1. Let $\mathcal{P}(D)$ be the power set of D and let $\mathcal{P}_k(D) \subset \mathcal{P}(D)$ be the the set of all subsets of D containing at least k elements: $\{\alpha_r\}_{r=k,\ldots,N}$ is a partition, *i.e.*, a disjoint covering, of $\mathcal{P}_k(D)$. This means that $\bigcup_{r=k}^{N} \alpha_r = \mathcal{P}_k(D)$ and $\alpha_r \cap \alpha_s = \emptyset$ for $r \neq s$.

Proposition 3. *Let $\mathbf{x} \in A$ be a given input pattern and let D_b be a generic bootstrap replicated data set. The probability of mapping* x *to class* 1 *by means of a $k-NN$ predictor is*

$$P(\phi(\mathbf{x}; D_b) = 1) = \sum_{r=k}^{N} P(\phi(\mathbf{x}; D_b) = 1 \mid D_b \in \alpha_r) P(D_b \in \alpha_r). \tag{2}$$

Proof. For fixed r, let $P(D_b \in \alpha_r,\ \phi(\mathbf{x}; D_b) = 1)$ be the joint probability of extracting a bootstrap data set belonging to α_r leading to majority vote 1. Since the classes α_r are a partition of $\mathcal{P}_k(D)$, the probability that a generic bootstrap replicated data set D_b leads to majority vote equal to 1 is obtained by summing $P(D_b \in \alpha_r,\ \phi(\mathbf{x}; D_b) = 1)$ over r. Eq. (2) follows immediately from the definition of conditional probability. □

Our aim is now to provide expressions for the two terms in the summation. To this end, let us introduce the following

Definition 2. *$S(r)$ is number of elements of α_r leading to majority vote* 1;

$$S(r) = \mathrm{Card}(\{D_b \in \alpha_r \mid \phi(\mathbf{x}; D_b) = 1\}).$$

Remark 2. The probability that a bootstrap replicated data set belonging to α_r leads to majority vote 1 is

$$P(\phi(\mathbf{x}; D_b) = 1 \mid D_b \in \alpha_r) = \frac{S(r)}{\mathrm{Card}(\alpha_r)}, \tag{3}$$

and the number of elements belonging to class α_r is

$$\mathrm{Card}(\alpha_r) = 2^{N-r}\binom{r-1}{k-1}. \tag{4}$$

In fact, the binomial coefficients accounts for all the possible choices of $k-1$ points from the first $r-1$ ($\mathbf{x}_r$ is fixed) and the term 2^{N-r} accounts for all the possible combinations of the remaining $N-r$ points of D.
As for the term $S(r)$, the following result holds (proof in [9]):

Proposition 4. *The number of elements of α_r leading to majority vote 1 is $S(r) = 2^{N-r}\tilde{S}(r)$ where*

$$\tilde{S}(r) = \sum_{j=\frac{k-y_r}{2}}^{k-1} \binom{\nu_1(r-1)}{j}\binom{\nu_{-1}(r-1)}{k-1-j}$$

and $\nu_1(r-1)$ and $\nu_{-1}(r-1)$ indicate the number of points in $D_{r-1} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}\}$, respectively labelled with 1 *and* -1.

For each r, computation of $S(r)$ involves dealing with summation of binomial coefficients, which, as N grows, makes Eq. (3) computationally impractical. In [9] a more efficient, recursive way for computing $S(r)$ is presented.

We work now at the derivation of the second probability term in the summation of Eq. (2). To this end, let $\overline{D_b(r)} = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{k-1}}, \mathbf{x}_r, \ldots\}$ be a given element of α_r, i.e, the indices i_j are fixed; moreover, let $P(\overline{D_b(r)})$ be its probability of extraction.

Proposition 5. *The probability $P(D_b \in \alpha_r)$ of extracting a generic bootstrap replicated data set D_b belonging to the class α_r is*

$$P(D_b \in \alpha_r) = \binom{r-1}{k-1} P(\overline{D_b(r)}).$$

Proof. immediate, considering that the binomial coefficient accounts for all the of possible choices of $k-1$ points from the first $r-1$ and $\mathbf{x}_r$ isfixed. □

Putting together Prop. 4 and 5, Eqs. (3) and (4) we can rewrite Eq. (2) as:

$$P(\phi(\mathbf{x}; D_b) = 1) = \sum_{r=k}^{N} \tilde{S}(r) P(\overline{D_b(r)}) \tag{5}$$

As for the term $P(\overline{D_b(r)})$, the following result holds (proof in [9]):

Proposition 6. *Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_n, \mathbf{v}_1, \ldots, \mathbf{v}_m\} \subseteq D$ be a subset of the training data. The probability of extracting a bootstrap replicated data set D_b containing the $\mathbf{u}_i$'s and not containing the $\mathbf{v}_i$'s is*

$$P(\mathbf{u}_1 \in D_b, \ldots, \mathbf{u}_n \in D_b, \mathbf{v}_1 \notin D_b, \ldots, \mathbf{v}_m \notin D_b) = P_{n0} \prod_{i=0}^{m-1} (1 - p_i),$$

where $p_i = 1 - \left(1 - \frac{1}{N-i}\right)^N$ (according to this definition $p_0 = \pi$) and P_{n0} can be computed recursively as it follows:

$$\begin{cases} P_{1j} = p_{m+j} & j = 0, \ldots, n-1 \\ P_{ij} = \left(1 - \frac{P_{i-1\ j+1}}{P_{i-1\ j}}(1 - p_m)\right) P_{i-1\ j} & i = 2, \ldots, n \qquad j = 0, \ldots, n-i. \end{cases}$$

Therefore, $P(\overline{D_b(r)}) = P_{k0} \prod_{i=0}^{r-k-1} (1 - p_i)$, and Eq. (5) can be rewritten as

$$P(\phi(\mathbf{x}; D_b) = 1) = P_{k0} \sum_{r=k}^{N} \tilde{S}(r) \prod_{i=0}^{r-k-1} (1 - p_i). \tag{6}$$

Now we can finally state our main claim:

Proposition 7. *Let $\mathbf{x} \in A$ be any given input pattern. The probability of mapping $\mathbf{x}$ to class 1 by means of a bagging $k-NN$ predictor satisfies the following relationship:*

$$\lim_{B \to \infty} P(\phi_B(\mathbf{x}) = 1) = \begin{cases} 1 \text{ if} & P(\phi(\mathbf{x}; D_b) = 1) > \frac{1}{2} \\ \frac{1}{2} \text{ if} & P(\phi(\mathbf{x}; D_b) = 1) = \frac{1}{2} \\ 0 \text{ otherwise.} \end{cases}$$

Proof. The proof follows the same argument as in Prop. 2. □

Remark 3. It is interesting to notice that

$$\lim_{N \to \infty} P(\overline{D_b(r)}) = \pi^k (1 - \pi)^{r-k}. \tag{7}$$

Beside providing a much simpler expression for $P(\overline{D_b(r)})$, this fact has an interesting interpretation: in the limit $N \to \infty$, the probability of extracting a

given replicated data set, $\overline{D_b(r)}$, is equal to the probability, π^k, of extracting $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{k-1}}$ and $\mathbf{x}_r$, multiplied by the probability $(1-\pi)^{r-k}$ of not extracting the remaining $(r-1)-(k-1)$ elements. This is the same as saying that, for large N, extractions of samples from the data set can be regarded as independent events.

Eq. (7) allows to write an approximated formula for Eq. (6):

$$P(\phi(\mathbf{x}; D_b) = 1) = \pi^k \sum_{r=k}^{N} \tilde{S}(r)\ (1-\pi)^{r-k}, \tag{8}$$

In Sec. 4, applicability of Eq. (8) in place of the exact one is illustrated in practical cases.

Remark 4. The smoothing effect of Bagging becomes apparent as we rearrange terms of Eq. (6) as it follows

$$P(\phi(\mathbf{x}; D_b) = 1) = \underbrace{P_{k0}\tilde{S}(k)}_{k \text{ terms}} + \underbrace{P_{k0}(1-p_0)\tilde{S}(k+1)}_{k+1 \text{ terms}} + \underbrace{P_{k0}(1-p_0)(1-p_1)\tilde{S}(k+2)}_{k+2 \text{ terms}} + \ldots,$$

and observe that $P_{n0}\prod_{i=0}^{m-1}(1-p_i)$ is a decreasing function of both n and m. This fact implies that the main contribution to $P(\phi(\mathbf{x}; D_b) = 1)$ is given by the first k terms, but also all the other terms contribute, in a decreasing way, to it.

Remark 5. Formulas for $P(\phi(\mathbf{x}; D_b) = 1)$ do not account for the bootstrap replicated data sets containing less than k elements. However, let $\overline{D_b} = \{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_s}\}$, with $s < k$. The probability of extracting any such data set is:

$$P(\overline{D_b}) = \sum_{r=1}^{k-1} \binom{N}{r} P_{r0} \prod_{i=0}^{N-r-1} (1-p_i) \xrightarrow[N\to\infty]{} 0.$$

This means that, for N large enough, it is a negligible probability.

4 Examples

Two applications of formulas derived in Sec. 3 are now illustrated.

4.1 Task 1

The first task consists in comparing estimates of $P(\phi_{\mathcal{B}}(\mathbf{x}) = 1)$ as provided by the exact formula (6), with those provided by the approximation (8) of it and by empirically bagged k-NN predictors. Moreover we will also consider truncated versions of Eq. (8):

Table 1. Misclassification rates of different k-NN predictors as compared to the exact one of Eq. (6). Columns denoted with 'E' refer to empirically bagged predictors built by aggregationg 20, 50, 100, 200, 500 base models. 'Appr.' denotes the approximate predictor of Eq. (8), while "Tr." denotes predictors derived from the approximate one by truncating the summation at $N^\star = 8, 12, 16, 20$ terms.

k	N	E 20	E 50	E 100	E 200	E 500	Appr.	Tr. 8	Tr. 12	Tr. 16	Tr. 20
1	20	1.9	0.2	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
1	50	2.5	0.3	0.0	0.0	0.0	0.0	**0.0**	0.0	0.0	0.0
3	20	6.8	4.3	3.6	2.0	1.3	1.0	**1.2**	0.9	1.0	1.0
3	50	8.9	7.4	5.4	5.0	2.5	0.3	**2.2**	0.3	0.3	0.3
5	20	7.4	4.6	4.0	2.6	2.4	0.9	15.5	**1.5**	1.0	0.9
5	50	6.8	4.0	2.8	1.8	1.2	0.3	16.0	**0.7**	0.3	0.3
7	20	7.9	4.5	3.3	2.4	1.6	1.4	48.4	12.1	**1.7**	1.4
7	50	7.8	4.5	3.4	3.0	1.7	0.3	50.4	12.3	**1.0**	0.2

$$P(\phi(\mathbf{x}; D_b) = 1) = \pi^k \sum_{r=k}^{N^\star} \tilde{S}(r)\ (1-\pi)^{r-k}, \text{ with } k < N^\star < N.$$

To this end, N labels in $\{-1, 1\}$ are randomly generated as to simulate sorting of data points in ascending distance from a "very difficult" test point (it is in the proximity of the class boundary that labels of neighbours are mostly affected by noise). For each such sequence of labels, a *true* class is computed as the one returned by the exact formula. The true class is then compared with those returned by approximated formulas and empirical bagging as obtained by aggregating an increasing number of k-NN predictors. Results reported in Tab. (1) are averaged over 1000 runs of the procedure.

The first thing to notice is that, for any reported k and N, the disagreement between empirical bagging and the exact formula decreases as the number of aggregated models is increased. While the task is somewhat extreme, it is interesting to see how, for $k > 1$, 500 aggregated models never suffice to reduce the disagreement with the exact formula to within 1%. Secondly, outcomes of the approximated formula are in excellent agreement with those of the exact formula (as expected by Eq. (7), the agreement is remarkably better for $N = 50$). Finally, the approximated formula can be truncated drastically (depending on k), without much affecting the outcomes. Bold figures in Tab. (1) refer to the minimum number of terms in the summation necessary to perform better than the empirically bagged predictor built with 500 aggregated models.

4.2 Task 2

The second task consists in comparing the classification performance of the exact, approximate and empirically bagged predictors. The example reported refer to a relatively simple, two-dimensional problem: 4 sets of points, 25 points each, were generated by sampling 4 two-dimensional Gaussian distributions, respectively centered in (-1,1/2), (0, $-1/2$), (0, 1/2) and (1, $-1/2$). Covariance matrices were diagonal for all the 4 distributions; variance was constant and equal to 2/5. Points coming from the sampling of the first two Gaussians were labelled with class -1;

Table 2. Misclassification rates of different k-NN predictors. Notations are the same of Tab. 1, except for the second column, which refers to the plain (unbagged) k-NN predictor, and the column "Exact", which refers to the exact formula (6).

k	k-NN	E 20	E 50	E 100	E 200	E 500	Exact	Appr.	Tr. 8	Tr. 12	Tr. 16	Tr. 20
1	13.2	13.0	13.28	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2	13.2
3	13.84	10.24	9.36	8.84	8.84	8.68	8.72	8.72	8.76	8.72	8.72	8.72
5	9.0	6.64	6.36	6.04	6.2	6.24	6.12	6.08	8.2	6.24	6.08	6.08
7	6.64	5.72	5.64	5.56	5.36	5.4	5.32	5.32	48.72	5.8	5.36	5.32

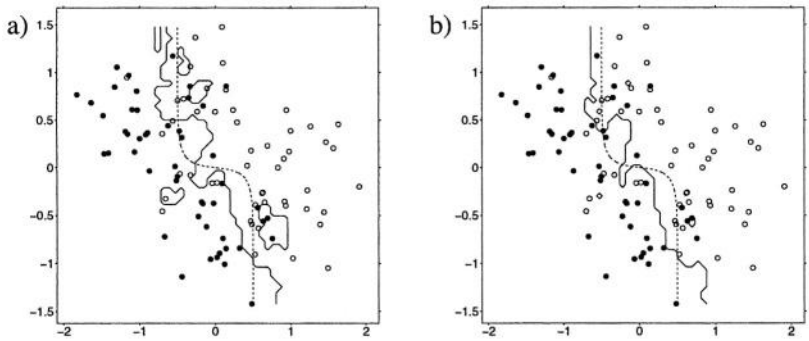


Fig. 1. Class boundaries (solid lines) obtained by applying a) plain 5-NN, b) bagged 5-NN predictors. The dashed lines correspond to the Bayes decision rule.

the others with class 1. Misclassification rates of predictors are measured against the optimal Bayes predictor on 2500 test points lying on an evenly spaced grid. Results are collected in Tab. 2

First we notice that, as expected, for $k = 1$ the plain k-NN predictor performs as the exact bagging predictor (see Prop. 2). Second, for $k > 1$ bagged predictors perform consistently better than plain k-NN. Thirdly, as the number of aggregated models increases, the empirical bagging predictors converge rapidly to the exact predictor. In fact, 100 aggregated models are sufficient to match the performance of the exact formula. For what concerns the approximated and truncated predictors, the agreement with the exact formula is excellent, and perfectly consistent with that observed in Case 1 above.

In Fig. 1 the class boundaries obtained by applying plain 5-NN and bagged 5-NN are shown: the misclassification error is of about 9% for plain 5-NN and 6% for bagged 5-NN (see Tab. 2). The figure clearly shows the smoothing effect obtained by applying bagging which allows, in this noisy problem, to improve predictions. The same behaviour was also observed for k equal to 3 and 7.

5 Conclusions

Let us finally summarize the main points of this paper:

a. for k-Nearest Neighbour predictors, Bagged models can be computed in exact, closed form;

b. the formula shows that: (1) Bagged 1-NN is perfectly equivalent to plain 1-NN; (2) for $k > 1$, Bagging has a smoothing effect on k-NN;
c. efficient approximations to the formula can be derived, which prove effective in typical cases.

As for further developments, we believe it would be interesting to investigate whether the method can be extended to predictors other than k-NN. To this end, let us notice that the partition $\{\alpha_r\}_{r=k,\ldots,N}$ of the set $\mathcal{P}_k(D)$ does not depend on the specific predictor ϕ adopted. The same consideration holds for Prop. 3, which allows decomposing the probability $P(\phi(\mathbf{x}; D_b) = 1)$ in terms of $P(\phi(\mathbf{x}; D_b) = 1 \mid D_b \in \alpha_r)$ and $P(\alpha_r \in D_b)$, as well as for Props. 5 and 6, which allow computing $P(\alpha_r \in D_b)$. Thus, at least in principle, everything reduces to estimating $P(\phi(\mathbf{x}; D_b) = 1 \mid D_b \in \alpha_r)$ for the predictor of choice. A promising next step may consist in deriving controlled approximations for kernel-based predictors as Radial Basis Functions or Support Vector Machines.

Acknowledgements

Authors wish to thank P. Tonella which made useful comment and an anonymous referee for highlighting weak points in an early version of the paper, thereby prompting several improvements.

References

1. Breiman, L. (1996). Bagging Predictors. *Machine Learning,* 24, 123–140.
2. Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* John Wiley, New York.
3. Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proc. 13th Int. Conf. Mach. Learn.* (pp. 148–146). San Francisco, CA: Morgan Kaufmann.
4. Freund, Y., & Schapire, R. E. (1997). A Decision-theoretic Generalization of Online Learning and an Application to Boosting. *J. Comput. Syst. Sci.,* 55:1, 119–139.
5. Quinlan, J. R. (1996). Bagging, Boosting, and C4.5. In *Proc. 13th Nat. Conf. on AI* (pp. 725–730). Cambridge, MA: AAAI Press/MIT Press.
6. Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple Classifier Systems,* (pp. 1–15). Springer-Verlag.
7. Buhlmann, P., & Yu, B. (2002). Analyzing Bagging. *Annals of Statistics,* 30, 927–961.
8. Jiang, W. (2000). Process Consistency for AdaBoost. Technical report, Dept. of Statistics, Northwestern University.
9. Caprile, B., Merler, S., & Furlanello, C. (2001). Exact Bagging with k-Nearest Neighbour Classifiers. Technical report, IRST, 0106-03.
10. Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap.* Chapman & Hall, New York.

Yet Another Method for Combining Classifiers Outputs: A Maximum Entropy Approach

Marco Saerens and François Fouss

Information Systems Research Unit (ISYS – IAG)
Université catholique de Louvain
Place des Doyens 1
B-1348 Louvain-la-Neuve, Belgium
{saerens,fouss}@isys.ucl.ac.be

Abstract. In this paper, we present a maximum entropy (maxent) approach to the fusion of experts opinions, or classifiers outputs, problem. The maxent approach is quite versatile and allows us to express in a clear, rigorous, way the a priori knowledge that is available on the problem. For instance, our knowledge about the reliability of the experts and the correlations between these experts can be easily integrated: Each piece of knowledge is expressed in the form of a linear constraint. An iterative scaling algorithm is used in order to compute the maxent solution of the problem. The maximum entropy method seeks the joint probability density of a set of random variables that has maximum entropy while satisfying the constraints. It is therefore the "most honest" characterization of our knowledge given the available facts (constraints). In the case of conflicting constraints, we propose to minimise the "lack of constraints satisfaction" or to relax some constraints and recompute the maximum entropy solution. The maxent fusion rule is illustrated by some simulations.

1 Introduction

The fusion of various sources of knowledge has been an active subject of research since more than three decades (for some review references, see [2], [6], [8]). It has recently been successfully applied to the problem of classifiers combination or fusion (see for instance [13]).

Many different approaches have been developped for experts opinions fusion, including weighted average (see for instance [2], [8]), Bayesian fusion (see for instance [2], [8]), majority vote (see for instance [1], [12], [16]), models coming from incertainty reasoning: fuzzy logic, possibility theory [14] (see for instance [3]), standard multivariate statistical analysis techniques such as correpondence analysis [18], etc. One of these approaches is based on maximum entropy modeling (see [17], [19]). Maximum entropy is a versatile modeling technique allowing to easily integrate various constraints, such as correlation between experts, reliability of these experts, etc.

In this work, we propose a new model of experts opinions integration, based on a maximum entropy model (for a review of maximum entropy theory and

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 82–91, 2004.

applications, see for instance [7], [9], [10] or [11]). In this paper, we use the term "experts opinions", but it should be clear that we can use exactly the same procedures for "classifiers combination". In other words, we could substitute "experts" by "classifiers" everywhere.

Here is the rationale of the method. Each expert expresses his opinion about the outcome of a random event, $y = i$, in the form of an a posteriori probability density, called a score. These scores are subjective expectations about this event. We also suppose that we have access to a reliability measure for each expert, for instance in the form of a probability of success, as well as a measure of the correlation between the experts. Each of these measures are combined properly by maximum entropy in order to obtain a joint probability density. Let us recall that the maximum entropy density is the density that is "least informative" while satisfying all the constraints; i.e. it does not introduce "extra ad hoc information" that is not relevant to the problem. Once this joint density is found, we compute the a posteriori probability of the event by averaging all the possible situations that can be encountered, i.e. by computing the marginal $\mathrm{P}(y = i|\mathbf{x})$, where $\mathbf{x}$ is the feature vector on which we base our prediction.

While the main idea is similar, our model differs from [19] in the formulation of the problem (we focus on quantities that are relevant to classification problems, and can easily be computed for classifiers: success rate, degree of agreement, etc) and in the way the individual opinions are aggregated. Furthermore, we also tackle the problem of incompatible constraints; that is, when there is no feasible solution to the problem, a situation that is not mentionned by [19].

Section 2 introduces the problem and our notations. Section 3 develops the maximum entropy solution. Section 4 presents some simulations results. Section 5 is the conclusion.

2 Statement of the Problem

Suppose we observe the outcome of a set of events, $\mathbf{x}$, as well as a related event, y, whose outcomes belong to the set $\{1, 2, \ldots, n\}$. We hope that the random vector $\mathbf{x}$ provides some useful information that allows to predict the outcome of y with a certain accuracy.

We also assume that domain experts (m experts in total) have expressed their opinion on the event y, based on the observation of $\mathbf{x}$: We denote by $d(k) = i$, with $i \in \{1, 2, \ldots, n\}$ and $k = 1, \ldots, m$, the fact that expert k chooses the outcome or alternative i – in other words, he takes decision i. In this framework, $\mathrm{P}(d(k) = i|\mathbf{x})$ will be interpreted as the personal expectation of the expert, i.e. the proportion of times a given expert k would choose alternative i, when observing $\mathbf{x}$ (for a general introduction to the concept of subjective probabilities, see [16]).

Our objective is to seek the joint probability density of the event, $y = i$, as well as the experts opinions, $d(k) = i_k$:

$$\mathrm{P}(y = i, d(1) = i_1, d(2) = i_2, \ldots, d(m) = i_m|\mathbf{x}) \tag{2.1}$$

This joint probability density will be estimated by using a maximum entropy argument that will be presented in the next section.

Prior knowledge on the problem, including expert's opinions, will be expressed as linear constraints on this joint density (2.1). In our case, there will be four different types of constraints (detailed in the four following subsections): constraints ensuring that (2.1) is a **probability density** (it sums to one), constraints related to the **opinion** of the experts, constraints related to the **reliability** of the experts, and constraints related to the **correlation** between experts.

2.1 Constraints Inducing a Probability Density

The first constraint simply states that the joint density sum to one:

$$\sum_{i,i_1,\ldots,i_m=1}^{n} \mathrm{P}(y=i, d(1)=i_1, d(2)=i_2, \ldots, d(m)=i_m|\mathbf{x}) = 1 \qquad (2.2)$$

This constraint will be called the sum (*sum*) constraint. Of course, we should also impose that the joint density is always positive, but this is not necessary (maximum entropy estimation leads to positive values, so that this constraint will be automatically satisfied).

2.2 Constraints Related to the Opinions of the Experts

Here, we provide information related to the expert's opinions. We will consider that each expert expresses his opinion about the outcomes, according to the observation $\mathbf{x}$:

$$\mathrm{P}(d(k)=i_k|\mathbf{x}) = \pi(d(k)=i_k|\mathbf{x}) \text{ for } k=1\ldots m, i_k=1\ldots n \qquad (2.3)$$

where $\pi(d(k)=i_k|\mathbf{x})$, the likelihood of choosing alternative i_k, is provided by expert k for each outcome $i_k \in \{1,2,\ldots,n\}$. In other words, each expert provides his likelihood of observing outcome i_k according to his subjective judgement. It indicates that, in average, expert k would choose alternative i_k with probability $\pi(d(k)=i_k|\mathbf{x})$ when he observes evidence $\mathbf{x}$. Of course, in the case of classifiers combination, each $\pi(d(k)=i_k|\mathbf{x})$ would correspond to classifier's outputs (assuming that each classifier provides as outputs a posteriori probabilities of belonging to a given class). This constraint will be called the opinion *(op)* constraint. Notice that (2.3) can be rewritten as

$$\sum_{i,i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_m} \mathrm{P}(y=i, d(1)=i_1, \ldots, d(m)=i_m|\mathbf{x}) = \pi(d(k)=i_k|\mathbf{x}) \quad (2.4)$$

for $k=1\ldots m$ and $i_k=1\ldots n$. Or, equivalently,

$$\sum_{i,i_1,\ldots,i_m} \delta(i_k-j_k)\,\mathrm{P}(y=i, d(1)=i_1, \ldots, d(m)=i_m|\mathbf{x}) = \pi(d(k)=j_k|\mathbf{x}) \quad (2.5)$$

where δ is the delta of Kronecker. This form (involving Kronecker deltas) is better suited for maximum entropy computation (see for instance [9]).

2.3 Constraints Related to the Reliability of the Experts

Some experts may be more reliable than others. We can express this fact by, for instance, recording the success rate of each expert. This can be expressed formally by

$$\sum_i \mathrm{P}(y = i, d(k) = i|\mathbf{x}) = \Delta(k|\mathbf{x}) \text{ for } k = 1 \ldots m \tag{2.6}$$

$\Delta(k|\mathbf{x})$ can be interpreted as the success rate for expert k, the probability of taking the correct decision (the probability that the opinion of the expert and the outcome of the event agree) when observing $\mathbf{x}$. If $\Delta(k|\mathbf{x}) = 1$, expert k is totally reliable in the sense that the judgement of the expert and the outcome of the experiment always agree. On the other hand, if $\Delta(k|\mathbf{x}) = 0$, expert k is always wrong (he always disagrees with the outcome of the experiment). Now, if the reliability is only known without reference to the context $\mathbf{x}$ (or we do not have access to this detailled information) we could simply state that it is independent of $\mathbf{x}$ which, of course, is much more restrictive:

$$\sum_i \mathrm{P}(y = i, d(k) = i|\mathbf{x}) = \Delta(k) \text{ for } k = 1 \ldots m \tag{2.7}$$

Where we do not require knowledge of the probability of success for all situations $\mathbf{x}$. This constraint will be called the reliability (*rel*) constraint. (2.7) can be rewritten as

$$\sum_{i,i_1,\ldots,i_m} \delta(i - i_k)\, \mathrm{P}(y = i, d(1) = i_1, \ldots, d(m) = i_m|\mathbf{x}) = \Delta(k) \tag{2.8}$$

2.4 Constraints Related to the Correlations between Experts

It is well known that experts opinions can be correlated. A possible choice for modeling experts correlations would be to provide

$$\sum_i \mathrm{P}(d(k) = i, d(l) = i|\mathbf{x}) = \sigma(k, l|\mathbf{x}) \text{ for } k, l = 1 \ldots m \tag{2.9}$$

It corresponds to the probability that expert k and expert l agree. If $\sigma(k, l|\mathbf{x}) = 1$, expert k and expert l always agree (they are totally correlated), while if $\sigma(k, l|\mathbf{x}) = 0$, they always disagree. If we only know the correlation without reference to the context $\mathbf{x}$, we must postulate independence with respect to the context, i.e.

$$\sum_i \mathrm{P}(d(k) = i, d(l) = i|\mathbf{x}) = \sigma(k, l) \text{ for } k, l = 1 \ldots m \tag{2.10}$$

This constraint will be called the correlation (*cor*) constraint. Once more, we can rewrite (2.10) as

$$\sum_{i,i_1,\ldots,i_m} \delta(i_k - i_l)\, \mathrm{P}(y = i, d(1) = i_1, \ldots, d(m) = i_m|\mathbf{x}) = \sigma(k, l) \tag{2.11}$$

We will now see how to compute the joint probability distribution satisfying the set of constraints (2.2), (2.3), (2.6), (2.9).

In the case of classifiers combination, the values of Δ and σ should be readily available based on statistics recorded on a training set or previous classification tasks.

3 The Maximum Entropy Approach

3.1 A Score of Aggregation for Expert's Opinions

As already stated, we would like to estimate the joint probability density

$$\mathrm{P}(y = i, d(1) = i_1, d(2) = i_2, \ldots, d(m) = i_m|\mathbf{x}) \tag{3.1}$$

satisfying the set of constraints (2.2), (2.3), (2.6), (2.9). The maximum entropy estimate of (3.1) will be denoted by

$$\widehat{\mathrm{P}}(y = i, d(1) = i_1, d(2) = i_2, \ldots, d(m) = i_m|\mathbf{x}) \tag{3.2}$$

with a hat. From this joint density, (3.2), we will compute the a posteriori probability of the true outcome $y = i$

$$\widehat{\mathrm{P}}(y = i|\mathbf{x}) = \sum_{i_1,\ldots,i_m} \widehat{\mathrm{P}}(y = i, d(1) = i_1, \ldots, d(m) = i_m|\mathbf{x}) \text{ for } i = 1 \ldots n \tag{3.3}$$

and this score will define our **score of aggregation** for expert's opinions. It represents the probability of outcome $y = i$ satisfying all the constraints provided by the experts and based on the estimated density that has maximum entropy. In equation (3.3), we average on all the possible situations that can appear in the context $\mathbf{x}$; that is, on all the different decisions of the experts, where each situation is weighted by its probability of appearance.

Notice that, in a different framework, Myung et al. [19] proposed to compute the a posteriori probability of y conditional on expert's probability of taking a given decision. This is, however, not well-defined since the experts provide a subjective probability density and not a clear decision: $\pi(d(k) = i_k|\mathbf{x})$ is not a random variable; the authors are therefore conditioning on a probability density and not an event.

If we define $\mathbf{d} = [d(1), d(2), \ldots, d(m)]^{\mathrm{T}}$ and $\mathbf{i} = [i_1, i_2, \ldots, i_m]^{\mathrm{T}}$, we can rewrite (3.3) in a more compact way as

$$\widehat{\mathrm{P}}(y = i|\mathbf{x}) = \sum_{\mathbf{i}} \widehat{\mathrm{P}}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) \text{ for } i = 1 \ldots n \tag{3.4}$$

We will now see how to compute these scores thanks to the maximum entropy principle.

3.2 The Maximum Entropy Estimate

Our aim is to estimate $\mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x})$ by seeking the probability density that has maximum entropy

$$I = -\sum_{i,\mathbf{i}} \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) \log \left[\mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x})\right] \tag{3.5}$$

among all the densities satisfying the constraints (2.2), (2.3), (2.6), (2.9). We therefore build the Lagrange function

$$\begin{aligned}
\mathcal{L} = & -\sum_{i,\mathbf{i}=1}^{n} \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) \log \left[\mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x})\right] \\
& + \lambda_{sum} \left[\sum_{i,\mathbf{i}=1}^{n} \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) - 1\right] \\
& + \sum_{k'=1}^{m} \sum_{i'_{k'}=1}^{n} \lambda_{op}(k', i'_{k'}) \left[\sum_{i,\mathbf{i}=1}^{n} \delta(i'_{k'} - i_{k'}) \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) - \pi(d(k') = i'_{k'})\right] \\
& + \sum_{k'=1}^{m} \lambda_{rel}(k') \left[\sum_{i,\mathbf{i}=1}^{n} \delta(i - i_{k'}) \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) - \Delta(k')\right] \\
& + \sum_{k'=1}^{m} \sum_{l'=1}^{m} \lambda_{cor}(k', l') \left[\sum_{i,\mathbf{i}=1}^{n} \delta(i_{k'} - i_{l'}) \mathrm{P}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) - \sigma(k', l')\right]
\end{aligned}$$

and compute the maximum with respect to the $\mathrm{P}(y = j, \mathbf{d} = \mathbf{j}|\mathbf{x})$. This problem has been studied extensively in the litterature. We show in [5] that $\widehat{\mathrm{P}}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x})$ takes the form

$$\begin{aligned}
\widehat{\mathrm{P}}(y = i, \mathbf{d} = \mathbf{i}|\mathbf{x}) = & G_{sum} \prod_{k=1}^{m} G_{op}(k, i_k) \\
& \times \prod_{k=1}^{m} \left[G_{rel}(k)\right]^{\delta(i - i_k)} \prod_{k=1}^{m} \prod_{l=1}^{m} \left[G_{cor}(k, l)\right]^{\delta(i_k - i_l)}
\end{aligned} \tag{3.6}$$

where the parameters G_{sum}, G_{op}, G_{rel}, G_{cor} can be estimated iteratively by an iterative scaling procedure (see next section).

3.3 Computing the Maximum Entropy Estimate

The first step is to verify that there is a feasible solution to the problem at hand. Since all the constraints are linear, a linear programming procedure can be used to solve this problem.

Once we have verified that there is indeed a feasible solution, an iterative scaling procedure allowing to estimate the G_{sum}, G_{op}, G_{rel}, G_{cor} can easily be derived (see for instance [4]). The iterative scaling procedure aims to satisfy in turn each constraint, and iterate on the set of constraints (as proposed by many authors, for instance [15]). It has been shown that this iterative procedure converges to the solution provided that there exists a feasible solution to the problem (that is, the set of constraints can be satisfied). Indeed, the entropy criterion is convex and the constraints are linear so that convex programming algorithms can be used in order to solve the problem.

3.4 The Case Where There Is No Feasible Solution

It can be the case that no solution satisfying the constraints (2.2), (2.3), (2.6), (2.9) exists. This means that there is a conflict between the different estimates π, Δ, σ, so that this situation cannot normally appear in reality. In that case, the user of the system should revise his different pieces of knowledge or data.

However, despite this conflicting situation, if the user nevertheless wants to compute an aggregated score, we have to relax in some way the set of constraints. There are two different ways of doing this: (1) by introducing slack variables that compute the lack of constraint satisfaction, or (2) to relax the equality constraints by providing intervals instead of exact values. These two approaches are introduced in the two next sections.

Introduction of Slack Variables. In this case, some equality constraints are relaxed. For instance, let us consider that we are willing to relax the reliability and correlation constraints for all experts. By introducing slack variables, ξ_k^{r+}, ξ_k^{r-}, ξ_{kl}^{c+}, ξ_{kl}^{c-}, measuring the lack of constraint satisfaction for each constraint, we have

$$\sum_i \mathrm{P}(y = i, d(k) = i|\mathbf{x}) + \xi_k^{r+} - \xi_k^{r-} = \Delta(k|\mathbf{x}) \text{ for } k = 1 \ldots m \tag{3.7}$$

$$\sum_i \mathrm{P}(d(k) = i, d(l) = i|\mathbf{x}) + \xi_{kl}^{c+} - \xi_{kl}^{c-} = \sigma(k, l|\mathbf{x}) \text{ for } k, l = 1 \ldots m \tag{3.8}$$

$$0 \leq \xi_k^{r+}, \xi_k^{r-}, \xi_{kl}^{c+}, \xi_{kl}^{c-} < \theta \tag{3.9}$$

where θ is a treshold provided by the user: the slack variables are not allowed to exceed this treshold. Consequently we want to minimize the lack of constraint satisfaction $\min \left[\sum_k (\xi_k^{r+} + \xi_k^{r-}) + \sum_{k,l} (\xi_{kl}^{c+} + \xi_{kl}^{c-}) \right]$ subject to constraints (2.2), (2.3), (3.7), (3.8), (3.9). This is a standard linear programming problem.

Introduction of Intervals. Another alternative would be to relax the equality constraints by providing intervals instead of exact values. Once again, let us consider that we are willing to relax the reliability and correlation constraints for

all experts. The problem would be reformulated as a maximum entropy problem with inequality constraints:

$$\Delta^{-}(k|\mathbf{x}) \leq \sum_{i} \mathrm{P}(y=i, d(k)=i|\mathbf{x}) \leq \Delta^{+}(k|\mathbf{x}) \text{ for } k=1\ldots m \tag{3.10}$$

$$\sigma^{-}(k,l|\mathbf{x}) \leq \sum_{i} \mathrm{P}(d(k)=i, d(l)=i|\mathbf{x}) \leq \sigma^{+}(k,l|\mathbf{x}) \text{ for } k,l=1\ldots m \tag{3.11}$$

Once more, numerical procedures related to iterative scaling can be used in order to compute the maximum entropy solution [4]. We have to maximize (3.5) subject to constraints (2.2), (2.3), (3.10), (3.11).

4 Simulation Results

For illustration purposes, we used the proposed combination rule in two different conditions. For each condition, we compute the following values:

Maximum entropy. If there is a feasible solution, we compute the *maximum entropy* solution to the problem.

Linear programming. If there is no feasible solution, we compute the *linear programming* solution to the problem (see (3.4)).

Weighted average. We also compute a *weighted average* solution, for comparison.

The weighted average is computed as follows. For each expert, we associate a weight, $w(k)$, which is a normalised (it sums to one) measure of his reliability and we assume that each reliability, $\Delta(k|\mathbf{x})$, is greater than 0.5 (the expert performs better than a random guess): $w(k) = (\Delta(k) - 0.5)/\sum_k(\Delta(k) - 0.5)$. The weighted average score is therefore $Score(i) = \sum_k w(k)\,\pi(d(k) = i|\mathbf{x})$.

Notice that we did not introduce correlation constraints in this set of simulations. For all simulations, we consider the case where there are three experts $\{1, 2, 3\}$ and two outcomes $\{0, 1\}$.

4.1 First Set of Simulations

We set, for experts' opinions, $\pi(d(1) = 0|\mathbf{x}) = 0.3$, $\pi(d(1) = 1|\mathbf{x}) = 0.7$, $\pi(d(2) = 0|\mathbf{x}) = 0.3$, $\pi(d(2) = 1|\mathbf{x}) = 0.7$, $\pi(d(3) = 0|\mathbf{x}) = 0.8$, $\pi(d(3) = 1|\mathbf{x}) = 0.2$. In other words, the two first experts agree, while the third one has an opposite opinion. For experts' reliability, we set, $\Delta(1) = 0.7$, $\Delta(2) = 0.7$, $\Delta(3) = z$, where $0.5 < z < 1$. The results are shown in Figure 1 (a), where we display $\widehat{\mathrm{P}}(y = 0|\mathbf{x})$ and $Score(0)$ in terms of the reliability of expert 3, i.e. z.

We observe that when the reliability of expert 3 is high, the fusion rules (that is, the experts combination rules) favour outcome 0. Notice that when $z > 0.8$, there is no feasible solution, that is, the constraints cannot be satisfied. Notice also that the maximum entropy solution is always in favour of outcome 0, in comparison with the weighted average ($\widehat{\mathrm{P}}(y = 0|\mathbf{x}) > Score(0)$).

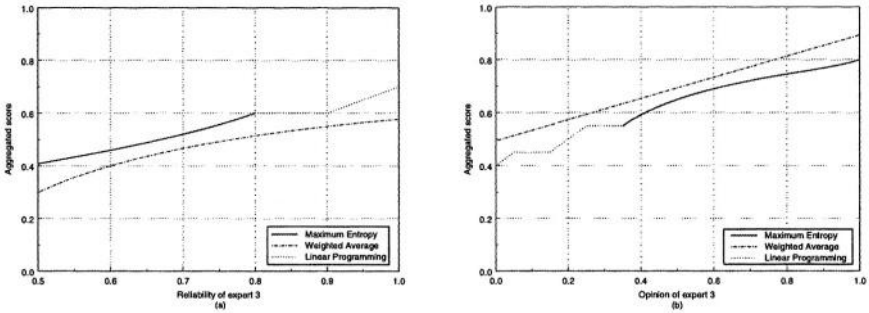


Fig. 1. Two examples of simulation of the fusion rule (see Sections 4.1 and 4.2 for details). We display the results of the three combination rules, maximum entropy, weighted average and linear programming in terms of (a) the reliability of expert 3 (b) expert 3's opinion.

4.2 Second Set of Simulations

In this second example, we set, for experts' opinions, $\pi(d(1) = 0|\mathbf{x}) = 0.85$, $\pi(d(1) = 1|\mathbf{x}) = 0.15$, $\pi(d(2) = 0|\mathbf{x}) = 0.8$, $\pi(d(2) = 1|\mathbf{x}) = 0.2$, $\pi(d(3) = 0|\mathbf{x}) = z$, $\pi(d(3) = 1|\mathbf{x}) = (1 - z)$ and, for the reliability, $\Delta(1) = 0.7$, $\Delta(2) = 0.75$, $\Delta(3) = 0.8$. The results are shown in Figure 1 (b), where we display $\widehat{\mathrm{P}}(y = 0|\mathbf{x})$ and $Score(0)$ in terms of the opinion of expert 3, i.e. z.

We observe that the weighted average rule is linear in the opinion of expert 3, while maxent is nonlinear. Below the value of $z = 0.35$, we see that the constraints are incompatible; in this case, the procedure described in 3.4 (linear programming) is used in order to compute the aggregation score.

5 Conclusion

We introduced a new way of combining experts opinions or classifiers outputs. It is based on the maximum entropy framework; maximum entropy seeks the joint probability density of a set of random variables that has maximum entropy while satisfying the constraints, i.e. it does not introduce any "additional ad hoc information". It is therefore the "most honest" characterization of our knowledge given the available facts. The available knowledge is expressed through a set of linear constraints on the joint density including a measure of reliability of the experts, and of the correlation between them. This way, the different constraints (representing the a priori knowledge about the problem) are incorporated within a single measure.

Iterative mathematical programming methods are used in order to compute the maximum entropy, with guaranteed convergence to the global maximum if, of course, there is a feasible solution. If there is a conflict between the available facts, i.e. between the constraints, so that there is no feasible solution, we could still compute the solution that is "closest" in some way to the constraints satisfaction.

However, even if this approach seems promising, it has not been evaluated in the context of classifiers combination. Further work will thus be devoted to the experimental comparison with more standard techniques such as those that were mentioned in the introduction. We therefore cannot currently provide any conclusion about the applicability (and performances) of this approach.

References

1. D. Chen and X. Cheng. An asymptotic analysis of some expert fusion methods. *Pattern Recognition Letters,* 22:901–904, 2001.
2. R. M. Cooke. *Experts in uncertainty.* Oxford University Press, 1991.
3. D. Dubois, M. Grabisch, H. Prade, and P. Smets. Assessing the value of a candidate: Comparing belief function and possibility theories. *Proceedings of the Fifteenth international conference on Uncertainty in Artificial Intelligence,* pages 170–177, 1999.
4. S.-C. Fang, J. Rajasekera, and H.-S. J. Tsao. *Entropy optimization and mathematical programming.* Kluwer Academic Publishers, 1997.
5. F. Fouss and M. Saerens. A maximum entropy approach to multiple classifiers combination. *Technical Report, IAG, Universite catholique de Louvain,* 2004.
6. C. Genest and J. V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science,* 36:114–148, 1986.
7. A. Golan, G. Judge, and D. Miller. *Maximum entropy econometrics: Robust estimation with limited data.* John Wiley and Sons, 1996.
8. R. A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation,* 7:867–888, 1995.
9. F. Jelinek. *Statistical methods for speech recognition.* The MIT Press, 1997.
10. J. N. Kapur and H. K. Kesavan. *The generalized maximum entropy principle (with applications).* Sandford Educational Press, 1987.
11. J. N. Kapur and H. K. Kesavan. *Entropy optimization principles with applications.* Academic Press, 1992.
12. J. Kittler and F. Alkoot. Sum versus vote fusion in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25(1):110–115, 2003.
13. J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(3):226–239, 1998.
14. G. J. Klir and T. A. Folger. *Fuzzy sets, uncertainty, and information.* Prentice-Hall, 1988.
15. H. Ku and S. Kullback. Approximating discrete probability distributions. *IEEE Transactions on Information Theory,* 15(4):444–447, 1969.
16. F. Lad. *Operational subjective statistical methods.* John Wiley and Sons, 1996.
17. W. B. Levy and H. Delic. Maximum entropy aggregation of individual opinions. *IEEE Transactions on Systems, Man and Cybernetics,* 24(4):606–613, 1994.
18. C. Merz. Using correspondence analysis to combine classifiers. *Machine Learning,* 36:226–239, 1999.
19. I. J. Myung, S. Ramamoorti, and J. Andrew D. Bailey. Maximum entropy aggregation of expert predictions. *Management Science,* 42(10):1420–1436, 1996.

Combining One-Class Classifiers to Classify Missing Data

Piotr Juszczak and Robert P.W. Duin

Information and Communication Theory Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, The Netherlands
{p.juszczak,r.p.w.duin}@ewi.tudelft.nl

Abstract. In the paper a new method for handling with missing features values in classification is presented. The presented idea is to form an ensemble of one-class classifiers trained on each feature, preselected group of features or to compute from features a dissimilarity representation. Thus when any feature values are missing for a data point to be labeled, the ensemble can still make a reasonable decision based on the remaining classifiers. With the comparison to standard algorithms that handle with the missing features problem it is possible to build an ensemble that can classify test objects with all possible occurrence of missing features without retrain a classifier for each combination of missing features. Additionally, to train such an ensemble a training set does not need to be uncorrupted. The performance of the proposed ensemble is compared with standard methods use with missing features values problem on several UCI datasets.

1 Introduction

The increasing resolution of the sensors increases also the probability that one or a group of features can be missing or strongly contaminated by noise. Data may contain missing features due to a number of reasons e.g. data collection procedure may be imperfect, a sensor gathering information may be distorted by unmeasurable effects yielding the loss of data. Several ways of dealing with missing feature values have been proposed. The most simple and straightforward is to ignore all missing features and use the remaining observations to design a classifier [1]. Other group of methods estimates values of the missing features from available data by: replacing missing feature values by e.g. their means estimated on a training set [2], [3]. Morin [4] proposed to replace missing feature values by values of these features from their nearest neighbors, in the available, lower dimensional space, from the training set. [5, 4, 1] described different solutions using the linear regression to estimate substitutes for missing features values. However, Little [5] showed that many such methods are inconsistent, i.e. discriminant functions designed form the completed training set do not converge to the optimal limit discriminant functions as sample size tends to infinity. At this moment, methods recommended as generally best are based on EM algorithm [6,1].

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 92–101, 2004.

However, because of the complexity of existing methods, neither of them can provides a solution for all cases with missing features, that can occur during classification. When an corrupted test point occurs a classifier is retrained or missing features is replaced by estimated ones which can lead to worse results than when the classification decision is based just on existing features [5, 1]. Additionally, in most of the proposed solutions to the missing feature problem it is assumed that training data is uncorrupted, thus potentially valuable data are neglected during training.

In this paper several techniques based on combining one-class classifiers [7] are introduced to handle missing feature. The classifiers are trained on one-dimensional problems, n-dimensional problem or features are combined in dissimilarity representations. The presented method can coupe with all possible situation of missing data, from single one to $N - 1$, without retraining a classifier. It also makes use of corrupted data available for training.

The layout of this paper is as follows: in section 2, the problem of missing feature values and combining one-class classifiers (*occs*) is addressed. In section 2.1 some possibility of combining one-class classifiers to handle missing features problem are discussed. Section 3 shows results on UCI datasets and discusses the relative merits and disadvantages of combining *occs*. Section 4 presents the discussion and conclusions.

2 Formal Framework

Suppose two sets of data are given: a training set

$$\mathcal{L} = \{(\mathbf{x}_m, \mathbf{y}_m) \ : \ \mathbf{x}_m \in \mathbb{R}^{\mathbf{p}_m}; \quad m = 1 \ldots M\},$$

and a test set

$$\mathcal{T} = \{\mathbf{x}_t \ : \ \mathbf{x}_t \in \mathbb{R}^{\mathbf{q}_t}; \quad t = 1 \ldots T\}, \qquad \text{where} \quad (\mathbf{p}_m, \mathbf{q}_t) \in \mathbb{R}^N.$$

Where **x**-s represent objects and **y**-s represent labels[1]. N is a number of all the features considered in a classification task. Each object $\mathbf{x}$ in $\mathcal{L}$ or $\mathcal{T}$ can reside in the different space. Even if x_{m_1} and x_{m_2} are represented in spaces of the same dimensionality $\|p_{m_1}\| = \|p_{m_2}\|$, the present features might be different $p_{m_1} \neq p_{m_2}$. Such a problem is called a classification with missing data in training and test sets.

Suppose a classifier is designed by using uncorrupted data. Assume that input (test) data are then corrupted in particularly known ways. How to classify such corrupted inputs to obtain the minimal error? For example, consider a classifier for data with two features, such that one of the features is missing for a particular object x to be classified. Fig. 1 illustrates a three-class problem, where for the test object x the feature f_1 is missing. The measured value of f_2 for x is x_{f_2}. Clearly, if we assume that the missing value can be substituted by the $mean(f_1)$,

[1] It is assumed that both the training set $\mathcal{L}$ and the test set $\mathcal{T}$ are corrupted.

x will be classified as y_2. However, if the priors are equal, y_3 would be a better decision, because $p(x_{f_1}|y_3)$, estimated on the training set is the largest of the three likelihoods. In terms of a set of existing features $\mathbf{F}$, the posteriors are [8]:

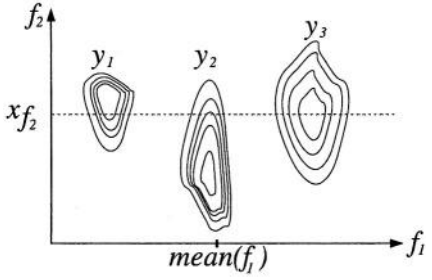


Fig. 1. Class conditional distributions for a three-class problem. If a test point misses the feature value from f_1 the optimal classification decision will be y_3 because $p(x_{f_1}|y_3)$ (estimated on the training set) is the largest.

$$P(y_i|\mathbf{F}) = \frac{\int g_i(\mathbf{F})\, p(\mathbf{F})\, d\mathbf{f}_-}{\int p(\mathbf{F})\, d\mathbf{f}_-} \qquad (1)$$

where $\mathbf{f}_-$ indicates the missing features, $g_i(\mathbf{F}) = P(y_i|\mathbf{F}, \mathbf{f}_-)$ is the conditional probability from a classifier. In short, equation (1) presents integrated, marginalization of the posterior probability over the missing features.

Several attempts were made to estimate missing feature values for a test object [1] e.g. by:
- solving a classification problem in an available lower-dimensional feature space $\mathbf{F}$ (obviously in this case no estimation of the missing data is required);
- replacing missing feature values in $\mathcal{T}$ by the means of known values from $\mathcal{L}$;
- replacing missing values in $\mathcal{T}$ by values from the nearest neighbor from $\mathcal{L}$;
- using the expectation-maximization algorithm to maximize e.g. the class posteriors. This method is the most complicated one and the assumption about underlaying data distribution has to be made.

In further experiments the first and the third methods mentioned above are used as a comparison to the proposed methods.

2.1 Combining One-Class Classifiers

In the problem of one-class classification the goal is to accurately describe one class of objects, called the target class, as opposed to a wide range of other objects which are not of interest, called outliers. Many standard pattern recognition methods are not well equipped to handle this type of problem; they require complete descriptions for both classes. Especially when one class is very diverse and ill-sampled, usually (two-class) classifiers yield a very bad generalization for this class. Various methods have been developed to make such a data description [7]. In most cases, the probability density of the target set is modeled. This requires a large number of samples to overcome the curse of dimensionality [9].

Since during a training stage it is assumed that only target objects maybe present, a threshold is set on tails of the estimated probability or distance d such that a specified amount of the target data is rejected, e.g. 0.1. Then in the test stage, the estimated distances d can be transformed to resemble posterior probabilities as follow $p(y|x) = \frac{1}{1+e^{-d}}$ for the target class and $1 - p(y|x)$ for the outlier class.

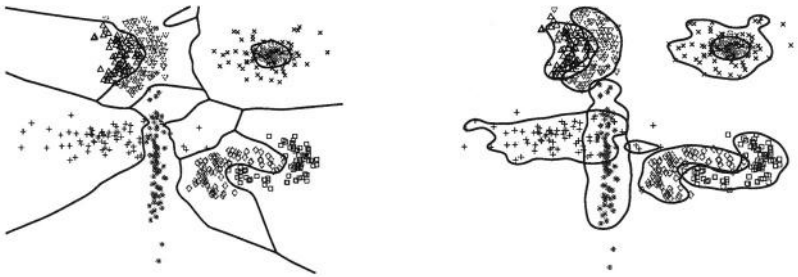


Fig. 2. A multi-class problem solved by: (left) combining two-class classifiers (one-vs-all approach), (right) combining one-class classifiers by the maximum resemblance to a model.

Fig. 2 illustrates differences between the solution to multi-class problem by combining two-class classifiers (one-vs-all approach) [9] and combining one class classifiers [7]. In the first approach, the entire data space is divided into parts being assigned to a particular class. A new object x **has to** be classified to one of the classes present in the training set. It means in a case of outliers the classification is ironies. In addition in one-vs-all or pairwise combining approach one has to compensate imbalance problem by e.g. settings probabilities to appropriate levels.

The right in Fig. 2 plot shows the *occs* combined by max rule. This means that in order to handle a multi-class problem, *occs* can be combined by the max rule or by a train combiner. In this approach, one assigns a new data point only to the particular class if it is in one of the described domains. If a new object x lies outside a region described by the target class, it is assigned to the outlier class. In the combination of two-class classifiers it appears that often the more robust mean combination rule is to be preferred. Here extreme posterior probability estimates are averaged out. In one-class classification only the target class is modeled $P(\mathbf{x}|\omega_{T_c})$ and a low uniform distribution is assumed for outlier class. This makes this classification problem asymmetric and extreme target class estimates are not canceled by extreme outlier estimates. However, the mean combination covers a broad domain in feature space [10], while the product rule has restricted range. Especially in high dimensional spaces this extra area will cover a large volume and potentially a large number of outliers.

2.2 Proposed Method

In this paper we propose several methods based on combing one-class classifiers to handle the missing features problem. Our goal is to build such an ensemble that dose not required retraining of a classifier for every combination of missing data and at the same time minimizes number of classifiers that has to be considered. In this section we will describe several ways of combining *occs* and some possibilities for the based classifiers.

First, two-class classifiers, combined like in one-vs-all method are considered; Fig. 2 (left) trained on all possible combination of missing feature values. In such case the number of base two-class classifiers that has to be trained is $K_{C,N} = (2^N - 1) \cdot \frac{C \cdot (C-1)}{2}$, where N is the number of features and C is the number of classes. Since all the features cannot be missing 1 is subtracted from all 2^N possibilities. For a problem with ten features and two classes, $K_{2,10} = 1023$ and for 20 features, $K_{2,20} = 1048575$. For such simple problems the number of classifier is already quite large.

On the other hand, if one-class classifiers are trained on all possible combination of missing features than the number of possibilities reduces to $K_{C,N} = (2^N - 1) \cdot C$ and the classification regions do not longer are considered as open spaces, Fig. 2 (right). However, for a large number of features this is a quite complicated study, since the number of classifiers is still cumbersome to handle and the system is difficult to validate.

In this paper, one of the proposed methods is to use one-class classifiers as base classifiers to combine, trained on one-dimensional problems and combine by fix combining rules: mean, product, max etc.. This reduces the number of classifiers that has to be in the pool as a combining possibilities to $N \cdot C$ for the fixed combining rules $K_{2,20} = 40$.

Below the way how to use fix (mean, product, and max) combining rules applied to the missing feature values problem in multi-class problems are described.

Mean combining rule: $y(\mathbf{x}|\omega_T) = \arg\max_c \left[\sum_{i=1}^{N'} P(x_i|\omega_{T_c}) \right]$

Product combining rule: $y(\mathbf{x}|\omega_T) = \arg\max_c \left[\prod_{i=1}^{N'} P(x_i|\omega_{T_c}) \right]$

Max combining rule: $y(\mathbf{x}|\omega_T) = \arg\max_c \left[\max_i P(x_i|\omega_{T_c}) \right]$

where $P(x_i|\omega_{T_c})$ is a probability that object x belongs to the target class C and N' is the number of available features. The probabilities $P(x_i|\omega_{T_c})$ estimated on single features are combined by fix rules. The test object $\mathbf{x}$ is classified to the class C with the maximum resemblance to it. However, because a single feature x_i is considered at time during classification the feature interactions are neglected. This can lower the performance of the proposed ensemble. This problem will be addressed in the section 3.2 of this paper.

Combining Dissimilarity Based Representations. The second method that is proposed in this paper, to handle missing features, is to combine non-missing features in the dissimilarity measure [11]. In this case, instead of training a classifier in the N' dimensional feature space it is trained in a dissimilarity space. In our experiments the sigmoid transformation from distances to dissimilarities is used:

$$dd_{jk} = \frac{1}{N'} \sum_{i=1}^{N'} \left[\frac{2}{1 + \exp(-\frac{d_{jk_i}}{\sigma_i})} - 1 \right] \qquad \text{where} \quad \sigma_i = \frac{\mathbf{d}_i}{N} \qquad (2)$$

where dd_{jk} is the computed dissimilarity between object j and k and d_{jk_i} is the Euclidean distance between those two objects. To increase the robustness in the case of missing features dissimilarities were averaged out over one-dimensional representations. σ_i is approximated by the average distance between training objects considered the single feature i at time.

3 Experiments

In the experiments as the base classifier a simple Parzen classifier was used with the threshold set to reject 0.05 of target objects in the training set [7]. The smoothing parameter was optimized by leave-one-out approach [12]. Linear Programming Dissimilarity-data Description (LPDD) [13] (dist) was used as the base classifier for combining dissimilarity representations with 0.05 of the target objects rejection rate. To combine classifiers resemblances to the target class $y(x|\omega_T)$ were transformed to posterior probabilities by the sigmoid transformation $\frac{1}{1+\exp(-y(x|\omega_T))}$. The fixed (mean, product and max) [14] combining rules are applied to posterior probabilities computed from the resemblance on single features.

The proposed methods were compared to two standard methods designed to handle missing data: training classifier in a lower, available feature space (lower) and replacing missing features in a test object from $\mathcal{T}$ by the features of their nearest neighbor from a training set $\mathcal{L}$ (f_{nn}). The experiments were carried out on some of UCI datasets [15]: WBCD - Wisconsin Breast Cancer Dataset (number of classes c = 2, number of features k = 9), MFEAT (c=10, k=649), CBANDS (c=24, k=30), DERMATOLOGY (c=6, k=34), SATELLITE (c=6, k=36). The total number of features in MFEAT dataset was reduced from the original number of 649 to 100 (MFEAT 100) and 10 (MFEAT 10) by a forward feature selection based on maximization of the Fisher criterion: the trace of ratio of the within- and between-scatter matrices $J = tr\{S_W^{-1} S_B\}$ [16], to avoid the curse of dimensionality. The datasets were spited randomly into the equally sized training and test sets. For each percent of missing features ([0:10:90]%) ten training sets and for each training set ten test sets were randomly generated 10×10.

3.1 Combining *occs* Trained on Single Features

In this section the ensemble built from classifiers trained on individual features are evaluated. It is assumed that each feature contributes similar, independent amount of information to the classification problem. Any interactions between features are neglected.

In Fig. 3 mean errors for different solution to the missing features problem for different multi-class problem are presented. The classifiers are trained on one-dimensional problems and combined by fix combining rules. In dism method corespondent dissimilarities are computed and LPDD is trained on all one-class classification problems. The results are compared with two standard methods

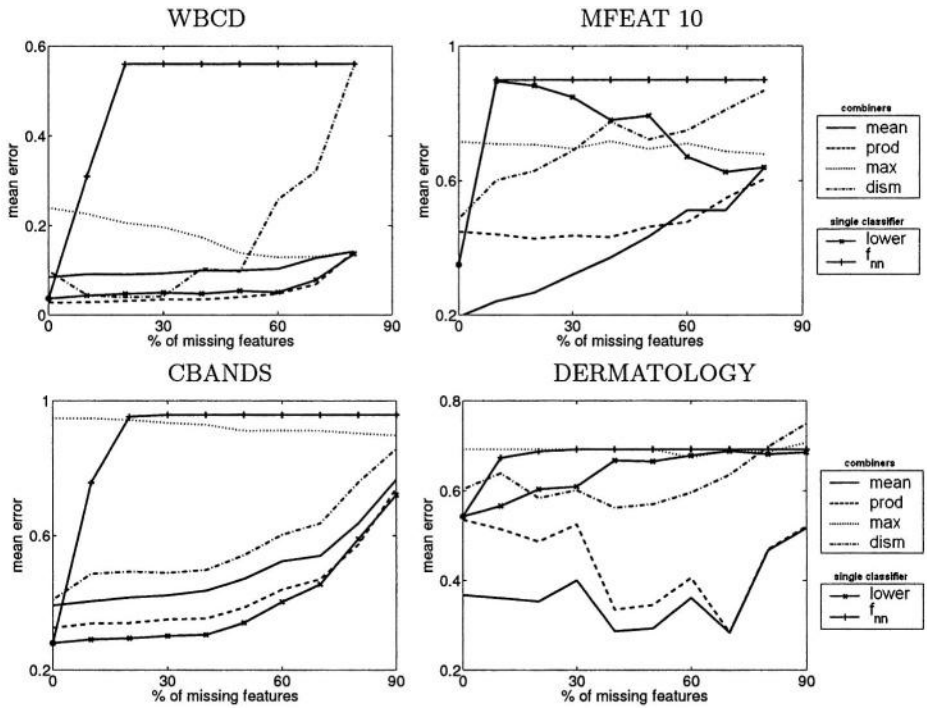


Fig. 3. Mean error for different percent of missing features for the combiners trained on single features for various combining rules: (mean, product, max), dism - dissimilarity representation LPDD. lower - the Parzen classifier trained on all available features. f_{nn} - the Parzen classifier trained on available features plus features from nearest neighbor from a training set. The results are averaged over 10×10 times; see text for details.

for missing features problem: lower - a classifier is trained on all available features neglecting missing features and f_{nn} missing feature values are replaced by features from the nearest neighbor of the test object in the training set. It can be observed that mean and product rule are performing the best for the entire range of missing features. It depends on the dataset which of this fix combining rules is better. The dissimilarity representation does not perform well, apart from WBCD, for which for a small percent of missing features the performance is comparable with fix combiners. The reason is that the computed dissimilarities on the training set, on all the features, are not resemble to dissimilarities computed on the test set with missing features. However, the dism method outperforms the standard f_{nn} method. The reasons for such poor performance of the f_{nn} method is that if more features are missing replacing them by features from the training set will cause less differences between test objects. The single classifier trained on all available features performs the best on the CBANDS dataset however is outperformed in other problems by fix combining rules. It can be concluded that more complicated problems split in simple ones and then combine can outperform a single, big classifier [17,18].

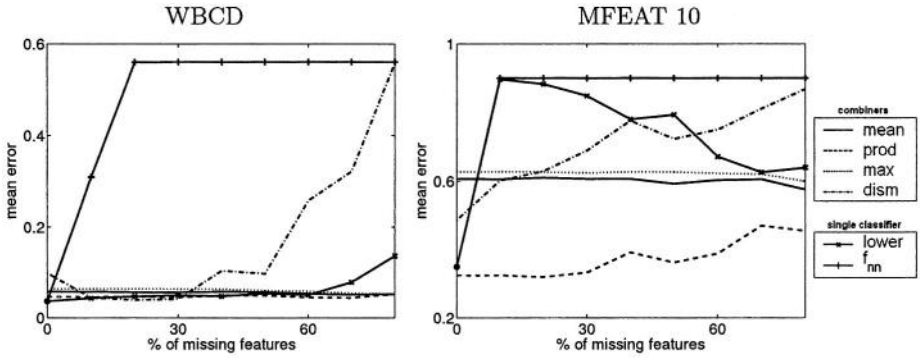


Fig. 4. Mean error for different percent of missing features for the combiners trained on (n+1) features for various combining rules: (mean, product, max), dism - dissimilarity representation LPDD. lower - the Parzen classifier trained on all available features. f_{nn} - the Parzen classifier trained on available features plus features from nearest neighbor from a training set. The results are averaged over 10×10 times; see text for details.

3.2 Combining *occs* Trained on $(n+1)$ Features

In the previous section it was assumed that every feature contributes a similar, independent amount of information to the classification problem. In this section we will study a possibility when a fixed number of features is always present or when there is a certain subset of features without which the classification is almost random e.g. for medical data like: name, age, height,..., examinations. It is probably possible to classify a patient to the healthy/unhealthy group without a name or age provided, but not without specific examination measurements. One of the possible solutions is to use a weighted combining rule [19].

In this paper, a different approach is proposed. Let us assume that the same n features are always present for the test objects. Therefore, instead of N possible missing features we have $N-n$ possibilities. In this case, we propose to train $(N-n-1)$ base one-class classifiers in a $(n+1)$-dimensional space. As a result, the base classifiers are highly depend. According to common knowledge on combining classifiers [14, 20], combining is beneficial when base classifiers differ. However, in our case, there is a trade-off between the number n of common features and how well the posterior probabilities are estimated. In Fig. 4, the mean error for $n=3$ for WBCD and MFEAT 10 is shown. The standard deviation varies between 1-2% from the mean value. The classifiers are trained on (n+1) features and then combined. Compared to the results showed in Fig. 3 the performance of fix combiners increases. The posterior probabilities are better estimated and some features dependencies are also included in the estimation. If an additional knowledge is available about a classification problem e.g. n features are always present by appropriate combining better classification performance can be achieved.

3.3 Small Sample Size Problem

In this section the performance of the proposed method are evaluated for small sample size problems [9]. In small sample size problems the number of objects per class is similar or smaller than the number of features.

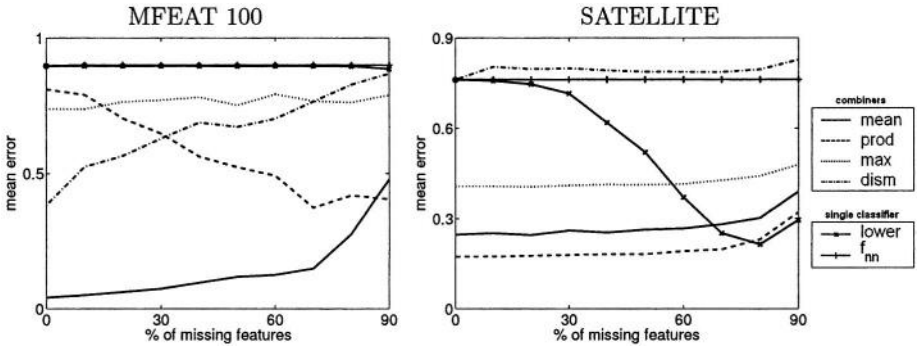


Fig. 5. Small sample size problems. Mean error for different percent of missing features for the combiners trained on single features for various combining rules: (mean, product, max), dism - dissimilarity representation LPDD. lower - the Parzen classifier trained on all available features. f_{nn} - the Parzen classifier trained on available features plus features from nearest neighbor from a training set. The results are averaged over 10×10 times; see text for details.

Fig. 5 shows the mean error for two small sample size problems. Because the probabilities are estimated on single feature the proposed method is robust to small sample size problems. The classifier statistics are better estimated and the constructed ensemble is robust against noise.

4 Conclusions

In this paper, several methods for handling missing feature values have been proposed. The presented methods are based on combining one-class classifiers trained on one-dimensional or (n+1) dimensional problems. Additionally, the dissimilarity based method is proposed to handle the missing features problem. Compared to the standard methods, our methods are much more flexible, since they require much less classifiers to consider and do not require to retrain the system for each new situation when missing feature values occur. Additionally, our method is robust to small sample size problems due to splitting the classification problem to N several smaller ones.

Acknowledgments

Authors thank Elżbieta Pękalska for helpful suggestions in preparation the final version of the paper. This work was partly supported by the Dutch Organization for Scientific Research (NWO).

References

1. Little, R.J.A., Rubin, D.B.: Statistical analysis with missing data. 2 edn. ISBN 0-471-18386-5. Wiley-Interscience (2002)
2. Chan, L.S., Dun, O.J.: Alternative approaches to missing values in discriminant analysis. J. Amer. Statist. Assoc. **71** (1976) 842–844
3. Dixon, J.K.: Pattern recognition with partly missing data. IEEE Transactions on Sys., Man and Cyber. (1979) 617–621
4. Morin, R.L., Raeside, D.E.: A reappraisal of distance-weighted k-nearest neighbor classification for pattern recognition with missing data. IEEE Trans. Syst. Man Cybern. **11** (1981) 241–243
5. Little, R.J.A.: Consistent regression methods for discriminant analysis with incomplete data. J. Amer. Statist. Assoc. **73** (1978) 319–322
6. Ghahramani, Z., Jordan, M.I.: Supervised learning from incomplete data via an em approach. In: NIPS. (1994)
7. Tax, D.M.J.: One-class classification. PhD thesis, Delft University of Technology (2001)
8. Ahmad, S., Tresp, V.: Some solutions to the missing feature problem in vision. In: NIPS. (1993) 393–400
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. 2nd edn. ISBN: 0-471-05669-3. Wiley Interscience (2001)
10. Tax, D.M.J., Duin, R.P.W.: Combining one-class classifiers. In: MCS. (2001) 299–308
11. Pekalska, E., Duin, R.P.W.: Dissimilarity representations allow for building good classifiers. PR Letters **23** (2002) 943–956
12. Duin, R.P.W.: On the choice of the smoothing parameters for parzen estimators of probability density functions. IEEE Transactions on Computers (1976)
13. Pekalska, E., Tax, D.M.J., Duin, R.P.W.: One-class lp classifiers for dissimilarity representations. In: NIPS. (2002) 761–768
14. Kittler, J., Hatef, M., Duin, R.P.W.: On combining classifiers. IEEE Transactions on PAMI **20** (1998)
15. Blake, C.L., Merz, C.J.: (UCI repository of machine learning databases)
16. Kittler, J.: Feature selection and extraction. Handbook of Pattern Recognition and Image Processing (1996) 59–83
17. Ho, T.K.: Data complexity analysis for classifier combination. In: MCS. (2001) 53–67
18. Raudys, S.: Multiple classification systems in the context of feature extraction and selection. In: MCS. (2002) 27–41
19. Littlestone, N., Warmuth, M.: Weighted majority algorithm. Information and Computation **108** (1994) 212–261
20. Duin, R.P.W.: The combining classifier: to train or not to train. In: ICPR. (2002)

Combining Kernel Information for Support Vector Classification

Isaac Martín de Diego[1], Javier M. Moguerza[2], and Alberto Muñoz[1]

[1] University Carlos III de Madrid, c/ Madrid 126, 28903 Getafe, Spain
{ismdiego,albmun}@est-econ.uc3m.es
[2] University Rey Juan Carlos, c/ Tulipán s/n, 28933 Móstoles, Spain
j.moguerza@escet.urjc.es

Abstract. In this paper we describe new methods to built a kernel matrix from a collection of kernels for classification purposes using Support Vector Machines (SVMs). The methods build the combination by quantifying, relative to the classification labels, the difference of information among the kernels. The proposed techniques have been successfully evaluated on a variety of artificial and real data sets.

1 Introduction

Support Vector Machines (SVMs) have proven to be a successful tool for the solution of a wide range of classification problems since their introduction in [3]. The method uses as a primary source of information a kernel matrix $K(i,j)$, where K is Mercer's kernel and i,j represent data points in the sample. By the representer theorem (see for instance [16]), SVM classifiers always take the form $f(x) = \sum_i \alpha_i K(x,i)$. The approximation and generalization capacity of the SVM is determined by the choice of the kernel K [4]. A common way to obtain SVM kernels is to consider a linear differential operator D, and choose K as Green's function for the operator D^*D, where D^* is the adjoint operator of D [15]. It is easy to show that $\|f\|^2 = \|Df\|^2_{L_2}$ [9]. Thus we are imposing smoothing conditions on the solution f. However, it is hard to know in advance which particular smoothing conditions to impose for a given data set. Fortunately, kernels are straightforwardly related to similarity (or equivalently distance) measures, and this information is actually available in many data analysis problems. In addition, working with kernels avoids the need to explicitely work with Euclidean coordinates. This is particularly useful for data sets involving strings, trees, microarrays or text data sets, for instance.

Nevertheless, using a single kernel may be not enough to solve accurately the problem under consideration. This happens, for instance, when dealing with text mining problems, where analysis results may vary depending on the document similarity measure chosen [8]. Thus, information provided by a single similarity measure (kernel) may be not enough for classification purposes, and the combination of kernels appears as an interesting alternative to the choice of the 'best' kernel.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 102–111, 2004.

The specific literature on the combination of kernels is rather in its beginnings. A natural approach is to consider linear combinations of kernels. This is the approach followed in [10], and is based on the solution of a semi-definite programming problem to calculate the coefficients of the linear combination. The solution of this kind of optimization problems is computationally very expensive [19] and, therefore, out of the scope of this paper. In addition, there is no reason why a linear combination should provide optimal results. Another approach is proposed in [2]. The method, called MARK, builds a classifier (not the specific kernel matrix) by a boosting type algorithm. Classification results shown in that work are very similar to those obtained with a SVM classifier.

In this paper we describe three methods to build a kernel matrix from a collection of kernels for classification purposes. We provide a general scheme for combining the available kernels. Our methods build the combination by quantifying, relative to the classification labels, the difference of information among the kernels.

The paper is organized as follows. Section 2 describes the proposed methods for combining kernels. The experimental setup and results on artificial and real data sets are described in Section 3. Section 4 concludes.

2 Methods

Let $K_1, K_2, \cdots, K_m$ be a set of m input kernels defined on a data set, and denote by K^* the desired output combination. We first concentrate on binary classification problems ($m = 2$) and then we will extend the method to the case $m > 2$. Let y denote the label vector, where $y_i \in \{-1, +1\}$.

To motivate the discussion, consider the average of the kernels involved:

$$K^* = \frac{1}{2}(K_1 + K_2)\,. \tag{1}$$

There is a straightforward similarity between (1) and the first term of the standard decomposition of a matrix into its symmetric and skew-symmetric parts: $K^* = \frac{1}{2}(K^* + K^{*T}) + \frac{1}{2}(K^* - K^{*T})$. If K^* is a symmetric matrix, then $K^* = K^{*T}$ and the skew-symmetric part will vanish. By analogy we will derive kernel combinations of the form

$$K^* = \frac{1}{2}(K_1 + K_2) + f(K_1 - K_2)\,, \tag{2}$$

such that if K_1 and K_2 tend to produce the same classification results, then $f(K_1 - K_2)$ becomes meaningless and (1) yields $K^* \simeq K_1 \simeq K_2$. The term $f(K_1 - K_2)$ will quantify the difference of information between kernels K_1 and K_2 for the aim of classification.

2.1 The Absolute Value (AV) Method

Consider the matrix $Y = diag(y)$, whose diagonal entries are the y_i labels. Our first proposal builds K^* through the formula:

$$K^* = \frac{1}{2}(K_1 + K_2) + \tau Y|K_1 - K_2|Y\,, \tag{3}$$

where τ is a positive constant to control the weight given to the term $f(K_1 - K_2)$. Each element $K^*(i,j)$ takes the form

$$K^*(i,j) = \frac{1}{2}(K_1(i,j) + K_2(i,j)) + \tau y_i y_j |K_1(i,j) - K_2(i,j)|\,. \tag{4}$$

The intuition underlying the method is next justified. Consider two data points i and j in the same class ($y_i y_j = 1$). Taking into account that $\max(x,y) = 1/2(x+y) + 1/2|x-y|$, it is direct to show that, for $\tau = 1/2$, $K^*(i,j) = \max(K_1(i,j), K_2(i,j))$. Analogously, from $\min(x,y) = 1/2(x+y) - 1/2|x-y|$, if i and j belong to different classes ($y_i y_j = -1$), then $K^*(i,j) = \min(K_1(i,j), K_2(i,j))$. Given that K^* can be interpreted as a similarity measure, if i and j are in the same class, the method guarantees that $K^*(i,j)$ will be as large as possible. On the other hand, if i and j belong to different classes, a low similarity between them can be expected. Hence, the method tends to move closer points belonging to the same class, and tends to separate points belonging to different classes. In the case of an asymmetric classification problem, this method reduces to the pick-out method, presented in [13].

Notice that positive definiteness of K^* is not guaranteed. Several solutions have been proposed to face this problem [14]: A first possibility is to replace K^* by $K^* + \lambda I$, for $\lambda > 0$ large enough to make all the eigenvalues of the kernel matrix positive. Another direct approach uses Multidimensional Scaling to represent the data set in an Euclidean space [7]. Finally it is also possible to define a new kernel matrix as $K^{*T}K^*$ [17]. In practice. there seems not to be a universally best method to solve this problem.

2.2 The Squared Quantity (SQ) Method

Our second proposal builds each element of K^* as:

$$K^*(i,j) = \frac{1}{2}(K_1(i,j) + K_2(i,j)) + \tau y_i y_j (K_1(i,j) - K_2(i,j))^2\,, \tag{5}$$

where τ plays the same role as in the AV method. Figure 1 shows different choices of $f(K_1 - K_2)$ for different values of τ. The straight lines correspond to the AV method. Notice that this is the upper limiting case of the SQ method. The lower limiting case is represented by the x-axis line, corresponding to the use of $\frac{1}{2}(K_1 + K_2)$. In addition, the SQ curves are differentiable everywhere.

As in the previous case, positive semidefiniteness is not assured and the same comments apply.

2.3 The Squared Matrix (SM) Method

This method provides a straightforward solution to the lack of positive semidefiniteness. While the SQ method works by squaring each element of the $K_1 - K_2$

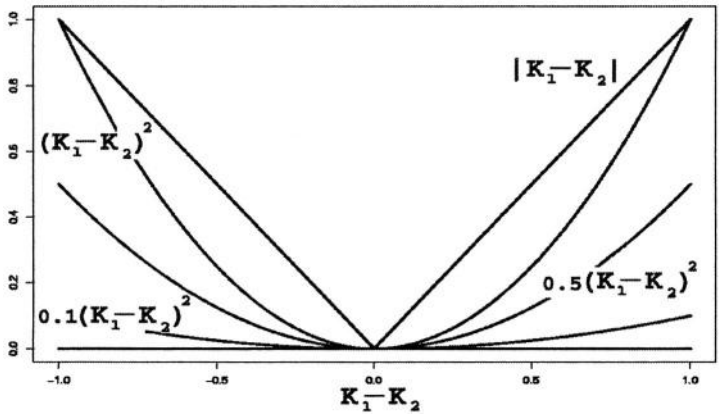


Fig. 1. Different choices of $f(K_1 - K_2)$ for different values of τ. The straight lines correspond to the AV method. The curves correspond to the SQ method.

matrix, the SM method works by considering the continuous operator defined as the square of the whole matrix, that is, $(K_1 - K_2)(K_1 - K_2)$. The combination formula now becomes:

$$K^* = \frac{1}{2}(K_1 + K_2) + \tau(Y(K_1 - K_2))^2, \tag{6}$$

where τ plays the same role as in the two previous methods. Given that matrices $Y(K_1 - K_2)(K_1 - K_2)Y$ and $(K_1 - K_2)^T(K_1 - K_2)$ have the same eigenvalues, the matrix built by the SM method is positive semidefinite. Hence, this method provides a matrix K^* arising from a Mercer's kernel.

2.4 Combining More than Two Kernels

Proceeding in a recursive way, the extension of the methods to the combination of more than two kernels is straightforward. For the sake of simplicity, consider the case $m = 3$. Let K_1, K_2 and K_3 be the considered kernel matrices, and K_{12}^* the resulting kernel obtained by using one of the preceding methods with K_1 and K_2 as input matrices. The final solution K^* will be obtained via the combination of K_{12}^* and K_3.

The order in which kernels are given to the recursive procedure is a current research issue. For instance, the order induced by the classification capacity of each kernel could be considered. In this paper, the classification capacity has been measured in terms of the classification success rates of each kernel.

3 Experiments

Next we test the performance of the previous methods on both artificial and real data sets. To check the performance of the methods described above, once

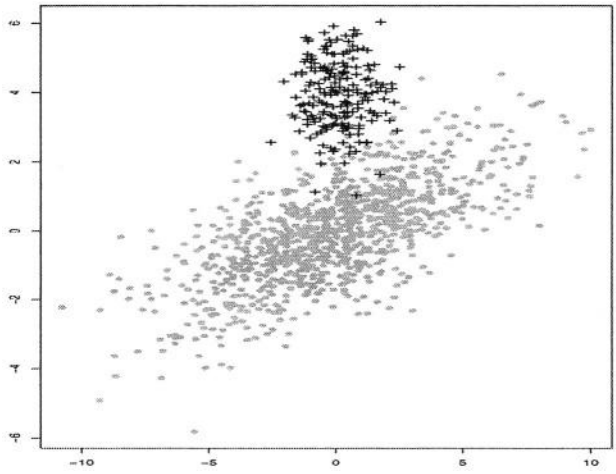


Fig. 2. Artificial data. Two groups with different scattering matrices.

the kernel matrix K^* has been contructed, it is used to train a SVM. For the AV, SQ and SM methods the value of the parameter τ has been assigned via cross-validation.

Given a non labelled data point x, $K(x, x_i)$ has to be evaluated. We can calculate two different values for $K(x, x_i)$, the first one assuming x belongs to class +1 and the second assuming x belongs to class – 1. For each assumption, all we have to do is to compute the distance between x and the SVM hyperplane and assign x to the class corresponding to the largest distance from the hyperplane.

In the following, for all the data sets, we will use 70% of the data for training and 30% for testing.

We have compared the proposed methods with the following classifiers: Multivariate additive regression splines (MARS) [5], Logistic Regression (LR), Linear Discriminant Analysis (LDA), k-nearest neighbour classification (KNN), the MARK combining method [2] and SVMs using a RBF kernel $K_c(x_i, x_j) = e^{-\|x_i - x_j\|^2/c}$, with $c = 0.5d$, where d is the data dimension (see [18]).

3.1 Artificial Data Sets

Two Groups with Different Scattering Matrices. This data set, shown in Figure 2, is made up by 1200 points in $\mathbb{R}^2$, divided in two groups. Each group corresponds to a normal cloud with different covariance matrices. Overlap between the two groups is apparent. We have defined two RBF kernels K_1 and K_2, where for each kernel, the constant c has been chosen as the average of the squared Euclidean distances within each group. Results on this experiment are shown in Table 1.

The new proposed methods based on the use kernel combinations achieve similar results. The decision function for the AV method involves significantly

Table 1. Classification errors for the two groups with different scattering matrices.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	1.1 %	1.2 %	12.9 %
AV	1.1 %	1.9 %	6.0 %
SQ	1.1 %	1.3 %	12.9 %
SM	1.1 %	1.3 %	28.6 %
K-NN	1.5 %	1.4 %	
LDA	2.5 %	1.9 %	
LR	2.6 %	2.6 %	
MARK	0.7 %	7.7 %	39.9%
MARS	2.6 %	2.6 %	
SVM	1.1 %	1.3 %	28.2 %

Table 2. Classification errors for the two groups with different scattering matrices using three kernels.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	1.2 %	1.3 %	17.4 %
AV	1.7 %	1.8 %	7.5 %
SQ	1.1 %	1.3 %	20.8 %
SM	1.0 %	1.3 %	55.3 %

less support vectors than the other methods. Notice that kernels K_1 and K_2 are not specially well suited for the problem at hand. However, their performance is quite successful.

To end with this data set, we check the performance of the proposed four combination methods when three kernels are used as input. We will combine the two kernels previously described and a RBF kernel with $c = 0.5d$. Results are shown in Table 2. Since previous results are very close to the true theoretical error, there is not much room for improvement and similar results are obtained.

Two Kernels with Complementary Information. This data set consists of 400 two-dimensional points (200 per class). Each group corresponds to a normal cloud with mean μ_i and diagonal covariance matrix $\sigma_i^2 I$. Here $\mu_1 = (3, 3)$, $\mu_2 = (5, 5)$, $\sigma_1 = 0.7$ and $\sigma_2 = -0.9$. We have defined two kernels from the projections of the data set onto the coordinate axes. The point in this example is that, separately, both kernels achieve a poor result (a test error of 15%). Table 3 shows the successful results obtained with the combination methods. In particular, the SM method attains the best overall results.

3.2 Real Data Sets

Cancer Data Set. In this section we have dealt with a database from the UCI Machine Learning Repository: the Breast Cancer data set [11]. The data set consists of 683 observations with 9 features each. For this data set we have

Table 3. Classification errors for the kernels with complementary information.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	2.3 %	2.7 %	9.2 %
AV	2.5 %	4.0 %	5.8 %
SQ	0.0 %	2.8 %	35.6 %
SM	5.4 %	2.2 %	20.3 %
K-NN	3.4 %	3.8 %	
LDA	7.5 %	7.8 %	
LR	7.6 %	7.7 %	
MARK	7.7 %	8.0 %	2.0%
MARS	3.4 %	3.9 %	
SVM	3.4 %	3.6 %	26.3 %

Table 4. Classification errors for the cancer data.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	0.0 %	3.4 %	11.4 %
AV	2.9 %	3.4 %	6.1 %
SQ	2.1 %	2.9 %	9.6 %
SM	0.0 %	3.9 %	11.2 %
K-NN	3.9%	4.8 %	
LDA	3.8 %	4.4 %	
LR	13.1 %	13.1 %	
MARK	0.0 %	4.4 %	18.3%
MARS	2.7 %	3.2 %	
SVM	0.0 %	4.4 %	49.5 %

defined two input kernels similar to those in the first example, that is, two RBF kernels K_1 and K_2, where for each kernel, the constant c has been chosen as the average of the squared Euclidean distances within each group. Results are shown in Table 4.

The SQ method shows the best overall performance. Once again the new combination methods provide better results than the SVM with a single kernel, using significantly less support vectors.

A Handwritten Digit Recognition Problem. The experiment in this section is a binary classification problem: the recognition of digits '3' and '9' from the Alpaydin and Kaynak database [1]. The data set is made up by 1134 records, represented by 32 × 32 binary images. We have employed two different methods to specify features in order to describe the images. The first one is the 4×4 method: features are defined as the number of ones in each of the 64 squares of dimension 4×4. The second method was introduced by Frey and Slate [6]: 16 attributes are derived from the image, related to the horizontal/vertical position, width, height, etc. This is a typical example with two sources of information possibly different and perhaps complementary. We use these representations to calculate two kernels from the Euclidean distance. The classification performance

Table 5. Classification errors for the handwritten digit data set.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	0.0 %	0.8 %	4.9 %
AV	0.8 %	0.3 %	13.9 %
SQ	0.0 %	0.8 %	32.6 %
SM	0.0 %	0.6 %	7.4 %
K-NN(4×4)	0.0 %	1.7 %	
K-NN(Frey-Slate)	0.0%	21.5 %	
LDA(4×4)	0.6 %	1.1 %	
LDA(Frey-Slate)	3.2 %	2.2 %	
LR(4×4)	0.0 %	1.7 %	
LR(Frey-Slate)	2.6 %	2.5 %	
MARK	0.0 %	1.4 %	13.0 %
MARS(4×4)	0.8 %	1.1 %	
MARS(Frey-Slate)	1.9 %	3.3 %	
SVM(4×4)	0.0 %	1.1 %	4.0 %
SVM(Frey-Slate)	6.4 %	6.6 %	12.2 %

for all the methods is tabulated in Table 5. In this case the AV and SM methods achieve the best overall performance. Our methods based on kernel combinations improve the results obtained using the rest of the techniques.

3.3 A Text Data Base

To check the methods in a high dimensional setting, we will work on a small text data base with two groups of documents. The first class is made up of 296 records from the LISA data base, with the common topic 'library science'. The second class contains 394 records on 'pattern recognition' from the INSPEC data base. There is a mild overlap between the two classes, due to records dealing with 'automatic abstracting' . We select terms that occur in at least 10 documents (obtaining 982 terms). Labels are assigned to terms by voting on the classes of documents in which these terms appear. The task is to correctly predict the class of each term. Following [12], we have defined the kernel K_1 by $K_1(i,j) = \frac{|x_i \wedge x_j|}{|x_i|} = \frac{\sum_k |\min(x_{ik}, x_{jk})|}{\sum_k |x_{ik}|}$,where $|x_i|$ measures the number of documents indexed by term i, and $|x_i \wedge x_j|$ the number of documents indexed by both i and j terms. Similarly, $K_2 = \frac{|x_i \wedge x_j|}{|x_j|}$. The task is to classify the database terms using the information provided by both kernels. Note that we are dealing with about 1000 points in 600 dimensions, and this is a near empty set. This means that it will be very easy to find a hyperplane that divides the two classes. Notwithstanding, the example is still useful to guess the relative performance of the proposed methods. Following the scheme of the preceding examples, Table 6 shows the results.

Our proposal of methods for the combination of kernels clearly outperform the rest of the methods. In particular, the AV method achieves the best performance. Notice that this is the most difficult example in this section and the one where the advantage of the combination kernel methods is more evident.

Table 6. Classification errors for the term data base. '–' indicates non convergence of the method.

Method	Train error	Test error	Support vectors
$1/2(K_1 + K_2)$	0.0 %	1.8 %	19.6%
AV	0.0 %	1.1 %	8.3 %
SQ	0.0 %	3.0 %	33.6 %
SM	0.0 %	1.8 %	16.8 %
K-NN	12.8 %	14.0 %	
LDA	0.0 %	31.4 %	
LR	– %	– %	
MARK	– %	– %	– %
MARS	– %	– %	
SVM	23.8 %	23.9 %	63.2 %

4 Conclusions

In this work we have proposed three new techniques for the combination of kernels within the context of SVM classifiers. This is an interesting approach to combine classifiers, since domain knowledge is often available in the form of kernel information. The suggested methods compare favorably to other well established techniques and in the comparison with other combining techniques in a variety of artificial and real data sets. Within the group of new kernel combining techniques proposed in this paper, there is not an overall better method. Further research will focus on the theoretical properties of the methods and extensions.

Acknowledgments

This research has been partially supported by grants TIC2003-05982-C05-05 from MCyT and PPR-2003-42 from Universidad Rey Juan Carlos, Spain.

References

1. E. Alpaydin and C. Kaynak. *Cascading Classifiers.* Kybernetika 34 (4) 369-374, 1998.
2. K. Bennett, M. Momma, and J. Embrechts. *MARK: A Boosting Algorithm for Heterogeneous Kernel Models.* Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.
3. C. Cortes and V. Vapnik. *Support Vector Networks.* Machine Learning, 20:273-297, 1995.
4. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.
5. J. Friedman. *Multivariate adaptative regression splines (with discussion).* Annals of Statistics, vol. 19, no. 1, 1-141, 1991.
6. P.W. Frey and D.J. Slate. *Letter Recognition Using Holland-Style Adaptive Classifiers.* Machine Learning, 6 (2) 161-182.

7. L. Goldfarb. *A unified approach to pattern recognition.* Pattern Recognition, 17 (1984) 575-582.
8. T. Joachims. *Learning to Classify Text using Support Vector Machines.* Kluwer, 2002.
9. M.I. Jordan. *Advanced Topics in Learning & Decision Making.* Course material available at www.cs.berkeley.edu/~jordan/courses/281B-spring01.
10. G.R.G. Lanckriet, N. Cristianini, P. Barlett, L. El Ghaoui and M.I. Jordan. *Learning the kernel matrix with semi-definite programming.* Proc. 19th Int Conf Machine Learning, pp. 323-330, 2002.
11. O.L. Mangasarian and W.H. Wolberg. *Cancer diagnosis via linear programming.* SIAM News, Volume 23, Numer 5,1990,1-18.
12. A. Muñoz. *Compound key word generation from document databases using a hierarchical clustering ART model.* Intelligent Data Analysis, vol. 1, pp. 25-48, 1997.
13. A. Muñoz, I. Martín de Diego and J.M. Moguerza. *Support Vector Machine Classifiers for Assymetric Proximities.* Proc. ICANN (2003), LNCS, Springer, 217-224.
14. E. Pekalska, P. Paclík and R.P.W. Duin. *A Generalize Kernel Approach to Dissimilarity-based Classification.* JMLR, Special Issue on Kernel Methods 2 (2) (2002) 175-211.
15. T. Poggio and F. Girosi. *Networks for Approximation and Learning.* Proceedings of the IEEE, 78(10): 1481-1497, 1990.
16. B. Schölkopf, R. Herbrich, A. Smola and R. Williamson. *A Generalized Representer Theorem.* NeuroCOLT2 TR Series, NC2-TR2000-81, 2000.
17. B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K. Müller, G. Rätsch and A. Smola. *Input Space versus Feature Space in Kernel-based Methods.* IEEE Transactions on Neural Networks 10 (5) (1999) 1000-1017.
18. B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola and R.C. Williamson. *Estimating the Support of a High Dimensional Distribution.* Neural Computation, 13(7):1443-1471 , 2001.
19. L. Vandenberghe and S. Boyd. *Semidefinite programming.* SIAM Review, 38(1):49-95, 1996.

Combining Classifiers Using Dependency-Based Product Approximation with Bayes Error Rate*

Hee-Joong Kang

Division of Computer Engineering, Hansung University
389 Samsun-dong 3-ga, Sungbuk-gu, Seoul, Korea
hjkang@hansung.ac.kr

Abstract. Combining classifiers using Bayesian formalism deals with a high dimensional probability distribution composed of a class and the decisions of classifiers. Thus product approximation is needed for the probability distribution. Bayes error rate is upper bounded by the conditional entropy of the class and decisions, so the upper bound should be minimized for raising the class discrimination. By considering the dependency between class and decisions, dependency-based product approximation is proposed in this paper together with its related combination method. The proposed method is evaluated with the recognition of unconstrained handwritten numerals.

1 Introduction

Combining classifiers using Bayesian formalism deals with a high dimensional probability distribution composed of a class and the decisions of classifiers. So, it is usually hard to compute the high dimensional probability distribution without any approximation in real applications. Thus product approximation is needed for the probability distribution. On the assumption that the decisions are conditionally independent of the given class, the high dimensional probability distribution is approximated with a product of two-dimensional component distributions and the decisions are combined with such a product by Xu et al.[1]. This assumption can be regarded as the special case of the first-order dependency among components. The first-order dependency-based product approximation (DBPA) was proposed by Chow and Liu using the measure of closeness criterion in [2], The measure of closeness was devised to measure how close an approximating distribution was to a true distribution for the product approximation. Afterwards, Kang et al. proposed the second-order DBPA scheme in [3] and the third-order DBPA scheme in [4] by considering more than the first-order dependency among components using the same measure of closeness. These DBPAs did not have any constraint in dealing with the components to approximate the high dimensional probability distribution.

On the other hand, Bayes error rate is upper bounded by the conditional entropy of class and variables, and so the upper bound should be minimized for

* This research was financially supported by Hansung University in the year of 2004.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 112–121, 2004.

raising the class discrimination. By considering the dependency between class and variables, another first-order DBPA was proposed by Wang and Wong [5]. That is, Wang and Wong defined the class-patterns (CP) mutual information for the product approximation, as the patterns were regarded as the variables. That is the difference between the product approximation by Wang and Wong and the product approximation by Chow and Liu in dealing with the components in the probability distribution. Kang and Lee applied the concept of CP mutual information to class-decisions (C-D) relationship in combining multiple classifiers and tried to combine multiple classifiers with the product approximation derived from C-D mutual information using Bayesian formalism [6]. The C-D mutual information provides theoretical ground so that the high dimensional probability distribution is optimally approximated with a product of low dimensional component distributions according to the order of dependency. Without any approximation, direct full dependency between class and decisions was considered in the method of Behavior-Knowledge Space (BKS) in [7]. However, the BKS method has both the possibility of high rejection rates due to unseen decisions and the exponential complexity in directly storing and estimating the high dimensional probability distribution. In this paper, another first-order and second-order DBPA schemes based on the results in [6] are proposed with Bayes error rate using the C-D mutual information, as the extended work of the first-order DBPA by Wang and Wong. The proposed methods are evaluated by combining multiple classifiers on the recognition of unconstrained handwritten numerals.

As for recognition experiments, the unconstrained handwritten numerals are from Concordia University [8] and the University of California, Irvine (UCI) [9]. Totally six classifiers are combined at abstract level, where these classifiers were developed by using the features or methodologies in [10,11]. The Bayesian combination methods based on those mentioned DBPAs are introduced in the recognition experiments as well as the BKS method.

This paper is organized as follows. Section 2 explains the product approximation schemes using the C-D mutual information with Bayes error rate. Bayesian combination using the proposed product approximation schemes is defined in Section 3. Experimental results for evaluating the proposed methods with other Bayesian combination methods are provided in Section 4 and the concluding remarks are given in Section 5.

2 Product Approximation Scheme Using C-D Mutual Information

A dependency-based approach with Bayes error rate plays an intermediate role between the independence assumption and the BKS method in several aspects. Considering the dth-order dependency makes the storage needs been $((K-d)\cdot L^{d+2})$ and it also makes potentially high rejection rates lowered due to the product approximation. In other words, the space complexity of the proposed approximation scheme, i.e. $O(L^{d+2})$, is from $O(L^3)$ of the first-order dependency

to $O(L^{K+1})$ of the BKS method, because $(1 \leq d \leq (K-1))$. The proposed approximation scheme also supports several probabilistic combination methods from the dependence tree method to the BKS method, according to the order of dependency d under permissible resources.

Hellman and Raviv proved an inequality expression between the Bayes error rate P_e and the conditional entropy $H(M|C)$ of class M and variables C, as the following Eq. (1) in [12]. This paper regards variables as decisions. The Bayes error rate P_e is upper bounded by the conditional entropy $H(M|C)$. Thus, the C-D mutual information $U(M; C)$ is defined from the conditional entropy in Eq. (1) and measures the degree of dependence between the class M and the decisions C, as the following expressions:

$$P_e \leq \frac{1}{2}H(M|C) = \frac{1}{2}(H(M) - U(M;C)) \tag{1}$$

$$U(M;C) = \sum_m \sum_c P(m,c) \log \frac{P(m,c)}{P(m)P(c)} \tag{2}$$

where $H(M)$ is the entropy. Minimizing the upper bound of P_e for class discrimination leads to maximizing the C-D mutual information $U(M;C)$, since the entropy $H(M)$ is constant with regard to C. Thus an optimal product set is obtained by maximizing the C-D mutual information.

2.1 First-Order DBPA

When K decisions, $C_1, \cdots, C_K$, are combined, a first-order DBPA is obtained by considering the first-order dependency among the decision components in the probability distribution. The approximating distribution of C is defined in terms of two-dimensional distributions as follows:

$$P_a(C_1, \cdots, C_K) = \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i(j)}}), \quad (0 \leq i(j) < j) \tag{3}$$

and the approximating distribution of C and M is defined in terms of three-dimensional distributions as follows:

$$P_a(C_1, \cdots, C_K, M) = \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i(j)}}, M), \quad (0 \leq i(j) < j) \tag{4}$$

$$P_a(C_1, \cdots, C_K|M) = \frac{1}{P(M)} \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i(j)}}, M), \quad (0 \leq i(j) < j) \tag{5}$$

and C_{n_j} is conditioned on both $C_{n_{i(j)}}$ and M, and where $(n_1, \cdots, n_K)$ is an unknown permutation of integers $(1, \cdots, K)$ and C_0 is a null component. And $P(C_{n_j}|C_0, M)$ is equal to $P(C_{n_j}, M)$, by definition. The first-order dependency makes the C-D mutual information expanded like the following expressions by using the Eqs. (3)–(5) and dropping the subscript n of C:

$$U(M;C) = \sum_m \sum_c P(m,c) \log \frac{P(m,c)}{P(m)P(c)}$$

$$= \sum_m \sum_c P(c,m) \log P(c|m) - \sum_m \sum_c P(c,m) \log P(c)$$

$$= -\sum_m P(m) \log P(m) + \sum_{j=1}^{K} \sum_m \sum_c P(c,m) \log P(C_j|C_{i(j)}, m)$$

$$- \sum_{j=1}^{K} \sum_c P(c) \log P(C_j|C_{i(j)})$$

$$= H(M) + \sum_{j=1}^{K} [I(C_j; C_{i(j)}, M) - I(C_j; C_{i(j)})] \tag{6}$$

$$H(M) = -\sum_m P(m) \log P(m) \tag{7}$$

$$\Delta I(C_j; C_{i(j)}) = I(C_j; C_{i(j)}, M) - I(C_j; C_{i(j)}) \ . \tag{8}$$

Thus, it is obvious that the total sum of Delta(Δ) first-order C-D mutual information $\Delta I(Cj; Ci(j))$ should be maximized. With the dependence tree method of Wang and Wong, we can determine both the unknown permutation and its conditioned permutation from the chosen optimal dependence tree. The time complexity to find the dependence tree is $O(n \log n)$.

2.2 Second-Order DBPA

A second-order DBPA is obtained by considering the second-order dependency among the decision components in the probability distribution. The approximating distribution of C is defined in terms of three-dimensional distributions as follows:

$$P_a(C_1, \cdots, C_K) = \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i2(j)}}, C_{n_{i1(j)}}), \ (0 \le i2(j) \le i1(j) < j) \tag{9}$$

and the approximating distribution of C and M is defined in terms of four-dimensional distributions as follows:

$$P_a(C_1, \cdots, C_K, M) = \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i2(j)}}, C_{n_{i1(j)}}, M), \ (0 \le i2(j) \le i1(j) < j) \tag{10}$$

$$P_a(C_1, \cdots, C_K|M) = \frac{1}{P(M)} \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i2(j)}}, C_{n_{i1(j)}}, M), \ (0 \le i2(j) \le i1(j) < j) \tag{11}$$

and C_{n_j} is conditioned on $C_{n_{i2(j)}}$, $C_{n_{i1(j)}}$, and M, and where $(n_1, \cdots, n_K)$ is an unknown permutation of integers $(1, \cdots, K)$ and C_0 is a null component. Thus $P(C_{n_j}|C_0, C_0, M)$ is equal to $P(C_{n_j}, M)$, and $P(C_{n_j}|C_0, C_{n_{i1(j)}}, M)$ is equal

to $P(C_{n_j}|C_{n_{i1(j)}}, M)$, by definition. The second-order dependency makes the C-D mutual information expanded like the following expressions by using the Eqs. (9)–(11) and dropping the subscript n of C:

$$
\begin{aligned}
U(M;C) &= \sum_m \sum_c P(c,m) \log \frac{P(c|m)}{P(c)} \\
&= \sum_{m,c} P(c,m) \log[\frac{1}{P(m)} \prod_{j=1}^{K} P(C_j|C_{i2(j)}, C_{i1(j)}, m)] \\
&\quad - \sum_c P(c) \log \prod_{j=1}^{K} P(C_j|C_{i2(j)}, C_{i1(j)}) \\
&= -\sum_m P(m) \log P(m) + \sum_{j=1}^{K} \sum_{m,c} P(c,m) \log P(C_j|C_{i2(j)}, C_{i1(j)}, m) \\
&\quad - \sum_{j=1}^{K} \sum_c P(c) \log P(C_j|C_{i2(j)}, C_{i1(j)}) \\
&= H(M) + \sum_{j=1}^{K} [I(C_j; C_{i2(j)}, C_{i1(j)}, M) - I(C_j; C_{i2(j)}, C_{i1(j)})] \qquad (12)
\end{aligned}
$$

$$
H(M) = -\sum_m P(m) log P(m) \qquad (13)
$$

$$
\Delta I(C_j; C_{i2(j)}, C_{i1(j)}) = I(C_j; C_{i2(j)}, C_{i1(j)}, M) - I(C_j; C_{i2(j)}, C_{i1(j)}) \ . \qquad (14)
$$

From the above derived Eq. (12), maximizing $U(M;C)$ leads to maximizing $\sum_{j=1}^{K} \Delta I(C_j; C_{i2(j)}, C_{i1(j)})$ which is the total sum of Δ second-order C-D mutual information, since remaining entropy term $H(M)$ is also constant with regard to C. Then, the next step is how to find an optimal product set by the second-order dependency from all the permissible product sets. Finding the optimal product set by the second-order dependency is to select the maximum sum of Δ second-order C-D mutual information covering Δ first-order C-D mutual information, as described in the following algorithm. From the found optimal product set, we can determine the unknown permutation $(n_1, \cdots, n_K)$ and their two unknown conditioned permutations $(n_{i2(1)}, \cdots, n_{i2(K)})$ and $(n_{i1(1)}, \cdots, n_{i1(K)})$. The time complexity of the algorithm is $O(n^2)$.

Input:

A set of $(K + 1)$-dimensional samples of C and M.

Output:

An optimal product set by the second-order dependency as per the Δ second-order C-D mutual information.

Method:

1. Estimate two-, three-, and four-dimensional marginal distributions from the samples.

2. Compute the weights $\Delta I(C_j; C_{i(j)})$ and $\Delta I(C_j; C_{i2(j)}, C_{i1(j)})$ for all pairs, and triplets of classifiers from the estimated marginal distributions.
3. Compute the maximum weight sum consisted of Δ first-order and Δ second-order C-D mutual information and find its associated optimal product set, as the following statements:

maxTweight = 0;
for $n = 1$ **to** no. of Δ first-order C-D mutual information **do**
  Tweight = weight of the n-th $\Delta I(C_j; C_{i(j)})$;
  while ((no. of untraversed classifiers) > 0) **do**
    choose one of untraversed classifiers and mark it traversed;
    choose the largest permissible Δ second-order C-D mutual information associated with the chosen classifier and one traversed classifier among all traversed classifiers;
    Tweight += weight of the chosen $\Delta I(C_j; C_{i2(j)}, C_{i1(j)})$;
  end
  maxTweight = *MAX*(*maxTweight*, *Tweight*);
  store *maxTweight* and its associated Δ first-order and Δ second-order C-D mutual information;
end
obtain maximum *maxTweight* and its associated Δ first-order and Δ second-order C-D mutual information;

By using the systematic approach for product approximation, the order of dependency considered can be easily extended to the dth-order under permissible computing resources. Considering the dth-order dependency makes the approximating distributions Eqs. (9)–(11) changed as to the order of dependency d. An optimal product set by the dth-order dependency consists of one by Δ first-order C-D mutual information, one by Δ second-order C-D mutual information, ..., one by Δ $(d-1)$st-order C-D mutual information, and multiple (i.e. $(K-d)$) component distributions by Δ dth-order C-D mutual information.

3 Bayesian Combination Methods

After the optimal dependence tree by the first-order dependency is found and all unknown permutations are determined, a Bayesian decision rule for the combination is derived from the Bayesian formalism and the optimal product set by the first-order dependency. For a hypothesized class m, a supported belief function $Bel(m)$ is defined by the following expressions using the Eq. (4):

$$Bel(m) = P(m \in \mathbf{M}|C_1, \cdots, C_K) \tag{15}$$
$$= \frac{P(C_1, \cdots, C_K, m)}{P(C_1, \cdots, C_K)} = \frac{\prod_{j=1}^{K} P(C_{n_j}|C_{n_{i(j)}}, m)}{P(C_1, \cdots, C_K)}$$
$$\approx \eta \prod_{j=1}^{K} P(C_{n_j}|C_{n_{i(j)}}, m) \tag{16}$$

with η as a constant that ensures that $\sum_{i=1}^{L} Bel(m_i) = 1$ and $(n_1, \cdots, n_K)$ is an unknown permutation of integers $(1, \cdots, K)$ where L is the number of classes. Therefore, the combination of classifiers by the first-order dependency is to determine a hypothesized class m which maximizes the supported belief function $Bel(m)$ in the Eq. (16).

After an optimal product set by the second-order dependency is found and all unknown permutations are determined, Bayesian combination rule is also derived from using the Bayesian formalism and the optimal product set by the second-order dependency. For a hypothesized class m, its supported belief function $Bel(m)$ is defined by the following expressions using the Eqs. (10) and (15):

$$\begin{aligned} Bel(m) &= P(m \in \mathbf{M} | C_1, \cdots, C_K) = \frac{P(C_1, \cdots, C_K, m)}{P(C_1, \cdots, C_K)} \\ &= \frac{\prod_{j=1}^{K} P(C_{n_j} | C_{n_{i2(j)}}, C_{n_{i1(j)}}, m)}{P(C_1, \cdots, C_K)} \\ &\approx \eta \prod_{j=1}^{K} P(C_{n_j} | C_{n_{i2(j)}}, C_{n_{i1(j)}}, m) \end{aligned} \tag{17}$$

with η as a constant that ensures that $\sum_{i=1}^{L} Bel(m_i) = 1$. Therefore, the combination of classifiers by the second-order dependency is to determine a hypothesized class m which maximizes the supported belief function $Bel(m)$ in the Eq. (17). Depending on the belief value $Bel(m)$, we can choose a maximized posterior probability $P^*(m \in M | C_1, \cdots, C_K)$, and then a combined decision is determined or not, according to the decision rule $D(C)$ given below:

$$D(C) = \begin{cases} m_i, & \text{if } Bel(m_i) = \max\limits_{m_j \in M} Bel(m_j) \\ L+1, & \text{otherwise} \end{cases} .$$

4 Experimental Results

The six classifiers, E_1, E_2, E_3, E_4, E_5, E_6, are used for the recognition experiments of the unconstrained handwritten numerals from Concordia University and the University of California, Irvine (UCI). These classifiers were developed by using the features in [10,11] or by using the structural knowledge of numerals, such as bounding, centroid, and the width of horizontal runs or strokes, at KAIST and Chonbuk National University. Some of them are back-propagation singular or modular neural networks and the others are rule-based modular classifiers. Classifiers E_2 and E_3 are modular architecture and use directional distance distribution and mesh features respectively. Classifiers E_1 and E_6 are singular architecture and use pixel distance function and contour features respectively. Since classifiers E_4 and E_5 were built by the structural knowledge obtained from Concordia numerals, they are not relatively good at UCI numerals due to high rejection rates. The details on the handwritten numeral databases

are in [8] and [9]. The Concordia numerals consist of two training data sets *A,B* and one test data set *T*. Each data set has 200 digits per class. The UCI samples consist of one training data set *tra,* one cross-validation data set *cv,* one writer-dependent test data set *wdep,* and one writer-independent test data set *windep.* Each data set has variable number of digits per class. While the data sets *cv, wdep* have about 95 digits per class, the data sets *tra, windep* have about 185 digits per class. The neural network classifiers were trained with the data sets *A, tra.* For an optimal product set in DBPA scheme, all the data sets except *T* and *windep* were used in each set of numerals. The performance of individual classifiers is shown in Table 1 with recognition and reliability rates for respective test data sets *T* and *windep.*

Table 1. Performance of individual classifiers

	Concordia *T*		UCI *windep*	
Classifier	Recognition(%)	Reliability(%)	Recognition(%)	Reliability(%)
E_1	96.00	96.00	93.77	93.77
E_2	95.95	95.95	97.11	97.11
E_3	84.45	96.24	91.82	96.95
E_4	90.95	99.02	67.67	93.11
E_5	88.15	98.38	70.01	94.80
E_6	94.15	94.15	96.66	96.66

The classifiers were combined with the test data sets *T*, *windep,* using the following combination methods: the BKS method in [7], and the several Bayesian combination methods as noted in Table 2. Among the Bayesian combination methods, CIAB method was proposed in [1], and ODB1, CODB1, and ODB2 methods were proposed in [3], and CODB2 and ODB3 methods were proposed in [4], and the DODB1 and DODB2 methods are proposed in this paper by using the Δ first-order and Δ second-order C-D mutual information, respectively.

Table 2. Bayesian combination methods

Notation	Full Term
CIAB	conditional independence assumption-based method
ODB1	first-order dependency-based method
CODB1	conditional 1st-order dependency-based method
ODB2	2nd-order dependency-based method
CODB2	conditional 2nd-order dependency-based method
ODB3	3rd-order dependency-based method
DODB1	Δ 1st-order dependency-based method
DODB2	Δ 2nd-order dependency-based method

The first experiment is to combine five classifiers evenly selected from the six classifiers, so six groups were made from 5G1 to 5G6. The best recognition rate in each group in Tables 3 and 4 was mainly obtained by the DODB1 or DODB2 method.

Table 3. Results of five classifiers on Concordia data set: T

Group	CIAB	ODB1	CODB1	ODB2	CODB2	ODB3	DODB1	DODB2	BKS
5G1	97.40	96.95	97.70	97.65	97.15	97.15	97.90	97.80	93.25
5G2	97.20	97.20	97.95	97.80	97.70	97.25	97.95	97.90	92.35
5G3	96.80	96.80	97.80	97.50	97.45	97.45	97.65	97.80	92.30
5G4	97.45	97.10	97.85	97.35	97.95	97.95	98.15	97.95	93.20
5G5	97.35	97.00	98.25	97.95	97.55	97.55	98.25	97.90	92.90
5G6	97.60	96.85	97.60	95.85	97.40	97.40	98.10	97.95	92.85

Table 4. Results of five classifiers on UCI data set: *windep*

Group	CIAB	ODB1	CODB1	ODB2	CODB2	ODB3	DODB1	DODB2	BKS
5G1	97.16	97.44	98.00	98.00	96.38	98.00	98.05	98.27	93.16
5G2	97.05	97.05	97.61	97.77	97.44	97.61	97.89	98.05	93.21
5G3	97.22	97.22	97.77	98.11	98.00	97.16	98.44	98.11	93.60
5G4	96.99	96.83	97.83	97.33	96.66	96.66	98.27	98.33	93.99
5G5	96.99	97.05	97.61	97.61	97.72	97.72	97.55	97.72	93.04
5G6	97.33	97.44	97.38	98.11	98.22	96.99	98.27	98.22	93.88

The second experiment is to combine all six classifiers, so two groups were made according to the source of test data, i.e. 6G1 is for Concordia University and 6G2 for UCI. The best recognition rate in each group in Table 5 was obtained by the DODB1 or DODB2 method.

Table 5. Results of six classifiers on data sets T and *windep*

Group	CIAB	ODB1	CODB1	ODB2	CODB2	ODB3	DODB1	DODB2	BKS
6G1	97.95	97.20	98.00	97.80	97.90	97.90	98.05	98.10	91.40
6G2	97.55	97.22	97.77	98.05	98.00	96.99	98.22	98.11	91.93

The experimental results supported that the proposed DBPAs with Bayes error rate and their combination methods, DODB1 and DODB2, contributed to improvement on the performance over other Bayesian combination methods by t-test at significance level 0.1 as for the group of five classifiers, although it required larger storage needs than the previous Bayesian methods for computing the Δ nth-order C-D mutual information. Particularly, the low recognition rates of the BKS method were caused by the lack of large enough and well representative training data sets.

5 Concluding Remarks

This paper extended the works of Wang and Wong to the second-order dependency and reviewed the dependency between class and decisions with the defined C-D mutual information and the BKS method, accordingly an algorithm using the Δ second-order C-D mutual information was described for the second-order

dependency, too. In order to raise the class discrimination power in combining multiple classifiers, the C-D mutual information was defined for the dependency between class and decisions, and the product approximation was found so that the upper bound of Bayes error rate should be minimized. Thus the best recognition rates were obtained with the proposed DBPAs and the superiority of their associated combination methods over other methods was statistically significant.

References

1. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. IEEE TSMC **22** (1992) 418–435
2. Chow, C.K., Liu, C.N.: Approximating Discrete Probability Distributions with Dependence Trees. IEEE Trans. on Information Theory **14** (1968) 462–467
3. Kang, H.J., Kim, K., Kim, J.H.: Optimal Approximation of Discrete Probability Distribution with *k*th-order Dependency and Its Applications to Combining Multiple Classifiers. PRL **18** (1997) 515–523
4. Kang, H.J.: Combining multiple classifiers based on third-order dependency for handwritten numeral recognition. PRL **24** (2003) 3027–3036
5. Wang, D.C.C., Wong, A.K.C.: Classification of Discrete Data with Feature Space Transform. IEEE TAC **AC-24** (1979) 434–437
6. Kang, H.J., Lee, S.W.: Combining Classifiers based on Minimization of a Bayes Error Rate. In: Proc. of the 5th ICDAR. (1999) 398–401
7. Huang, Y.S., Suen, C.Y.: A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals. IEEE Trans. on PAMI **17** (1995) 90–94
8. Suen, C.Y., Nadal, C., Legault, R., Mai, T.A., Lam, L.: Computer Recognition of Unconstrained Handwritten Numerals. Proc. of IEEE (1992) 1162–1180
9. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
10. Matsui, T., Tsutsumida, T., Srihari, S.N.: Combination of Stroke/Background Structure and Contour-direction Features in Handprinted Alphanumeric Recognition. In: Proc. of the 4th IWFHR. (1994) 87–96
11. Oh, I.S., Suen, C.Y.: Distance features for neural network-based recognition of handwritten characters. IJDAR **1** (1998) 73–88
12. Hellman, M.E., Raviv, J.: Probability of error, equivocation, and the Chernoff bound. IEEE Trans. on Information Theory **IT-16** (1970) 368–372

Combining Dissimilarity-Based One-Class Classifiers

Elżbieta Pękalska, Marina Skurichina, and Robert P.W. Duin

ICT Group, Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, The Netherlands
{e.pekalska,m.skurichina,r.p.w.duin}@ewi.tudelft.nl

Abstract. We address a one-class classification (OCC) problem aiming at detection of objects that come from a pre-defined target class. Since the non-target class is ill-defined, an effective set of features discriminating between the targets and non-targets is hard to obtain. Alternatively, when raw data are available, dissimilarity representations describing an object by its dissimilarities to a set of target examples can be used.
A complex problem can be approached by fusing information from a number of such dissimilarity representations. Therefore, we study both the combined dissimilarity representations (on which a single OCC is trained) as well as fixed and trained combiners applied to the outputs of the base OCCs, trained on each representation separately. An experiment focusing on the detection of diseased mucosa in oral cavity is conducted for this purpose. Our results show that both approaches allow for a significant improvement in performance over the best results achieved by the OCCs trained on single representations, however, concerning the computational cost, the use of combined representations might be more advantageous.

1 Introduction

Novelty detection problems arise in applications, where anomalies or outliers should be recognized. Given training examples, the goal is to describe the so-called target class such that resembling objects are accepted as targets and outliers (non-targets) are rejected. Such a detection has to be performed in an unknown or ill-defined context of alternative phenomena. Examples refer to health diagnostics, machine condition monitoring, industrial inspection or face detection. The target class is assumed to be well sampled and well defined. The alternative non-target (outlier) set is usually ill-defined: it is badly sampled (even not present at all) with unknown and hard to predict priors. If available, such non-targets might be structured in ways not represented in the training set. For such types of problems *one-class classifiers* (OCCs) may be very suitable [15, 10], as they are domain or boundary descriptors.

Since the non-target class is ill-defined, in complex problems, an effective set of features discrimination between targets and non-targets cannot be easily found. Hence, it seems appropriate to build a representation on the raw data.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 122–133, 2004.

The dissimilarity representation, describing objects by their dissimilarities to the target examples, may be effective for such problems since it naturally protects the target class against unseen novel examples. Therefore, we will study dissimilarity representations to train one-class classifiers. Optimal representations and dissimilarity measures cannot be found if one of the classes is missing or badly sampled. On the other hand, when one analyzes a particular phenomenon, the model knowledge can be captured by various dissimilarity representations describing different problem characteristics. In this way, a problem is tackled from a wider perspective: each additional representation may incorporate useful information. Combining OCCs becomes, thereby, a natural technique needed for solving ill-defined (unbalanced) detection problems. Note, however, that standard two-class classifiers should be preferred if the non-target class is well represented.

Although such problems are often met in practice, representative standard datasets do not exist yet. They should be based on the raw data and various dissimilarity measures should be available. Our procedures here are not intended for general multi-class problems for which other, more suitable, techniques exist. Our methodology is applicable to difficult problems where the target examples are provided with or without additional outlier examples. For that reason, the effectiveness of the proposed procedures is illustrated with just a single, yet complex, application, i.e. the detection of diseased mucosa in oral cavity.

Two approaches are compared within this application. The first one focuses on combining dissimilarity representations into a single one, while the second approach considers a combiner operating on the outputs of the OCCs. This study extends results of our earlier work [9] devoted to usual classification tasks. Note, however, that OCCs do not directly estimate the posterior probabilities since they rely on information on a target class. OCCs output a sort of a signed distance to the boundary.

2 One-Class Classifiers for Dissimilarity Representations

Consider a representation set $R = \{p_1, p_2, \ldots, p_n\}$, which is a set of representative objects. $d(x, p_i)$ denotes a dissimilarity between the objects x and p_i, independently from their initial representations. In general, we do not require metric properties of d, since non-metric dissimilarities may arise when shapes or objects in images are compared; see e.g. [6]. The usefulness of d is judged by its construction and a fit to the problem; d should be relatively small for objects resembling each other in reality and large for objects that differ. Obviously, the non-negativity and reflexivity, i.e. $d(x, y) \geq 0$ and $d(x, x) = 0$ are taken as granted. Thereby, a dissimilarity representation (DR) of an object x is expressed as a vector $D(x, R) = [d(x, p_1), d(x, p_2), \ldots, d(x, p_n)]$. For a collection of training objects $T = \{t_1, t_2, \ldots, t_N\}$, it extends to a $N \times n$ dissimilarity matrix $D(T,R)$. In general, R might be a subset of T ($R \subseteq T$) or they might be distinct sets.

There are three principal learning approaches, referring to three interpretations of DRs, for which a particular methodology can be adapted. In the *pre-*

topological approach (I), the dissimilarity values are interpreted directly, hence they can be characterized in pretopological spaces [7,12], where the neighborhoods play a significant role. The *embedding* approach (II) builds on a spatial representation, i.e. an embedded pseudo-Euclidean configuration such that the dissimilarities are preserved [5,8]. In the *dissimilarity space* approach (III), one considers $D(x,R): \mathcal{X} \to \mathcal{R}^r$ as a data-depending mapping to the so-called dissimilarity space. In such a space, every dimension corresponds to a dissimilarity $D(\cdot, p_i)$ to a particular object $p_i \in R$. So, the dimensions convey a homogeneous type of information. The property that dissimilarities should be small for similar objects (belonging to the same class) and large for distinct objects, gives a possibility for a discrimination. Thereby, $D(\cdot, p_i)$ can be interpreted as an attribute.

Below, some exemplar one-class classifiers are described, which in practice rely on some proximity $f_{\text{prox}}(x, \omega_T)$ of an object x to the target class ω_T is computed. To decide whether an object belongs to the target class or not, a threshold γ on f_{prox} should be determined. A standard way is to supply a fraction r_{fn} of (training) target objects to be rejected by the OCC (a false negative ratio) [14,13]. This means that γ is set up such that $\int \mathcal{I}(f_{\text{prox}}(x, \omega_T) > \gamma)\, d\mu(x) = r_{\text{fn}}$, where $\mathcal{I}$ is the indicator function and μ is some measure. r_{fn} is a small value to prevent a high acceptance of outliers as targets. In other cases, γ can be determined as the $(1-r_{\text{thr}})$-percentile of the sorted sequence of the proximity outputs computed for the training (target) examples. r_{thr} is then a user-specified fraction. Unless stated otherwise, $R \subseteq T$ consists of the target examples only.

I. Neighborhood-Based OCC. The nearest-neighbor data description (NNDD) is realized by the classifier $\mathcal{C}_{\text{NNDD}}$ indirectly built in a pretopological space. The proximity function relies on the nearest neighbor dissimilarities. For n target training objects t_i, a vector of averaged nearest neighbor dissimilarities $d_{nn}(t_i, R) = \frac{1}{k}\sum_{j=1}^{k} d(t_i, p_{t_i}^j)$, where $p_{t_i}^j$ is the j-th nearest neighbor of t_i in R, *is* obtained. Then, a threshold γ is determined based on the $(1-r_{\text{thr}})$-th percentile of the sorted sequence of d_{nn}. The classifier becomes then:

$$\mathcal{C}_{\text{NNDD}}(D(x,R)) = \mathcal{I}(d_{nn}(x,R) \le \gamma) = \mathcal{I}(\frac{1}{k}\sum_{j=1}^{k} d(x, p_x^j)) \le \gamma), \quad p_x^j \in R. \quad (1)$$

II. Generalized Mean-Class OCC (GMDD). Assume a symmetric representation $D(R, R)$, where R consists of the targets only. Any such matrix D can be embedded in a pseudo-Euclidean space[1] given dissimilarities are preserved perfectly [5,8,7]. ($\mathcal{E}$ becomes Euclidean iff D is Euclidean.) If $D(T, R)$, $R \subset T$, is given, then $\mathcal{E}$ is determined by $D(R, R)$ and the remaining $T \backslash R$ objects are then projected to $\mathcal{E}$. In the embedded space $\mathcal{E}$, a simple OCC can be designed relying on the distance to the mean vector of the target class. This can be, however, carried out without performing the exact embedding. It can be proved

[1] A pseudo-Euclidean space $\mathcal{E} := \mathcal{R}^{(p,q)}$ is a non-degenerate indefinite inner product space such that the inner product $\langle \cdot, \cdot \rangle_{\mathcal{E}}$ is positive definite on $\mathcal{R}^p$ and negative definite on $\mathcal{R}^q$. $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{E}} = \sum_{i=1}^{q} x_i y_i - \sum_{i=p+1}^{p+q} x_i y_i$ and $d_{\mathcal{E}}^2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{E}}^2 = \langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle_{\mathcal{E}} = d_{\mathcal{R}^p}^2(\boldsymbol{x}, \boldsymbol{y}) - d_{\mathcal{R}^q}^2(\boldsymbol{x}, \boldsymbol{y})$. Since $\mathcal{E}$ is a linear space, many properties based on inner products can be appropriately extended from the Euclidean case.

that the proximity function $f_{\text{prox}}(x, \omega_T) = ||\boldsymbol{x}_\mathcal{E} - \bar{\boldsymbol{x}}_\mathcal{E}||^2_\mathcal{E}$ in $\mathcal{E}$ (where $\boldsymbol{x}_\mathcal{E}$ is the projection of $D(x, R)$ to $\mathcal{E}$) is equivalently computed by the use of square dissimilarities as $f_{\text{prox}}(x, \omega_T) = \frac{1}{n}\sum_{i=1}^{n} d^2(x, p_i) - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d^2(p_i, p_j)$; see [8, 9,7] for details. Then, a threshold γ is determined as the $(1-r_{\text{thr}})$-th percentile of the sorted sequence of $f_{\text{prox}}(t_i, \omega_T)$. The generalized mean-class data description (GMDD) becomes then:

$$\mathcal{C}_{\text{GMDD}}(D(x, R)) = \mathcal{I}(\frac{1}{n}\sum_{i=1}^{n} d^2(x, p_i) - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} d^2(p_i, p_j) \leq \gamma). \quad (2)$$

III. Linear Programming Dissimilarity-Data Description (LPDD). This OCC was proposed by us in [9]. It is designed as a hyperplane $H: \boldsymbol{w}^T D(x, R) = \rho$ in a dissimilarity space that bounds the target data from above (we assume that d is bounded) and which is attracted towards the origin. Non-negative dissimilarities impose both $\rho \geq 0$ and $w_i \geq 0$. This is achieved by minimizing $\rho/||\boldsymbol{w}||_1$, which is the max-norm distance of the hyperplane H to the origin in the dissimilarity space. Hence, H can be determined by minimizing $\rho - ||\boldsymbol{w}||_1$. Assuming that $||\boldsymbol{w}||_1 = 1$ (to avoid any arbitrary scaling of $\boldsymbol{w}$), H is found by the minimization of ρ only. A target class is then characterized by a linear proximity function on dissimilarities with the weights $\boldsymbol{w}$ and the threshold ρ. The LPDD is then defined as:

$$\mathcal{C}_{\text{LPDD}}(D(x, R)) = \mathcal{I}(\sum_{w_j \neq 0} w_j D(x, p_j) \leq \rho), \quad (3)$$

where w_j are found as the solution to a soft-margin linear programming formulation (the hard-margin case is then straightforward) with $\nu \in (0, 1]$ being the upper bound on the target rejection fraction in training (here $\nu := r_{fn}$ is used) [9]:

$$\begin{aligned} &\min \rho + \tfrac{1}{\nu N}\textstyle\sum_{i=1}^{N} \xi_i \\ &\text{s.t. } \boldsymbol{w}^T D(p_i, R) \leq \rho + \xi_i, \quad \textstyle\sum_j w_j = 1,\ w_j \geq 0,\ \rho \geq 0,\ \xi_i \geq 0, \quad i = 1, 2, .., N. \end{aligned}$$

As a result, sparse solutions are obtained, i.e. only some w_j are non-zero. Objects of R corresponding to such non-zero weights are called *support objects* (SO). The LPDD can be extended to handle example outliers as well. A label variable $y_i \in \{+1, -1\}$ is used to encode the targets (1) and outliers (-1). The formulation above remains the same, but the main constraint changes to $y_i(\boldsymbol{w}^T D(p_i, R)) \leq y_i \rho + \xi_i$.

2.1 How Good Is an OCC?

To study the behavior of an OCC, the ROC curve [2,14] is often used. It is a function of the true positive (target acceptance) versus the false positive (outlier acceptance) ratio. Example outliers are necessary for its evaluation. In principle, an OCC is trained with a fixed target rejection ratio r_{fn} for which the threshold is determined. This OCC is then optimized for one point on the ROC curve.

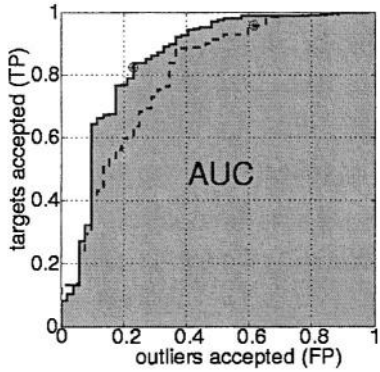


Fig. 1. A ROC curve for the LPDD.

To compare the performance of various classifiers, the AUC measure is used [1], It computes the Area Under the Curve, which is the total OCC's performance integrated over all the thresholds. The larger AUC, the better the OCC; e.g. in Fig. 1, the solid curve (the LPDD trained using outliers) indicates a better performance than the dashed curve (the LPDD trained on the targets only). The stars indicate points for which the thresholds were optimized.

2.2 Combined Representations

Learning from distinct DRs can be realized by combining them into a new representation and then training a single OCC. As a result, a more powerful representation may be obtained, allowing for a better discrimination. Suppose that K representations $D^{(\tau)}(T,R)$, $\tau = 1,2,\ldots,K$, all based on the same R, are given. Assume that the dissimilarity measures are similarly bounded (if not they can be scaled appropriately), since only then we can somehow relate their values to each other (otherwise we would need to compare not the direct values but the corresponding percentiles). The DRs can be combined, for instance, in the following ways:

D_{comb}	Expression
Avr	$D_{\text{avr}}(t_i,p_j) = \frac{1}{K}\sum_{\tau=1}^{K} D^{(\tau)}(t_i,p_j)$
Prod	$D_{\text{prod}}(t_i,p_j) = \sum_{\tau=1}^{K} \log\left(1+D^{(\tau)}(t_i,p_j)\right)$
Min	$D_{\min}(t_i,p_j) = \min_\tau\{D^{(\tau)}(t_i,p_j)\}$
Max	$D_{\max}(t_i,p_j) = \max_\tau\{D^{(\tau)}(t_i,p_j)\}$

The DRs are combined into one representation by using a sort of fixed rules, usually applied when outputs of two-class classifiers are combined. Note that a DR can be interpreted as a collection of weak classifiers, where each of them is understood as a dissimilarity $D^{(\tau)}(\cdot,p_i)$ to a particular object p_i. In contrary to probabilities, a small dissimilarity value $D^{(\tau)}(t_j,p_i)$ is an evidence of a good

'performance', indicating here that the object t_j is similar to the target p_i. In general, different dissimilarity measures focus on different aspects of the data. Hence, each of them estimates a proximity of an object x to the target p_i as $D^{(\tau)}(x, p_i)$. So, D_{avr} yields an average proximity estimator. When, the dissimilarity measures are independent (e.g. one built on statistical and the other on structural object properties), the product combiner can be of interest. Logically, both D_{avr} and D_{prod} should integrate the strengths of various representations. Here, D_{prod} is expressed such that very small numbers are avoided (they could arise when multiplying close-to-zero dissimilarities). The min operator chooses the minimal dissimilarity value $D^{(\tau)}(x, p_i)$, $\tau = 1, \ldots, K$, hence the maximal evidence for an object x resembling the target t_i. The max operator works the other way around.

2.3 Combined Classifiers

One usually combines classifiers based on their posterior probabilities. The outputs of the OCCs may be converted to estimates of probabilities [14] and standard fixed combiners, such as mean, product and majority voting, can be considered. Here, we also like to proceed with the exact OCCs outputs. For this reason, we focus the LPDDs. Each LPDD is determined by a hyperplane $H^{(\tau)}$ in the dissimilarity space $D^{(\tau)}(T, R)$. The distances to the hyperplane are realized by weighted linear combinations of the form $d_H^{(\tau)}(t_i) = \sum_{w_j^{(\tau)} \neq 0} w_j^{(\tau)} D^{(\tau)}(t_i, p_j) - \rho$. As a result, one may construct an $n \times K$ dissimilarity matrix $D_H = [d_H^{(1)}(T), \ldots, d_H^{(K)}(T)]$ expressing the non-normalized signed distances between the n training objects and K 'base' classifiers. Hence, again an OCC can be trained on D_H. This means that an OCC becomes a *trained combiner* now, re-trained by using the same training set (ideally, an additional validation set should be used). The LPDD can be used again, as well as some other feature-based OCCs. (Although the values of D_H become negative for the targets and positive for the outliers, they are bounded, so the LPDD can be constructed.) Additionally, two other standard data descriptions (OCCs) are used, where a proximity of an object to the target class relies on the *k*-mean information or density estimation by the Parzen kernels [13], respectively (the appropriate thresholds are set up as described in section 2).

3 Experiments and Results

The data consist of autofluorescence spectra acquired from healthy (target) and diseased (outlier) mucosa in the oral cavity [11,16]. The measurements were taken at 11 different anatomical locations using six excitation wavelengths 365, 385, 405, 420, 435 and 450 nm. We will denote them by $v_1 - v_6$. After preprocessing [16], each spectrum consists of 199 bins. In total, 856 and 132 spectra representing healthy and diseased tissue, respectively, are given for each wavelength. The spectra are normalized to have a unit area; see also Fig. 2. Two cases are here investigated: combining various DRs for a fixed wavelength of 365 nm

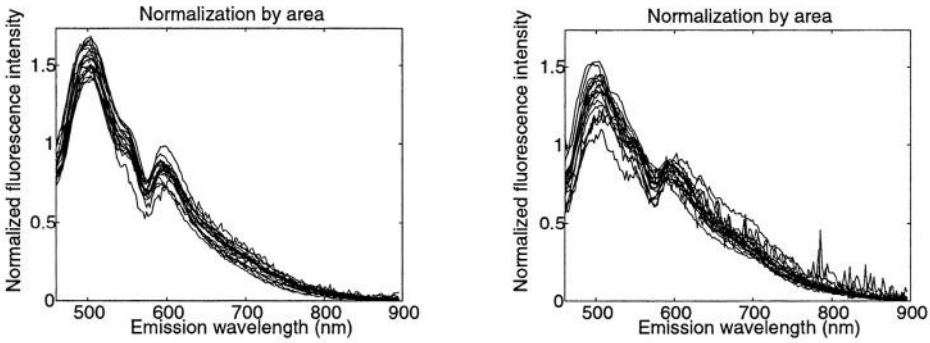


Fig. 2. Examples of normalized autofluoresence spectra for healthy (left) and diseased (right) patients for the excitation wavelength of 365 nm.

(experiment I) and combining representations derived for all the wavelengths (experiment II).

The objects are 30 times randomly split into the training set T and the test set S in the ratio of 60% : 40%, respectively. R, $R \subset T$, consists of the targets only, while T contains additional outliers. $|R| = 514$, $|T| = 594$ and $|S| = 394$ (337/57 healthy/diseased patients). If an OCC cannot use outlier information in the training stage, then it relies on $D^{(\tau)}(R, R)$ only. In the testing stage, $D^{(\tau)}(S, R)$ are used. Since we want to combine the representations directly, they should have a similar range. This is achieved by scaling all the initial $D^{(\tau)}$ by the maximal value of $D^{(\tau)}$ determined on the training data. So, further on, $D^{(\tau)}$ are assumed to be scaled appropriately. The LPDD is trained with $\nu = 0.05$ and the 3-NNDD and the GMDD use the threshold of 0.05. If the LPDD is trained using outlier information, it is denoted as $\mathcal{C}^{\text{out}}_{\text{LPDD}}$, otherwise, as $\mathcal{C}_{\text{LPDD}}$. Trained combiners use the zero threshold. All the experiments are done using DD-Tools [13] and PRTools [3].

Five dissimilarity representations $D^{(1)} - D^{(5)}$ are considered for the normalized spectra in experiment I (wavelength 365 nm). The first three DRs are based on the l_1 (city block) distances computed between the smoothed spectra themselves ($D^{(1)}$) and their first and the second order Gaussian-smoothed ($\sigma = 3$ samples) derivatives ($D^{(2)}$ and $D^{(3)}$, respectively). The zero-crossings of the derivatives indicate the peaks and valleys of the spectra, so they are informative. The differences between the spectra focus on the overlap, the differences in first derivatives emphasize the locations of peaks and valleys, while the differences in second derivatives indicate the tempo of changes in spectra. $D^{(4)}$ is based on the spherical geodesic distance $d_{(4)}(\boldsymbol{x}, \boldsymbol{y}) = r \arccos(\boldsymbol{x}^T \boldsymbol{y})/1^2$. $D^{(5)}$ is based on the Bhattacharyya distance, a divergence measure between two probability distributions. This measure is applicable, since the normalized spectra, say, s_i, can be considered as unidimensional histogram-like distributions. They are constant on disjoint intervals $I_1, \ldots, I_N$, such that $s_i(x) = \sum_{z=1}^{N} h_z^i \mathcal{I}(x \in I_z)$, where $h_z^i \geq 0$. The Bhattacharyya distance [4] is then: $d_{(5)}(s_i, s_j) = -\log\left(\sum_{z=1}^{N} (h_z^i h_z^j)^{1/2} |I_z|\right)$.

Table 1. Experiment I: the AUC performances (in %), averaged over 30 runs, of OCCs built either on the combined DRs or fixed and trained combiners applied to the OCCs outputs. All DRs are considered for the excitation wavelength of 365 nm. SO denotes support objects. The standard deviations of the means are in parenthesis.

Single DRs: OCCs trained on $D^{(\tau)}$						
DR	$\mathcal{C}_{3-\text{NNDD}}$ (1)	$\mathcal{C}_{\text{GMDD}}$ (2)	$\mathcal{C}_{\text{LPDD}}$ (3)	#SO	$\mathcal{C}^{\text{out}}_{\text{LPDD}}$ (3)	#SO
$D^{(1)}$	80.9 (0.5)	77.0 (0.6)	72.3 (0.7)	2.5	79.6 (0.5)	5.5
$D^{(2)}$	86.0 (0.4)	78.4 (0.5)	72.0 (0.7)	2.8	83.1 (0.5)	5.8
$D^{(3)}$	86.7 (0.4)	78.1 (0.6)	78.1 (0.7)	2.9	84.2 (0.5)	5.3
$D^{(4)}$	81.8 (0.5)	76.6 (0.6)	68.0 (0.9)	2.9	80.2 (0.5)	6.1
$D^{(5)}$	85.5 (0.4)	77.3 (0.5)	75.1 (0.6)	2.1	80.1 (0.5)	2.5
Combined DRs: OCCs trained on D_{comb} ($D^{(1)}-D^{(5)}$)						
D_{comb}	$\mathcal{C}_{3-\text{NNDD}}$ (1)	$\mathcal{C}_{\text{GMDD}}$ (2)	$\mathcal{C}_{\text{LPDD}}$ (3)	#SO	$\mathcal{C}^{\text{out}}_{\text{LPDD}}$ (3)	#SO
Avr	95.5 (0.2)	94.6 (0.3)	93.0 (0.3)	4.1	93.4 (0.3)	5.1
Prod	95.7 (0.2)	94.9 (0.3)	93.6 (0.3)	4.6	93.6 (0.4)	7.6
Min	85.6 (0.4)	84.6 (0.4)	84.7 (0.5)	14.6	87.1 (0.9)	15.7
Max	93.5 (0.3)	90.6 (0.4)	84.7 (0.8)	7.1	89.0 (0.6)	10.5
Fixed combiners built on the OCCs outputs from $D^{(1)}-D^{(5)}$						
Combiner	$\mathcal{C}_{3-\text{NNDD}}$ (1)	$\mathcal{C}_{\text{GMDD}}$ (2)	$\mathcal{C}_{\text{LPDD}}$ (3)		$\mathcal{C}^{\text{out}}_{\text{LPDD}}$ (3)	
Mean	98.0 (0.2)	94.4 (0.4)	90.7 (0.6)	—	93.8 (0.3)	—
Prod	98.0 (0.1)	81.3 (0.6)	87.8 (0.5)	—	91.1 (0.3)	—
Voting	98.3 (0.1)	95.9 (0.2)	95.5 (0.2)	—	97.0 (0.2)	—
Trained combiners built on the LPDDs outputs from $D^{(1)}-D^{(5)}$						
Combiner	$\mathcal{C}_{3-\text{NNDD}}$ (1)	$\mathcal{C}_{\text{GMDD}}$ (2)	$\mathcal{C}_{\text{LPDD}}$ (3)	#SO	$\mathcal{C}^{\text{out}}_{\text{LPDD}}$ (3)	#SO
LPDD	—	—	90.1 (0.5)	4.9	95.8 (0.2)	5.0
5-means	—	—	88.0 (0.4)	—	91.1 (0.4)	—
Parzen	—	—	90.5 (0.4)	—	94.5 (0.3)	—

In experiment II, DRs are derived for all excitation wavelengths. The first three measures $D^{(1)} - D^{(3)}$ are used. For each measure, six DRs are combined over the excitation wavelength $v_1 - v_6$ and, in the end, all 18 DRs are combined, as well.

Fixed combiners are also built on the outputs of single OCCs (the outputs need to be converted to posterior probabilities, e.g. as in [14]). Additionally, trained OCC combiners are constructed on the outputs of single LPDDs. The trained combiners are the LPDD and the k-means and Parzen data descriptions [13].

The following observations can be made from experiment I; see Table 1. Both an OCC trained on the combined representations and a trained or fixed combiner on the OCCs outputs improve the AUC performance of each single OCC trained on $D^{(\tau)}$. Concerning the combined representations, the element-wise average and product combiners perform better than the min and max operators. The 3-NNDD seems to give the best results; they are somewhat better than the ones obtained from the GMDD and and the LPDD trained on $D_{\text{comb}}(T, R)$. However, in the testing stage, both the 3-NNDD and the GMDD rely on computing

dissimilarities to all 514 objects of R, while the LPDD is based on maximum 16 support objects (see *#SO* in Table 1; the SO are determined during training). Hence, if some outliers are available for training, the LPDD can be recommended from the efficiency point of view. The fixed and trained combiners on the OCCs outputs perform well. In fact, the best overall results are reached for the fixed majority voting combiner. However, combiners require more computations; first five OCCs are trained on each $D^{(\tau)}$ separately and then, the final combiner is applied. Yet, if the LPDD $\mathcal{C}^{out}_{\mathrm{LPDD}}$ is used for training, then the testing stage is cheap: the dissimilarities to 27 objects have to be computed (sum of the support objects for single representations).

Due to lack of space, in Table 2 only some (the best) combining techniques are presented. Again, both an OCC trained on the combined representations (by the average and product) and a fixed or trained combiner on the OCCs outputs significantly improve the AUC performance (by more 10%) of each single OCC. By using all the six wavelengths and three dissimilarity measures (18 in total), all the combining procedures yield nearly perfect performances, i.e. mostly 99.5% or more. The trained combiners on the LPDDs outputs are somewhat worse (possibly due to overtraining) than the majority voting combiner, however, they are similar to the results of the mean combiner. Since the spectra derived from various wavelengths describe different information, an OCC built on their combined representation allows for reaching a somewhat better AUC performance than an OCC built on the DR combined for a single wavelength. From the computational point of view, either an LPDD trained on the combined DR or a fixed voting combiner on the LPDDs outputs should be preferred.

4 Conclusions

Here we study procedures of detecting one-class phenomena based on a set of training examples, performed in an unknown or ill-defined context of alternative phenomena. Since a proximity of an object to a class is essential for such a detection, dissimilarity representations (DRs) can be used as the ones which focus on object-to-target dissimilarities. The discriminative properties of various representations can be enhanced by a proper combining. Three different one-class classifiers (OCCs) are used: the NNDD (based on the nearest neighbor information), the GNMD (a generalized mean classifier in an underlying pseudo-Euclidean space) and the LPDD (a hyperplane in the corresponding dissimilarity space), which offers a sparse solution.

DRs directly encode evidences for objects which lie in close or far neighborhoods of the target objects. Hence, they can naturally be combined (after a proper scaling) into one representation, e.g. by an element-wise averaging. This is beneficial, since only one OCC can be trained, ultimately. From our study on the detection of diseased mucosa in oral cavity, it follows that DRs combined by average or product have a larger discriminative power than any single one. We also conclude that by combining information of DRs derived for spectra of different excitation wavelengths is somewhat more beneficial than by using only

Table 2. Experiment II: the AUC performances (in %), averaged over 30 runs. **Single DRs:** single OCCs built on DRs for six excitation wavelengths (only the worst and the best AUCs; $|\#SO| = 2-7$ for the LPDD). **Combined DRs:** OCCs built on the D_{comb} combined over six wavelengths and fixed $D^{(\tau)}$. **Fixed combiners:** fixed rules applied to the outputs of the trained OCCs and **trained combiners:** combiners trained on the outputs of the LPDDs, both combined over six wavelengths. 'ALL' refers to the results on all 6×3 (six wavelengths and three measures) DRs. SO denotes support objects.

Single DRs: OCCs trained on $D^{(\tau)}$ for different v_i								
	$D^{(1)}$		$D^{(2)}$		$D^{(3)}$		ALL	
$\mathcal{C}_{3-\text{NNDD}}$	80.9 - 84.8 (0.5)		82.8 - 87.0 (0.5)		83.5 - 88.8 (0.5)		80.9 - 88.8 (0.5)	
$\mathcal{C}_{\text{GMDD}}$	77.0 - 79.4 (0.7)		77.9 - 81.7 (0.6)		75.4 - 81.6 (0.6)		75.4 - 81.7 (0.7)	
$\mathcal{C}_{\text{LPDD}}$	62.8 - 72.4 (0.8)		65.5 - 72.8 (0.8)		70.7 - 77.5 (0.8)		62.8 - 77.5 (0.8)	
$\mathcal{C}^{\text{out}}_{\text{LPDD}}$	78.3 - 81.7 (0.9)		73.5 - 83.1 (0.7)		77.7 - 83.2 (0.6)		73.5 - 83.2 (0.6)	
Combined DRs: OCCs trained on D_{comb} combined over $v_1 - v_6$								
$\mathcal{C}_{3-\text{NNDD}}$, D_{comb}	$D^{(1)}$		$D^{(2)}$		$D^{(3)}$		ALL	
Avr	97.7 (0.2)	—	97.6 (0.2)	—	96.8 (0.1)	—	99.6 (0.0)	—
Prod	97.7 (0.2)	—	97.7 (0.2)	—	96.9 (0.1)	—	99.7 (0.0)	—
$\mathcal{C}_{\text{GMDD}}$, D_{comb}	$D^{(1)}$		$D^{(2)}$		$D^{(3)}$		ALL	
Avr	97.2 (0.2)	—	97.2 (0.2)	—	96.0 (0.1)	—	99.6 (0.0)	—
Prod	97.3 (0.2)	—	97.4 (0.2)	—	96.3 (0.1)	—	99.6 (0.0)	—
$\mathcal{C}_{\text{LPDD}}$, D_{comb}	$D^{(1)}$	#SO	$D^{(2)}$	#SO	$D^{(3)}$	#SO	ALL	#SO
Avr	96.6 (0.3)	5.2	97.1 (0.3)	4.2	95.6 (0.2)	3.6	99.5 (0.1)	4.3
Prod	96.9 (0.2)	5.7	97.2 (0.3)	4.0	95.8 (0.2)	3.7	99.6 (0.0)	4.9
$\mathcal{C}^{\text{out}}_{\text{LPDD}}$, D_{comb}	$D^{(1)}$	#SO	$D^{(2)}$	#SO	$D^{(3)}$	#SO	ALL	#SO
Avr	96.7 (0.1)	5.1	97.1 (0.1)	4.0	95.6 (0.1)	3.6	99.5 (0.0)	4.5
Prod	96.8 (0.1)	7.3	97.2 (0.2)	5.8	95.8 (0.1)	5.0	99.6 (0.1)	6.6
Fixed combiners applied to the OCCs outputs								
$\mathcal{C}_{3-\text{NNDD}}$ outputs	$D^{(1)}$		$D^{(2)}$		$D^{(3)}$		ALL	
Mean	97.8 (0.1)	—	98.0 (0.1)	—	98.2 (0.2)	—	99.6 (0.1)	—
Prod	98.6 (0.1)	—	98.5 (0.1)	—	98.6 (0.1)	—	99.6 (0.0)	—
Voting	97.6 (0.1)	—	98.7 (0.1)	—	98.6 (0.1)	—	99.8 (0.0)	—
$\mathcal{C}_{\text{GMDD}}$ outputs	$D^{(1)}$		$D^{(2)}$		$D^{(3)}$		ALL	
Mean	94.3 (0.4)	—	94.2 (0.3)	—	94.3 (0.3)	—	98.3 (0.2)	—
Prod	96.0 (0.2)	—	96.4 (0.1)	—	96.7 (0.1)	—	99.7 (0.0)	—
Voting	96.7 (0.2)	—	97.4 (0.1)	—	97.6 (0.1)	—	99.6 (0.1)	—
Fixed and trained combiners applied to the $\mathcal{C}_{\text{LPDD}}$ outputs								
Combiner	$D^{(1)}$	#SO	$D^{(2)}$	#SO	$D^{(3)}$	#SO	ALL	#SO
Mean	92.7 (0.4)	—	92.9 (0.4)	—	91.8 (0.3)	—	94.5 (0.2)	—
Prod	95.7 (0.9)	—	95.7 (1.0)	—	95.7 (0.5)	—	98.7 (0.6)	—
Voting	95.7 (0.4)	—	96.8 (0.2)	—	97.9 (0.1)	—	99.3 (0.1)	—
LPDD	89.3 (0.4)	5.9	91.5 (0.4)	5.9	94.6 (0.2)	5.9	96.6 (0.3)	13.2
Parzen	92.1 (0.3)	—	94.4 (0.3)	—	94.9 (0.3)	—	98.2 (0.1)	—
Fixed and trained combiners applied to the $\mathcal{C}^{\text{out}}_{\text{LPDD}}$ outputs								
Combiner	$D^{(1)}$	#SO	$D^{(2)}$	#SO	$D^{(3)}$	#SO	ALL	#SO
Mean	93.7 (0.4)	—	93.6 (0.5)	—	95.6 (0.4)	—	98.8 (0.3)	—
Prod	95.4 (0.8)	—	96.2 (0.9)	—	97.2 (0.5)	—	99.5 (0.6)	—
Voting	96.3 (0.4)	—	96.8 (0.2)	—	98.0 (0.1)	—	99.5 (0.1)	—
LPDD	95.7 (0.2)	6.0	96.5 (0.2)	6.0	95.8 (0.2)	6.0	99.1 (0.1)	16.3
Parzen	95.5 (0.2)	—	96.8 (0.2)	—	96.2 (0.2)	—	98.9 (0.1)	—

one fixed wavelength, yet different dissimilarity measures. In the former case, all the OCCs on the combined representations performed about the same, while in the latter case, the LPDD trained on the targets seemed to be worse. The fixed OCC combiners have also been applied to the outputs of single OCCs. The overall best results are reached for the majority voting rule. The trained OCC combiners, applied to the outputs of single LPDDs, performed well, yet worse than the voting rule. Concerning the computational issues, either the LPDD on the combined representations should be used or the majority voting combiner applied to the LPDDs outputs.

Further studies on new problems need to be conducted in the future.

Acknowledgments

This work is supported by the Dutch Organization for Scientific Research (NWO) and the Dutch Organization for Applied Research (STW), which supplied the data [16].

References

1. A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition,* 30(7): 1145–1159, 1997.
2. P.S. Bradley, O.L. Mangasarian, and W.N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing,* 10:209–217, 1998.
3. R.P.W. Duin, P. Juszczak, de D. Ridder, P. Paclík, E. Pękalska, and D. Tax. PR-Tools, a Matlab toolbox for pattern recognition, 2004.
4. F. Esposito, D. Malerba, V. Tamma, H.H. Bock, and F.A. Lisi. *Analysis of Symbolic Data,* chapter Similarity and Dissimilarity. Springer-Verlag, 2000.
5. L. Goldfarb. A new approach to pattern recognition. In *Progress in Pattern Recognition,* volume 2, pages 241–402. Elsevier Science Publishers B.V., 1985.
6. D.W. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with non-metric distances: Image retrieval and class representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 22(6):583–600, 2000.
7. E. Pękalska. *working title: Dissimilarity-based pattern recognition.* PhD thesis, Delft University of Technology, The Netherlands, expected in 2004.
8. E. Pękalska, P. Paclík, and R.P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research,* 2(2):175–211, 2001.
9. E. Pękalska, D.M.J. Tax, and R.P.W. Duin. One-class LP classifier for dissimilarity representations. In *NIPS,* pages 761–768. MIT Press, Cambridge, MA, 2003.
10. B. Schölkopf, J.C. Platt, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation,* 13:1443–1471, 2001.
11. M. Skurichina and R.P.W. Duin. Combining different normalizations in lesion diagnostics. In *Supplementary Proc. ICANN/ICONIP,* pages 227–230, Turkey, 2003.
12. B.M.R. Stadler, P.F. Stadler, G.P. Wagner, and W. Fontana. The topology of the possible: Formal spaces underlying patterns of evolutionary change. *Journal of Theoretical Biology,* 213(2):241–274, 2001.

13. D.M.J. Tax. DD-Tools, a Matlab toolbox for data description, outlier and novelty detection, 2003.
14. D.M.J. Tax and R.P.W. Duin. Combining one-class classifiers. In *Multiple Classifier Systems, LNCS,* volume 2096, pages 299–308. Springer Verlag, 2001.
15. D.M.J. Tax and R.P.W. Duin. Support vector data description. *Machine Learning,* 54(1):45–56, 2002.
16. D.C.G. de Veld, M. Skurichina, M.J.H. Witjes, and et.al. Autofluorescence characteristics of healthy oral mucosa at different anatomical sites. *Lasers in Surgery and Medicine,* submitted, 2003.

A Modular System for the Classification of Time Series Data

Lei Chen, Mohamed Kamel, and Ju Jiang

Pattern Analysis and Machine Intelligence Lab
Systems Design Engineering
University of Waterloo, Ontario
N2L3G1, Canada

Abstract. While the field of classification is witnessing excellent achievement in recent years, not much attention is given to methods that deal with the time series data. In this paper, we propose a modular system for the classification of time series data. The proposed approach explores the diversity through various input representation techniques, each of which focuses on a certain aspect of the temporal patterns. The temporal patterns are identified by aggregation of the decisions of multiple classifiers trained through different representations of the input data. Several time series data sets are employed to examine the validity of the proposed approach. The results obtained from our experiments show that the performance of the proposed approach is effective as well as robust.

1 Introduction

Many real applications are interested in the knowledge varying over time. Currently, the temporal classification has been widely adopted in areas such as climate control research, medical diagnosis, economic forecast, sound recognition etc. However, while the classification of the non-temporal information has great achievement in recent years, temporal classification techniques are scarce. Temporal patterns contain dynamic information and are difficult to be represented by traditional approaches. Classifiers such as neural network, decision trees don't function well when applied to temporal data directly. Other solutions which employ temporal models are normally domain-dependent and could not be applied to general problems. In the current directions of the machine learning research, Multiple Classifier Systems(MCSs) have been proved to be an effective way to improve the classification performance and are widely used to achieve high pattern-recognition performances [9]. Currently, they have become one of the major focuses in this area. Generally, a MCS is composed of three modules: diversification, multiple decision-making and decision combination. Fig. 1 demonstrates a general conceptual framework for MCSs.

Multiple decision-making is normally regarded as a process in which the classifier searches correct answers in its knowledge space. The diversity module collectively covers the knowledge space available. In the decision combination module, multiple decisions are combined under different rules to make the final decisions. The use of MCSs may be an effective way for temporal classification problem. Although temporal patterns are often complex and difficult to identify, this task can be divided into several

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 134–143, 2004.

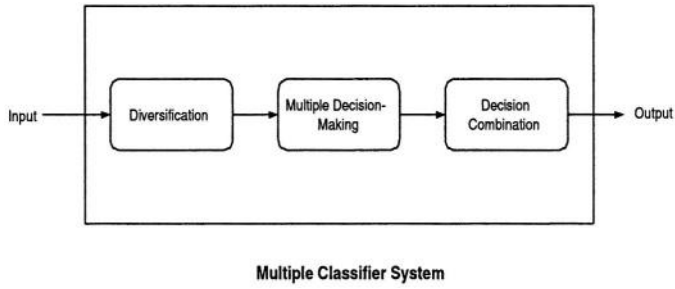


Fig. 1. Conceptual Framework of MCSs

simpler sub-tasks. Therefore, if it is possible to find an ensemble of classifiers, each of which effectively fulfills one of the sub-tasks, the temporal patterns could be well identified through the aggregation of solutions of the ensemble.

The organization of this paper is as follows: section 2 reviews the related work. In section 3, we propose a modular system to achieve diversity at the input level. The experiments and discussion are presented in section 4. Section 5 concludes the work of this paper.

2 Related Works

The issue of diversity has long been the focus of many research works on MCSs. Here we review work related to temporal data classification. Diversity at the training level is achieved by combining an ensemble of homogeneous or heterogeneous classifiers. Sanchos et. al.[14] employ an ensemble of five base classifiers which are either Self-Organizing Mapping(SOM) or Multiple Layer Perceptron(MLP) for the classification of the time series data. Each of the SOM or MLP shares the same architecture but has different initial weights. While it is quite innovative to combine the supervised and unsupervised learning in this research, it has several weaknesses. At first, the euclidian distance is used to calculate the posteriori probability of each class in the SOM classifiers. However, it is known that the euclidian distance is not very effective when measuring the similarity of the time-dependent data. Therefore, it is some questionable to adopt this method to generate the posteriori probability. In addition, the mapping between the cluster and the class label may not be the one to one mapping which is assumed implicitly by the author. Ghosh et. al.[6,8] employ an ensemble of Artificial Neural Networks(ANNs) with the weighted average fusion technique to identify and classify the underwater acoustic signals. The feature space of the input data is composed of a 25-dimensional feature vectors extracted from the raw signals. In the feature vectors, there are 16 wavelet coefficient, 1 value denoting signal duration and 8 other temporal descriptors and spectral measurements. One of the weaknesses for this research is that the feature extraction is *ad hoc*. The number and type of the features may be only optimal for certain applications such as the oceanic signal data.

The previous techniques mainly achieved the diversity of the MCSs at the training level by introducing the different type of classifiers or different sets of parameters.

The work of Valentini and Masulli [17] states the trade-off between accuracy and independence of the base classifiers. Therefore, the number of independent classifiers with high accuracy may be limited, which could limit the capability of MCSs to achieve the diversity at the training level. While the previous researchers were interested in combining the different classifiers, recent focus in this area has shifted to diversity at the input level. González et.al. [7] and Diez et.al. [3] propose interval based classification systems for time series data, in which the AdaBoost technique is adopted . The basic idea is that each time series data is represented by a set of temporal predicates and the final classification results are obtained based on the combination of these predicates. The objective of this research, as claimed by the author, is to find a non-domain specific temporal classification technique. However, the selection and definition of the temporal predicates may be *ad hoc* in themselves. In addition, it is also difficult to decide the parameters such as the intervals of the variables and the size of the searching window. Dietrich et. al. [5] propose three different kinds of architectures for the fusion of decisions based on these local features:Classification, Decision Fusion and Temporal Fusion(CDT), Decision Fusion, Classification and Temporal Fusion(DCT) and Classification, Temporal Fusion, Decision Fusion(CTD). CDT and CTD adopt the hierarchical fusion architecture. They differ from each other in first fusing the decisions of the base classifiers on different type of features in the same window or the same type of features over various windows. DCT first connects the p features in each window into a vector. Then, the final decision is made based on the fusion of the decisions of the base classifiers on the p-feature vector in each window. In [4], Dietrich et.al. further proposed three another architectures for fusion of decisions: Multiple Decision Template(MDT), Decision Template(DT) and Clustered Multiple Decision Template(CMDT). MDT and DT are similar with CDT and DCT respectively. CMDT improves MDT by assigning a set of multiple templates $\{DT_i^1, DT_i^2, ..., DT_i^k\}$ to each class w_i. The weakness of this research is concluded as following:(1) The decision of the parameters such as the size of the sliding window is data-dependent. (2) The sliding window method may cause information loss for high-order temporal patterns. Hsu et.al. [10] propose a hierarchical mixture model called specialist-moderator network for the time series classification. It combines recurrent ANN classifiers in a bottom-up architecture. The primary contribution of this work is the model's ability to identify the intermediate targets based on a partition of the input channels. One of the common drawbacks of the hierarchical classifier system like hierarchical mixture of experts is that it is not robust and the failure of one base classifier immediately affects the performance of the whole system.

3 Modular Approach

The validity of the diversity is based on the following two observations: (1) the ability of the individual classifier to identify the discriminating patterns is limited in some situations. (2) the set of the patterns identified by a classifier C_i depends on the representation of the input data. The representation technique is defined as a mapping :

$$f : D \rightarrow R \tag{1}$$

Where D is the input data space and R is the result data space, that is $R_j = f_j(D)$. Suppose that $Pattern_i(R_j)$ represents the pattern set identified by classifier C_i under

the representation technique $f_j(D)$ and *Pattern(D)* represents all the patterns in the data set *D*. Therefore, there is:

$$L_i^j = Pattern(D) - Pattern_i(R_j) \tag{2}$$

where L_i^j is the set of the patterns hidden from classifier C_i with the representation technique $f_j(D)$. Suppose each pattern is equally important. From equation 2, it is clear that the size of the set L_i^j should be minimized to achieve the optimal performance. Let L_C^R represents the set of the patterns hidden from the ensemble of classifiers $C = \{C_i\}$, in which the base classifier C_i adopts a certain representation technique R_j ($R = \{R_j\}$). Therefore, there is

$$L_C^R = \bigcap_{i=1}^{L} \bigcap_{j=1}^{M} L_i^j \tag{3}$$

and

$$L_C^R \subseteq L_i^j \tag{4}$$

where $i = 1..L, j = 1..M$. Equation 4 shows MCSs may be superior to any combination of the individual classifier and representation technique if the diversity is appropriately balanced. In this research, we mainly focus on examining the effects of the diversity achieved through different input representations. In this paper, a set of homogeneous classifiers *C* is employed with various representation techniques $R = \{ R_1, ...R_M \}$. Compared to resampling [2] and input decimation [12], there is less information loss when processing the input information. The decomposition of the input space reduces the information available for the individual classifiers. Thus, the classifiers may not be well trained. The decomposition of the feature space may conceal some high-order patterns, therefore degrades the performance of the whole system. While the diversity through input transformation [16] overcomes the drawbacks of the resampling and input decimation, the achieved diversity is limited. For example, the global information of the times series data such as the mean, variance is also important to the classification. The proposed approach allows various representation of input besides the transformation and tends to have a better performance. Compared to the non-temporal data, it is more difficult for the temporal patterns to be represented by individual classifiers in general. Therefore, the set L_i^j tends to be large and the performance of the individual classifiers may not be good in some situations. In the proposed approach, each Temporal Representation Processor(TRP) is an expert to a certain type of temporal information, which implements a certain representation technique, $f_j(D)$. The complete information in the temporal patterns could be well identified by aggregation of the classifiers trained with different representation of the input data. Most importantly, this strategy doesn't assume the comprehensiveness of any TRP for classifier C_i. That is, each TRP may just process only part of the temporal information. Thus, the complexity in the design of TRP is significantly reduced. Fig. 2 shows the architecture of the proposed approach.

In the proposed approach, each TRP maps the time series data *X*, ($X = \{X_t\}, t = \{1...T\}$), to another domain, *R*. The TRP is composed by a series of components, each of which implements a certain algorithm to process the time series data. Several representation techniques for the time series data are discussed in the following.

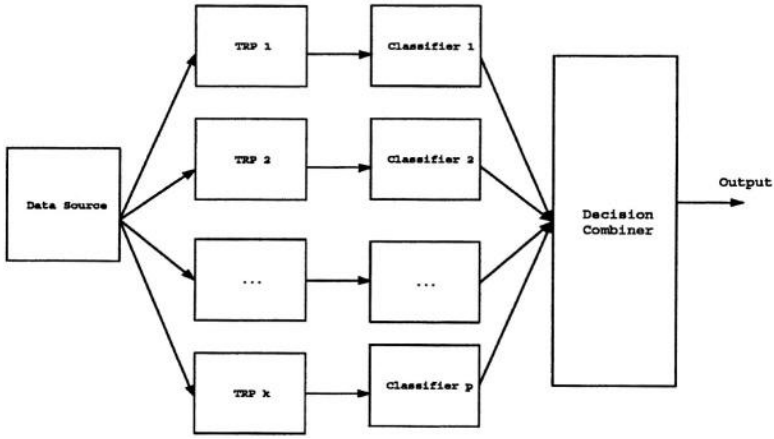


Fig. 2. Proposed Architecture

– Filter

The filter is employed to remove a certain type of information in the time series data. For example, the local fluctuations of the time series data should be removed if the objective is to obtain the smoothed values. In contrast, the long-term fluctuations should be removed to get the residuals from the smoothed value. There are various types of filters depending on their functional requirements. Among the popular schemes to produce a smoothed time series are the exponential smoothing scheme which weight past observations using exponentially decreasing weights. The family of the exponential smoothing schemes includes Single Exponential Smoothing(SES), Double Exponential Smoothing(DES) and Triple Exponential Smoothing(TES). The difference of these methods is the number of parameters which are used to represent the trend and seasonal information in the time series data. There is no parameters to represent the trend and seasonal information in SES. While the trend information is given into consideration in DES, the seasonal information is just neglected. The TES is the most complex one. It employs a set of parameters to represent both the trend and seasonal information. As an example, the function of DES is given as:

$$S_t = \alpha X_t + (1-\alpha)(S_{t-1} + b_{t-1})$$
$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1}$$
$$b_0 = \frac{(X_2 - X_1) + (X_3 - X_2) + (X_4 - X_3)}{3}$$

where S_i is the estimated value space, X_i is the input time series data. α is the smoothing rate which are in the interval [0,1]. When $\alpha \to 1$, the smoothed data is exactly the same as the original one. γ is the learning rate. The larger the value of γ is, the more the estimated data is affected by the previous experience. Obviously, different combinations of the parameters generate different series of temporal data. The DES(α, γ) stands for the algorithm with the parameters α and γ.

- Spectral Analyst
  Spectral analysis is a useful exploratory diagnostic tool in the analysis of many types of the time series data. It provides the information of the 'hidden periodicity' in the data. The most widely used technique in the spectral analysis is the Discrete Fourier Transformation(DFT). The effects of the transformation is to map the data in the time domain to the frequency domain. The transformed data could be directly fed into the traditional classifiers. Other spectral analysis methods that are also used quite often include the Wavelet transform, Hilbert transform, short-time Fourier transform.
- Differentiation Generator(DG)
  The differentiation generator is a special type of filter which calculates the differentiation within a temporal channel. A differentiation generator $DG(p,q)$ has two parameters: *order* and *distance,* which are represented by p, q respectively. For the first order and one distance differentiation generator, it is calculated as:

$$DG(1,1) = X_t - X_{t-1} \tag{5}$$

  Similarly, the p order and q distance differentiation generator is calculated by:

$$DG(p,q) = DG(p-1, q+t) - DG(p-1, t) \tag{6}$$

  The differentiation generator has several functionalities. At first, it could remove the effects of the initial values. For example, the effects of the shift could be eliminated by DG(1,1). In addition, it could remove the trend information in the time series data by differentiating the given temporal channel until it becomes stationary. This is helpful to those TRPs which only focus on the stationary information in the time series data.

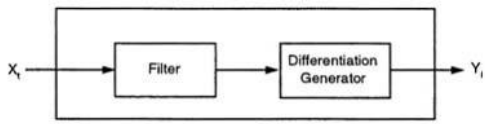


Fig. 3. Temporal Representation Processor

The components in TRP could be connected either sequentially or in parallel to fulfill a certain task. The design of TRPs depends on the representation logic. For example, if the functionality of a certain TRP is to "smooth the time series data and eliminate the 'shift' effects", the task is fulfilled by the sequential connection of the filter and differentiation generator (Fig. 3). The connection mechanism of the components further expands the ability of TRPs to process the temporal information.

4 Experimental Results and Discussion

In this section, we use several popular time series data sets to demonstrate the feasibility of the proposed approach. These data sets are used by previous researchers and

Table 1. Characteristics of the Data Sets

Data Set	Classes	Instances	Frames
CBF	3	600	200
Control Chart(CC)	6	600	60
CC+Label Noise	6	600	60
Waveform(WF)	3	600	21
WF+Data Noise	3	600	40

available from the UCI repository or related references. The characteristics of the data is summarized in table 1.

- The CBF data is introduced by Saito[13] as an artificial problem. The learning task is to distinguish the data from three classes:Cylinder(c), Bell(b) and Funnel(f). The models of the three classes are:

$$c(t) = (6+\eta)\cdot\chi_{[a,b]}(t) + \varepsilon(t)$$
$$b(t) = (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\frac{t-a}{b-a} + \varepsilon(t)$$
$$f(t) = (6+\eta)\cdot\chi_{[a,b]}(t)\cdot\frac{b-t}{b-a} + \varepsilon(t)$$

  η and $\epsilon(t)$ are obtained from the standard normal distribution $N(0,1)$. a is an integer obtained from the uniform distribution in [16,32]. c is another integer obtained from the uniform distribution in [32,96]. The integer b is the sum of a and c. $\chi_{[a,b]} = 1$ if $t \in [a,b]$. Otherwise, $\chi_{[a,b]} = 0$.
- The CC data(from UCI repository) contains 600 examples of control charts which were synthetically generated. There are six different classes of control charts: (A) Normal (B) Cyclic (C) Downward shift (D) Upward shift (E) Increasing trend (F) Decreasing trend.
- The data set Control Chart+Label Noise(CC+LN) is synthetically produced in this research to examine the performance of the proposed approach in the presence of label noise in the data. This data is generated in the same way as the control chart data except that the label of the training data is randomly generated. That is, there is a subset of training data that may not be correctly labelled.
- The Waveform was introduced by Breiman et al.[1]. The purpose is to distinguish between three classes, defined by the evaluation in the time frames $t = 1, 2...21$, of the following models:

$$x_1(t) = uh_1(t) + (1-u)h_2(t) + \varepsilon(t)$$
$$x_2(t) = uh_1(t) + (1-u)h_3(t) + \varepsilon(t)$$
$$x_3(t) = uh_2(t) + (1-u)h_3(t) + \varepsilon(t)$$

  where $h_1(t) = max(6 - |i-7|, 0)$, $h_2(t) = h_1(t-8)$, $h_3(t) = h_1(t-4)$. u is a random variable with uniform distribution in (0,1) and $\varepsilon(t)$ is the standard normal distribution.

- The Waveform+Data Noise(WF+DN)[7] is generated in the same way as the previous models except that 19 frames are added to the end of each time series. The value of the 19 frames follows the standard normal distribution $N(0,1)$. This data set is used to test the performance of the proposed approach in the presence of the data noise in the data set.

The experiments are designed to examine the accuracy as well as the reliability of the proposed approach. The single classifier is employed as the benchmark to evaluate the performance of the MCSs. *Raw, DFT and DES+DG* represent the single Probabilistic Neural Network(PNN) [15] classifier with raw time series data, the data with the DFT and the data processed with the DES and DG respectively. The test results are shown in table 2, which summarizes the Mean of Correct Ratio(MCR) and Varaiance of Correct Ratio(VCR) of the different approaches from 10 continuous experiments. The entry is in the format of $\mathbf{MCR\%(VCR\times 10^{-3})}$. For the Bagging approach, an ensemble of 12 homogeneous PNN classifiers are employed. We adopt the bootstrapping resampling technique in this experiment. Each base classifier is trained with 80 % of the training data which are randomly selected from the training data set.

Table 2. Experimental Results

Data Set	CBF	CC	CC+5%	CC+10%	WF	WF+DN
Raw	72.5(1.0)	81.0(1.8)	81.8(1.4)	77.9(2.3)	93.7(0.3)	90.5(0.2)
DFT	84.8(0.5)	72.8(2.2)	73.5(1.3)	70.2(4.1)	86.1(0.2)	83.2(0.9)
DES+DG	77.0(0.4)	85.8(0.4)	84.2(0.4)	80.6(1.1)	92.2(0.4)	88.4(0.2)
Bagging	72.2(0.8)	80.6(1.7)	79.5(0.9)	80.3(1.2)	93.8(0.1)	89.4(0.2)
NTRPs-BF	71.3(0.7)	84.0(5.3)	78.0(2.1)	78.0(2.2)	91.7(0.5)	89.7(0.3)
TRPs-BF	85.5(4.0)	88.6(1.8)	88.4(1.7)	83.7(5.7)	92.9(0.3)	90.7(0.2)
TRPs-DT	91.0(0.8)	92.2(0.4)	91.3(0.8)	90.5(0.6)	92.9(0.4)	90.8(0.7)

We examine the proposed diversity approach with different fusion techniques including the Bayesian Fusion(TRPs-BF) and Decision Template(TRPs-DT) approaches [11]. For the these experiments, a MCS with 12 homogeneous PNN base classifiers are employed, which are trained with different representation of the time series data. One of the TRPs is composed of 1 spectral analyst which implements the DFT algorithm. Another TRP implements a dummy function which just presents the original data to the classifier. Other TRPs are composed of a filter, which implements the DES and DG sequentially. 10 different TRPs are generated by using different parameters of the DES. Finally, we also examine the effects of diversity at the training level. The tuple *NTRPs-BF* in table 2 stands for the MCS which is combined with the bayesian fusion but has no diversity at the input level. It employs a homogeneous set of 12 PNNs with the raw data representation. In all of the experiments, the data sets are randomly separated into 60% training and 40% testing. For the trained fusion techniques including the DT and BF etc., 66.7% of the training data are used to train the PNN classifier and the remaining 33.3% used as a validation set to estimate the decision distribution of the base classifiers.

The experiment results are divided into three groups for discussion: without noise (CBF,CC,WF), with label noise (CC,CC+5% CC+10%) and with data noise (WF, WF+Data Noise). Both the CC and WF data sets are included in two different groups to show the trend of the performance of various approaches in the presence of the noise. There are several observations from the performance of the individual classifier approaches including the Raw, DFT and DES+DG. (1) The first group of experiments show that these techniques are data-dependent. For example, the highest MCR for Raw is 93.7% in the WF data set while it is as low as 72.5% in CBF. (2) Different representation techniques have strength in different data sets. For the CBF, the MCR of DFT is 12.3% higher than the Raw and 7.8% higher than the DES+DG. For the CC and WF, DES+DG and Raw has the best performance respectively. This provides a foundation for the proposed MCSs to achieve consistent performance over various data sets. (3) Different techniques present different level of robustness in the presence of noise. The second group of experiments show that DES+DG is relatively sensitive to label noise while the RAW and DFT are more robust. Therefore, it is possible for the MCS to achieve robust performance by combining the different methods. For the resampling technique, the performance of Bagging is similar to the Raw. This demonstrates that the improvement with the resampling technique on the temporal classification may be limited. The most interesting comparison is between the NTRPs-BF and the TRPs-BF since these two approaches have the same multiple-decision and decision combination modules, but TRPs-BF achieves the diversity at the input level through different representations. We found that the TRPs-BF outperforms the NTRPs-BF in all of the three groups of experiments. In particular, the MCR of TRPs-BF is 13.8 % more than the NTRPs-BF in CBF. It shows that the diversity through various input representation could be an effective way for temporal classification. The TRPs-DT is even much better than the TRPs-BF. In particular, TRPs-DT appears to be more robust to label and data noise. For example, when there are 10% of label noise in the CC data, its MCR only drops 1.7% compared to the one without label noise. In addition, the TRPs-DT also significantly outperforms other methods which are previously discussed in general.

5 Conclusion

In this paper, we propose a modular system for the classification of the time series data in which the diversity through various input representations is employed. Since the patterns identified by the individual classifiers depend on the representation of the input data, it is possible that the performance of MCSs could be improved by aggregation of the different discriminating patterns. In particular, this approach could be effective for temporal classification since the temporal patterns contain dynamic information and are difficult to be fully represented by an individual technique. The experimental results show that the proposed approach significantly outperforms other methods in general.

Acknowledgement

This research is partially supported by Natural Sciences and Engineering Research Council of Canada.

References

1. Breiman, L., Friedman, J.H., Olshen, A., Stone, C.J.:Classification and regression trees. Chapman and Hall, New York, 1993
2. Duda, Richard O., Hart, Peter E., Stork, David G.: Pattern classification. 2nd Edition, Published by Wiely-Interscience, December 28, 2000
3. Diez, J. J. R., González, C. J. A.: Applying boosting to similarity literals for time series classification. MCS2000 LNCS **1857** (2000) 210–219
4. Dietrich, C. , Palm, G., Schwenker, F.: Decision templates for the classification of bioacoustic time series. Proceedings of IEEE Workshop on Neural Networks and Signal Processing (2002) 159–168
5. Dietrich, C., Schwenker, F. , Palm, G.: Classification of time series utilizing temporal and decision fusion. MCS2001 LNCS **2096** (2001) 378–387
6. Ghosh, J., Beck, S., Chu, C.C.: Evidence combination techniques for robust classification of short-duration oceanic signals. In SPIE Conf. on Adaptive and Learning Systems, SPIE Proc. **1706** (1992) 266–276
7. González, C. J. A., Diez, J, R.:Time series classification by boosting interval based literals. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial **11** (2000) 2–11
8. Ghosh, J., Deuser, L., Beck, S.: A neural network based hybrid system for detection,characterization and classification of short-duration oceanic signals. IEEE Journal of Ocean Engineering **17**(4) (1992) 351–363
9. Giancinto, G., Roli, F.: Dynamic classifier selection based on multiple classifier behaviour. Pattern Recognition **34**(9) (2001) 1879–1881
10. Hsu, William H., Ray, Sylvian R.: Construction of recurrent mixture models for time series classification. In Proceedings of the International Joint Conference on Neural Networks **3** (1999)1574–1579
11. Kuncheva, L., Bezdek, J., Duin, R.: Decision templates for multiple classifier fusion: an experimental comparision, Pattern Recognition, Vol.34, 2001, pp.299–314
12. Oza, Nikunj C., Tumer,K.: Input decimation ensembles: decorrelation through dimensionality reduction. MCS2001 LNCS **2096** (2001) 238–247
13. Naoki Saito: Local feature extraction and its applications using a library of bases. Phd thesis, Department of Mathematics, Yale University, 1994
14. Sancho, Q. Isaac Moro, Alonso, C., Rodrìguez, J. J.: Applying simple combining techniques with artificial neural networks to some standard time series classification problems. Artificial Neural Networks in Pattern Recognition (2001) 43–50
15. Specht, D.F.:Probabilistic neural networks, Neural Networks **3** (1990) 109–118.
16. Sirlantzis, S.Hoque, Fairhurst, M.C.: Input space transformation for multi-classifier systems based on n-tuple classifiers with application to handwriting recognition. MCS2003 LNCS **2709** (2003) 356–365
17. Valentini, G., Masulli, F.: Ensembles of learning machines. Neural Nets WIRN Vietri-2002, **2486** (2002) 3–19

A Probabilistic Model Using Information Theoretic Measures for Cluster Ensembles

Hanan Ayad, Otman Basir, and Mohamed Kamel

Pattern Analysis and Machine Intelligence Lab
Systems Design Engineering, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{hanan,mkamel}@pami.uwaterloo.ca
http://pami.uwaterloo.ca/

Abstract. This paper presents a probabilistic model for combining cluster ensembles utilizing information theoretic measures. Starting from a co-association matrix which summarizes the ensemble, we extract a set of *association distributions,* which are modelled as discrete probability distributions of the object labels, conditional on each data object. The key objectives are, first, to model the associations of neighboring data objects, and second, to allow for the manipulation of the defined probability distributions using statistical and information theoretic means. A Jensen-Shannon Divergence based Clustering Combination (JSDCC) method is proposed. The method selects cluster prototypes from the set of association distributions based on entropy maximization and maximization of the generalized JS divergence among the selected prototypes. The method proceeds by grouping association distributions by minimizing their JS divergences to the selected prototypes. By aggregating the grouped association distributions, we can represent empirical cluster conditional probability distributions of the object labels, for each of the combined clusters. Finally, data objects are assigned to their most likely clusters, and their cluster assignment probabilities are estimated. Experiments are performed to assess the presented method and compare its performance with other alternative co-association based methods.

1 Introduction

Unsupervised classification, or data clustering is an essential tool for exploring and searching for groups in unlabelled data. It is a challenging problem because the clusters inherent in the data can be of arbitrarily different shapes and sizes. A large number of clustering techniques have been developed over the years [1,2]. However, many are limited to rinding clusters of specific shapes and structures and may fail when the data reveal cluster shapes and structures that do not match their assumed model. For instance, the K-means which is one of the simplest and computationally efficient clustering technique, can easily fail if the true clusters inherent in the data are not hyper-spherically shaped.

Recently, there has been an emergent interest in studying cluster ensembles to enhance the quality and robustness of data clustering and to accommodate a

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 144–153, 2004.

wider variety of data types and clusters structures [3–10]. Some of the research have relied on using a co-association matrix as a voting medium for finding the combined partitioning. Fred and Jain [5], used single link (SLink) hierarchical clustering to produce a final partitioning of the data. Cluster-based Similarity Partitioning Algorithm (CSPA) is a consensus function introduced by Strehl and Ghosh [3] in which graph partitioning is applied to the co-association matrix resulting in a consensus partitioning. In [8,9], we used the co-association matrix to construct a Weighted Shared nearest neighbors Graph (WSnnG) and applied a weighted version of the same graph partitioning software tool METIS [11] used in CSPA to find the consensus partitioning.

In this paper we present a probabilistic model derived from the co-association matrix, in which each object's co-associations are modelled as a probability distribution, which we refer to as association distribution. The model is presented in Section 2. A proposed information theoretic clustering combination process, which generates groups of association distributions is presented in Section 3. Groups of association distributions are aggregated through averaging to represent empirical cluster conditional probability distributions for each of the combined clusters. The model allows for cluster assignment probabilities to be estimated. Experimental results and analysis are presented in Section 4. Five datasets with various cluster structures are used to evaluate the performance of the proposed method at different values of the design parameters and in comparison with alternative methods that operate on the co-association matrix.

It is noted that diversity among the data clusterings of the ensemble can be created in a number of different ways, such as random restarts of a single clustering technique, or the use of different clustering techniques, or data resampling as has been reported recently in [12,13]. In this paper, we use multiple K-means clusterings using random initial restarts.

2 Probabilistic Model for Cluster Ensembles

2.1 Problem Formulation

Let $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ be a set of n data objects described as d-dimensional vectors of features. Let $\{x_1, \cdots, x_n\}$ denotes a set of n labels corresponding to each of the data objects. Let a cluster ensemble consists of m data clusterings of the n d-dimensional data vectors $\{\mathbf{x}_i\}_{i=1}^n$. While the clusterings are performed on the objects as vectors in their d-dimensional feature space, the modelling and combination methods described in this work deals exclusively with object labels, rather than objects as feature vectors. Throughout the paper, we will use the terms objects and object labels interchangeably to refer to object labels denoted by $\{x_i\}_{i=1}^n$, unless it is otherwise specified.

Let $\{\mathbf{y}_1, \cdots, \mathbf{y}_m\}$ be m n-dimensional labelling vectors representing the data clusterings where each clustering $\mathbf{y}_i$ of the n objects consists of a number k_i of clusters. Each entry y_{ij} in the vector $\mathbf{y}_i$ represents the cluster label, i.e. index of the cluster, to which data object x_j is assigned, such that $y_{ij} \in \{1, \cdots, k_i\}$. In this paper, the m clusterings are combined resulting in a clustering $\mathbf{y}$ which consists

of k^* clusters where k^* is prespecified. In addition, an n-dimensional probability vector is generated which gives estimate of the probability of association of each object to its assigned cluster.

2.2 Association Distributions

Let **S** be an $n \times n$ co-association matrix which summarizes the generated ensemble. Each entry s_{ij} of **S** represents the ratio of the number of times objects x_i and x_j co-occur in the same cluster to the number of clusterings m.

Let X be a discrete random variable which takes as values the object labels $\{x_1, \cdots, x_n\}$. Given an object label x_i, define an association distribution $p(x|x_i)$ as a probability distribution of the random variable X. We have a total of n association distributions, given each object label x_i. The probability values assumed by $p(x|x_i)$ are computed as follows:

$$P(X = x_j|x_i) = \frac{s_{ij}}{\sum_{k=1}^{n} s_{ik}} \qquad \forall j \in \{1, \cdots, n\}$$

That is, each association distribution is simply computed by normalizing each row/column of **S.** Hence, $p(x|x_i)$ satisfies the two conditions: $P(X = x_j|x_i) \geq 0, \forall j \in \{1, \cdots, n\}$, and $\sum_{j=1}^{n} P(X = x_j|x_i) = 1$. Each data object x_i contributes $\frac{1}{n}p(x|x_i)$ to the estimated probability distribution $p(x)$ of X, i.e.,

$$p(x) = \frac{1}{n}\sum_{i=1}^{n} p(x|x_i)$$

Suppose that the data objects are partitioned into k^* disjoint clusters, $\{c_j\}_{j=1}^{k^*}$, therefore, $p(x)$ can be written as follows:

$$p(x) = \sum_{j=1}^{k^*} P(c_j)p(x|c_j)$$

where $P(c_j)$ are the probabilities of the clusters and can be estimated by n_j/n, such that n_j is the number of data objects in cluster c_j. The cluster conditional probability distribution $p(x|c_j)$ of X is estimated by

$$p(x|c_j) = \frac{1}{n_j}\sum_{i=1}^{n_j} p(x|x_{i_j})$$

where x_{i_j} is the i^{th} object in cluster c_j. That is, $p(x|c_j)$ is the average of the n_j association distributions $p(x|x_{i_j})$.

But these clusters $\{c_j\}_{j=1}^{k^*}$ and their cluster conditional distributions $p(x|c_j)$ are unknown and represent exactly what we need to find. Therefore, by determining how to group the set of n association distributions, we can compute by aggregation through averaging an empirical cluster conditional probability

distribution of X for each cluster. Section 3 presents an information-theoretic consensus clustering process developed for grouping of association distributions.

After computing $p(x|c_j)$ for all $j \in \{1, \cdots, k^*\}$, a final object assignment (or re-arrangement) is performed where each object x_i is assigned to its most likely cluster c_j which satisfies that $P(X = x_i|c_j) \geq P(X = x_i|c_l)$ for $l \neq j$, and $l \in \{1, \cdots, k^*\}$.

Hence, each object is assigned to a particular cluster $\{c_i\}_{i=1}^{k^*}$. Furthermore, estimated assignment probabilities of objects to a cluster c_j, $P(c_j|x)$ can be computed using Bayes rule as follows.

$$P(c_j|x) = \frac{p(x|c_j)P(c_j)}{p(x)} = \frac{p(x|c_j)P(c_j)}{\sum_{l=1}^{k^*} p(x|c_l)P(c_l)}$$

Notice that the cluster conditional probability distribution introduced here is a discrete probability function defined on the finite space of object labels, rather than the feature vectors as conventionally assumed in supervised classification.

3 The Clustering Combination Process

In this section we present a heuristic information-theoretic method for grouping association distributions into k^* clusters, which as discussed in Section 2.2 is the basis for empirically estimating a cluster conditional probability distribution for each of the combined clusters. We call this method the Jensen-Shannon Divergence based Clustering Combination (JSDCC) as it mainly utilizes the Jensen-Shannon divergence.

We will use the short hand notation $p_i(x)$ and $p_i(x_j)$ to refer to $p(x|x_i)$ and $P(X = x_j|x_i)$ respectively. For details on the information theoretic measures used, the reader is referred to [14]. The Shannon entropy [15] $H(p_i)$ measures the information content of $p_i(x)$ and is given by

$$H(p_i) = -\sum_{j=1}^{n} p_i(x_j) \log p_i(x_j)$$

The Jensen-Shannon JS divergence is a measure of distance between the probability distributions $p_i(x)$, and $p_j(x)$ and is given by

$$JS(p_i, p_j) = H(\bar{p}_{ij}) - \left(\frac{H(p_i) + H(p_j)}{2}\right) \tag{1}$$

where $\bar{p}_{ij}(x)$ is the average of the probability distributions $p_i(x)$ and $p_j(x)$.

The JS is symmetric, bounded [16] and $JS(p_i, p_j) = 0 \iff p_i(x) = p_j(x)$, $\forall x$. Furthermore, it can be generalized to measure the divergence between any finite number r of probability distributions, and allows for weighted averages among distributions. The most general form of the JS divergence is given in Equation 2 where $\pi = \{\pi_1, \pi_2, \cdots \pi_r\}$ is the set of normalized weights. In this paper equal weights are used.

$$JS_{\pi}(\{p_i(x)|1 \leq i \leq r\}) = H\Big(\sum_{i=1}^{r} \pi_i p_i(x)\Big) - \sum_{i=1}^{r} \pi_i H(p_i) \quad (2)$$

The method starts by selecting k^* prototypes $T = \{t_1, \cdots, t_{k^*}\}$ from the set of association distributions $\{p(x|x_i)\}_{i=1}^{n}$, which will be used as initial representatives of the k^* clusters. From an information theory point of view, we would like to select the most informative prototypes, yet they should be as divergent as possible from each others. Hence, information content measured using the entropy is used to rank the distributions. But, we cannot select the k^* prototypes with the largest entropies, because although they can indeed contain a great deal of information, it may be the same information. So, we need to also choose them so that they are not redundant. Therefore we want to select those prototypes that have maximum divergence among themselves, where divergence is measured using the generalized Jensen-Shannon divergence given in Equation 2. In principle, this involves a search over $\binom{n}{k^*}$ different prototype sets, which is an enormous search for large n. Therefore, we use a incremental method which assesses prototypes individually based on their entropies and their divergences from previously selected prototypes, as described in Algorithm 1.

Algorithm 1 Selection of Prototypes

1: $t_1 \leftarrow \arg\max_{p_i(x)} H(p_i)$
2: **for all** $j \in \{2, \cdots, k^*\}$ **do**
3: **for all** $l \in \{1, \cdots, n\}$ **do**
4: compute Divergence $D(p_l) \leftarrow JS_{\pi}(p_l, t_l, t_2, \cdots, t_{j-1})$ as given in Equation 2, such that $\pi = 1/j$
5: **end for**
6: select $t_j \leftarrow \arg\max_{p_i(x)} H(\arg\max_{p_i(x)} D(p_i))$
7: **end for**

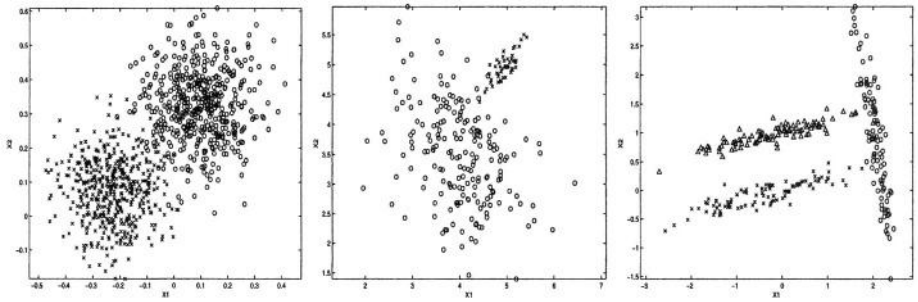
Following the selection of k^* cluster prototypes, a distribution merging procedure is used to merge each of the n association distributions with the prototype with minimum JS divergence. The procedure is summarized in Algorithm 2.

4 Experimental Analysis

Experiments are performed using the K-means clustering algorithm. The number of clusterings generated m are 10 and 50. We use fixed values of $k_i = k$ for all the m clusterings within an ensemble. Different values of k are used starting from the true number of clusters in a dataset and gradually increasing k. The final number of combined clusters k^* represent the number of true clusters and is assumed known. For each combination of k and m, we evaluate the quality of combined clusterings generated using SLink, JSDCC, CSPA and WSnnG using the commonly used F-measure. We also show the average ensemble's F-measure.

Algorithm 2 Merging of distributions

1: **for all** $i \in \{1, \cdots, n\}$ **do**
2: **for all** $j \in \{1, \cdots, k^*\}$ **do**
3: compute $JS(p_i, t_j)$ as given in Equation 1
4: **end for**
5: Let $t \leftarrow \arg\min_{t_j \in T} JS(p_i(x), t_j)$ and merge $p_i(x)$ with t.
6: **end for**
7: Average merged k^* clusters of distributions to estimate $\{p(x|c_i)\}_{i=1}^{k^*}$
8: Assign objects to their most likely clusters and compute assignment probabilities (See Section 2.2)



(a) (b) (c)

Fig. 1. Data sets (a) 2D2K dataset consists of 2 Gaussian clusters, not linearly separable, (b) 2D2C-Non-Spherical dataset has 2 unbalanced non spherical clusters, and (c) 2D3C-Strings has 3 string-like clusters of the same size

We use five dataset to evaluate the performance of the different methods in a number of different situations. The first is the 2D2K dataset used in [3] and downloaded from http://www.strehl.com/. We use a random sample of 300 data points. The dataset is shown in Figure 1 (a) and consists of two Gaussian clusters which are not linearly separable. The second dataset is 2D2C-Non-Spherical shown in Figure 1 (b), which is artificially generated and consists of 2 ellipsoidal clusters with different sizes (200,50), and covariances. The third dataset is 2D3C-Strings, shown in Figure 1 (c), and consisting of three ellipsoidal and elongated clusters of equal sizes (100,100,100), and different orientations. The fourth is the Iris dataset available from the UCI machine learning repository and consists of 4-dimensional points in 3 clusters, one linearly separable and 2 interleaving. The fifth is the Wisconsin diagnostic breast cancer data (WDBC), also available from the UCI repository, and consists of 569 instances and 30 numeric attributes, with class distribution of 357 benign, 212 malignant

Notice that the ensemble approach in [5] used varying k_i. In [3,8,9] various clustering techniques were used to generate the ensembles, and in [9], the vote threshold and the number of nearest neighbors were varied. Here, the focus is on evaluating their respective underlying combination methods on the ensembles

Table 1. F-Measure for the 2D2K data set

m	k	Ensemble's Mean	k^*	Consensus Functions			
				SLink	JSDCC	CSPA	WSnnG
10	2	0.986	2	0.986	0.986	0.963	0.963
10	4	0.731	2	0.627	0.986	0.980	0.976
10	6	0.579	2	0.666	0.983	0.973	0.973
10	8	0.535	2	0.666	0.979	0.973	0.976
10	10	0.497	2	0.976	0.983	0.960	0.966
50	2	0.986	2	0.986	0.986	0.963	0.963
50	4	0.725	2	0.704	0.983	0.980	0.980
50	6	0.582	2	0.665	0.979	0.980	0.980
50	8	0.533	2	0.665	0.983	0.976	0.976
50	10	0.480	2	0.665	0.979	0.976	0.976
Average				0.7606	0.9827	0.9724	0.9729

Table 2. F-Measure for the 2D2C-Non-Spherical

m	k	Ensemble's Mean	k^*	Consensus Functions			
				SLink	JSDCC	CSPA	WSnnG
10	2	0.743	2	0.769	0.769	0.729	0.725
10	4	0.643	2	0.984	0.976	0.718	0.722
10	6	0.546	2	0.984	0.984	0.729	0.722
10	8	0.471	2	0.725	0.924	0.722	0.725
10	10	0.414	2	0.729	0.608	0.722	0.725
50	2	0.716	2	0.769	0.769	0.729	0.733
50	4	0.637	2	0.984	0.972	0.729	0.729
50	6	0.544	2	0.984	0.984	0.722	0.726
50	8	0.474	2	0.984	0.980	0.733	0.729
50	10	0.415	2	0.984	0.984	0.726	0.729
Average				0.8896	0.8950	0.7259	0.7265

specified by the parameters m and k as shown in the first two columns of the results Tables 1, 2, 3, 4, and 5.

From the results, it is noted that in cases of 2D2K, 2D3C-Strings, Iris, and WDBC there is a decline observed in terms of the F-measure with the SLink approach which is believed to be due to its inherent chaining effect particularly when the clusters are not linearly separable. The CSPA and WSnnG did not adapt successfully to the cluster structure of the 2D2C-NonSperical, whereas the SLink and JSDCC performed well in most ensembles, by uncovering the cluster structure which in this case the K-means (see average at $k = 2$) has failed to find. The 2D3C-Strings was the hardest on the K-means (see average at $k = 3$), whereas at some combination of m and k the performance of JSDCC, CSPA and WSnnG was relatively good in discovering the cluster structure, yet a dependency on k is clearly observed. In the case of the Iris, JSDCC, CSPA and WSnnG outperformed SLink, and in case of WDBC, JSDCC was best.

Table 3. F-Measure for the 2D3C-Strings Dataset

m	k	Ensemble's Mean	k^*	Consensus Functions			
				SLink	JSDCC	CSPA	WSnnG
10	3	0.652	3	0.663	0.690	0.7	0.695
10	5	0.631	3	0.752	0.700	0.686	0.722
10	7	0.644	3	0.664	0.669	0.729	0.669
10	9	0.561	3	0.590	0.926	0.959	0.873
10	11	0.513	3	0.576	0.811	0.929	0.896
50	3	0.655	3	0.690	0.690	0.659	0.635
50	5	0.628	3	0.725	0.690	0.635	0.722
50	7	0.627	3	0.751	0.751	0.641	0.736
50	9	0.568	3	0.699	0.924	0.926	0.809
50	11	0.511	3	0.699	0.618	0.953	0.923
Average				0.6809	0.7469	0.7817	0.7680

Table 4. F-Measure for the Iris dataset

m	k	Ensemble's Mean	k^*	Consensus Functions			
				SLink	JSDCC	CSPA	WSnnG
10	3	0.872	3	0.891	0.891	0.853	0.839
10	5	0.759	3	0.758	0.831	0.966	0.946
10	7	0.708	3	0.758	0.890	0.96	0.919
10	9	0.637	3	0.758	0.831	0.78	0.826
10	11	0.590	3	0.758	0.831	0.80	0.753
10	13	0.520	3	0.771	0.973	0.973	0.886
50	3	0.846	3	0.891	0.891	0.853	0.839
50	5	0.755	3	0.831	0.897	0.919	0.893
50	7	0.684	3	0.758	0.831	0.973	0.946
50	9	0.622	3	0.758	0.966	0.979	0.693
50	11	0.573	3	0.758	0.897	0.979	0.973
50	13	0.525	3	0.758	0.973	0.973	0.686
Average				0.7873	0.8918	0.9173	0.8499

Table 5. F-Measure for the WDBC Data

m	k	Ensemble's Mean	k^*	Consensus Functions			
				SLink	JSDCC	CSPA	WSnnG
10	2	0.844	2	0.844	0.844	0.675	0.635
10	3	0.799	2	0.890	0.890	0.818	0.802
10	4	0.764	2	0.682	0.779	0.839	0.792
10	5	0.679	2	0.683	0.855	0.694	0.692
10	6	0.594	2	0.683	0.877	0.840	0.844
50	2	0.844	2	0.844	0.844	0.675	0.635
50	3	0.799	2	0.743	0.890	0.815	0.801
50	4	0.764	2	0.682	0.821	0.797	0.792
50	5	0.678	2	0.683	0.855	0.700	0.692
50	6	0.588	2	0.683	0.854	0.844	0.675
Average				0.7417	0.8509	0.7697	0.7360

The average performance over all the ensemble configurations shown for each dataset is as follows. In the case of the 2D2K dataset, JSDCC improves over the lowest performing combination method by 29%. In the case of the 2D2C-Non-Spherical, JSDCC was best and improves over the lowest by 23% and by 20% over the K-means at the true number of cluster (k=2). In the case of 2D3C-Strings, it is lower by 5% than the higest performing method and improves over the lowest by 8% and by 14% over the K-means at true number of clusters (k=3). In the case of the Iris data, it is 2% lower than the highest and improving by 14% over the lowest. Finally, in the case of WDBC, it is the best, and improves over the lowest combination method by 16%.

5 Discussion and Conclusion

The co-association matrix represents a voting medium allowing the generation of consensus clustering. In this paper, we proposed a probabilistic model based on the co-association matrix in which the definition of association distributions allowed for the generation of empirical cluster conditional association distributions for each of the combined clusters. The model allowed the evaluation of the probabilities with which objects belong to the each cluster. The key objectives of developing this model are to represent the associations of neighboring data objects and to allow for the manipulation of their association distributions using probabilistic and information theoretic tools. Noticeably, the model expresses of the same idea of shared nearest neighbors [17,8,9], since the association distribution is indeed a function of each object's co-associated neighbors. A Jensen-Shannon divergence based Clustering Combination (JSDCC) method was developed for grouping of association distributions.

The JSDCC method has a quadratic computational complexity $(O(n^2))$, since the computation of the JS divergence is $O(n)$. Future work will focus on optimizing the approach for scalability to large datasets. For instance, we can cut down the number of distance computations by processing all the objects that has high s_{ij} values with selected prototypes. That is, we can use both the s_{ij} values and divergences jointly, leaving divergence computation for less obvious cases. Although the space complexity of the co-association matrix itself is $O(n^2)$, it is possible in future work to limit the number of co-associated objects to the K-nearest neighbors which will reduce the space requirement to $O(Kn)$.

Acknowledgements

We would like to thank the anonymous referees for their helpful comments. This work was partially funded by an NSERC strategic grant.

References

1. A.K. Jain, M.N Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys,* 31(3):264–323, September 1999.
2. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

3. A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining partitionings. In *Conference on Artificial Intelligence (AAAI 2002),* pages 93–98, Edmonton, July 2002. AAAI/MIT Press.
4. A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR),* 3:583–617, December 2002.
5. A. Fred and A.K. Jain. Data clustering using evidence accumulation. In *Proceedings of the 16th International Conference on Pattern Recognition. ICPR 2002,* volume 4, pages 276-280, Quebec City, Quebec, Canada, August 2002.
6. A. Fred and A. K. Jain. Robust data clustering. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2003,* Madison - Wisconsin, USA, June 2003.
7. Evgenia Dimitriadou, Andreas Weingessel, and Kurt Hornik. Voting-merging: An ensemble method for clustering. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Artificial Neural Networks-ICANN 2001,* pages 217-224, Vienna, Austria, August 2001. Springer.
8. H. Ayad and M. Kamel. Finding natural clusters using multi-clusterer combiner based on shared nearest neighbors. In *Multiple Classifier Systems: Fourth International Workshop, MCS 2003, UK, Proceedings.,* pages 166-175, 2003.
9. H. Ayad and M. Kamel. Refined shared nearest neighbors graph for combining multiple data clusterings. In *The 5th International Symposium on Intelligent Data Analysis IDA 2003. Berlin, Germany, Proceedings. LNCS. Springer.,* 2003.
10. A. Topchy, A.K. Jain, and W. Punch. Combining multiple weak clusterings. In *IEEE Intl. Conf. on Data Mining 2003, Proceedings,* pages 331-338, Melbourne, F1., November 2003.
11. George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, Department of Computer Science and Engineering, University of Minnesota, 1995.
12. B. Fischer and J.M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25(4):513-518, 2003.
13. S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling based method for class discovery and visualization of gene expression microarray data. *Machine Learning,* 52(1-2):91-118, 2003.
14. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, USA, 1991.
15. C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal,* 27:379-423, 1948.
16. J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory,* 37(1):145-151, 1995.
17. R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared nearest neighbors. *IEEE Transactions on Computers,* C-22(11):1025-1034, November 1973.

Classifier Fusion Using Triangular Norms

Piero Bonissone, Kai Goebel, and Weizhong Yan

GE Global Research, One Research Circle, Niskayuna, NY 12309, USA
(bonissone,goebelk,yan}@research.ge.com

Abstract. This paper describes a method for fusing a collection of classifiers where the fusion can compensate for some positive correlation among the classifiers. Specifically, it does not require the assumption of evidential independence of the classifiers to be fused (such as Dempster Shafer's fusion rule). The proposed method is associative, which allows fusing three or more classifiers irrespective of the order. The fusion is accomplished using a generalized intersection operator (T-norm) that better represents the possible correlation between the classifiers. In addition, a confidence measure is produced that takes advantage of the consensus and conflict between classifiers.

1 Introduction

Design of a successful classifier fusion system consists of two important parts: design of the individual classifiers, selection of a set of classifiers [9, 17], and design of the classifier fusion mechanism [14]. Key to effective classifier fusion is the diversity of the individual classifiers. Strategies for boosting diversity include: 1) using different types of classifiers; 2) training individual classifiers with different data set (bagging and boosting); and 3) using different subsets of features.

In the literature we can find many fusion methods derived from different underlying frameworks, ranging from Bayesian probability theory [18], to fuzzy sets [5], Dempster-Shafer evidence theory [1], group decision-making theory (majority voting [10], weighted majority voting, and Borda count [7]). Existing fusion methods typically do not address correlation among classifiers' output errors. For example, methods based on Dempster-Shafer theory assume evidential independence [16, p. 147].

In real-world applications, however, most individual classifiers exhibit correlated errors due to common data sources, domain knowledge based filtering, etc. Typically, they all tend to make the same classification errors on the most difficult patterns. Prior research has focused on relating the correlation in the classifiers' errors to the ensemble's error [17], quantifying the degree of the correlation [8, 13], and minimizing the correlation degree between the classifiers [11]. However, not much attention has been devoted to investigating fusion mechanisms that can compensate for the impact of the classifiers' error correlations.

This paper investigates the impact of parameter changes within a T-norm based framework for a set of n classifiers. Rather than using pairwise correlations of classifiers' errors as a method to select a subset of classifiers, we focus on the overall be-

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 154–163, 2004.

havior of the fusion. We use a single parameter in an attempt to provide a *uniform* compensation for the correlation of the classifiers' errors. We explore the impact of such compensation on the final classification by using four performance metrics. We empirically obtain the value of the parameter that gives us the best solution in this performance space.

2 Fusion via Triangular Norms

We propose a general method for the fusion process, which can be used with classifiers that may exhibit any kind of (positive, neutral, or negative) correlation with each other. Our method is based on the concept of Triangular Norms, a multi-valued logic generalization of the Boolean intersection operator. To justify the use of the intersection operator it can be argued that the fusion of multiple decisions, produced by multiple sources, regarding objects (classes) defined in a common framework (universe of discourse) consists in determining the underlying of degree of consensus for each object (class) under consideration, i.e., the intersections of their decisions. With the intersections of multiple decisions one needs to account for possible correlation among the sources, to avoid under- or over-estimates. Here we explicitly account for this by the proper selection of a T-norm operator.

We combine the outputs of the classifiers by selecting the generalized intersection operator (T-norm) that better represents the possible correlation between the classifiers. With this operator we will intersect the assignments of the classifiers and compute a derived measure of consensus. Under certain conditions, defined in section 2.2, we will be able to perform this fusion in an associative manner, e.g., we will combine the output of the fusion of the first two classifiers with the output of the third classifier, and so on, until we have used all available classifiers. At this stage, we can normalize the final output (showing the degree of selection as a percentage), identify the strongest selection of the fusion, and qualify it with its associated degree of confidence.

2.1 Triangular Norms

Triangular norms (T-norms) and Triangular conorms (T-conorms) are the most general families of binary functions that satisfy the requirements of the conjunction and disjunction operators, respectively. T-norms $T(x,y)$ and T-conorms $S(x,y)$ are two-place functions that map the unit square into the unit interval, i.e., $T(x,y):[0,1]\times[0,1]\rightarrow[0,1]$ and $S(x,y):[0,1]\times[0,1]\rightarrow[0,1]$. They are monotonic, commutative and associative functions. Their corresponding boundary conditions, i.e., the evaluation of the T-norms and T-conorms at the extremes of the [0,1] interval, satisfy the truth tables of the logical AND and OR operators. They are related by the DeMorgan duality, which states that if $N(x)$ is a negation operator, then the T-conorm $S(x,y)$ can be defined as $S(x,y) = N(T(N(x), N(y)))$.

In Bonissone and Decker [4], six parameterized families of T-norms and their dual T-conorms were discussed and analyzed by the authors. Of the six parameterized families, one family was selected due to its complete coverage of the T-norm space and its numerical stability. This family, originally defined in [15], has a parameter *p* that spans the space of T-norms. By selecting different values of p we can instantiate T-norms with different properties. See [3, 15] for more detailed information.

2.2 Determination of a Combined Decision

Let us define *m* classifiers $S_1, \ldots S_m$, such that the output of classifier S_j is the vector $\boldsymbol{I}^j$ showing the normalized decision of such classifier to the N classes. In this representation, the last $(N+1)^{th}$ element represents the universe of all classes, U, and is used to indicate the classifier's lack of commitment, i.e. no-decision:

$$\boldsymbol{I}^j = [\, I^j(1), I^j(2), \ldots, I^j(N+1)], \text{ where } I^j(i) \in [0,1] \text{ subject to: } \sum_{i=1}^{N+1} I^j(i) = 1$$

We define the un-normalized fusion of the outputs of two classifiers S_1 and S_2 as:

$$F(I^1, I^2) = Extraction\left[Outerproduct(I^1, I^2, T)\right] = Extraction[A] \quad (1)$$

where the outer-product is a well-defined mathematical operation, which takes as arguments two N-dimensional vectors $\boldsymbol{I}^1$ and $\boldsymbol{I}^2$ and generates as output the *N*x*N* dimensional array A. Each element *A(i,j)* is the result of applying the operator *T* to the corresponding vector elements, namely $\boldsymbol{I}^1(i)$ and $\boldsymbol{I}^2(j)$, i.e.:

$$A(i,j) = T[\boldsymbol{I}^1(i), \boldsymbol{I}^2(j)]. \quad (2)$$

The *Extraction* operator recovers the un-normalized output in vector form:

$$Extraction[A] = [I'(1), I'(2), \ldots, I'(N+1)] \quad (3)$$

$$\text{where : } I'(i) = A(i,i) + A(i,N+1) + A(N+1,i) \text{ for i = 1, N} \quad (4)$$

$$\text{and } I'(N+1) = A(N+1, N+1) \quad (5)$$

The extraction operator leverages the fact that all classes are disjoints. A justification for equations (4) and (5) can be found in the case analysis illustrated in eq. (7).

There is an infinite number of T-norms. Therefore, to cover their space we used the family proposed by Schweizer and Sklar [15], in which each T-norm is uniquely defined by a real valued parameter *p*. We selected the six most representative ones for some practical values of information granularity, as listed in Table 1.

Table 1. Example of Outer Product using as operator the function T(x,y).

T-norm	Value of *p*	Correlation Type
$T_1(x,y) = \max(0, x+y-1)$	p = -1	Extreme case of negative correlation
$T_2 = Max(0, x^{0.5} + y^{0.5} - 1)^2$	p = -0.5	Partial case of negative correlation
$T_3 = x * y$	$p \to 0$	No correlation
$T_4(x,y) = (x^{-0.5} + y^{-0.5} - 1)^{-0.5}$	p = 0.5	Mildly Positive Correlation
$T_5(x,y) = (x^{-1} + y^{-1} - 1)^{-1}$	p = 1	Partial case of positive correlation
$T_6 = \min(x,y)$	$p \to \infty$	Extreme case of positive correlation

The selection of the best T-norm to be used as intersection operation in the fusion of the classifiers depends on the potential correlation among their errors. $\mathbf{T_6}$ (the minimum operator) should be used when one classifier subsumes the other one ***(extreme case of positive correlation)***. $\mathbf{T_3}$ should be selected when the classifiers errors are ***uncorrelated*** (similar to the evidential independence in Dempster-Shafer). $\mathbf{T_1}$ should be used if the classifiers are mutually exclusive ***(extreme case of negative correlation).*** The operators $\mathbf{T_2}$, $\mathbf{T_4}$, and $\mathbf{T_5}$ should be selected when the classifiers errors show intermediate stages of negative ($\mathbf{T_2}$) or positive correlation ($\mathbf{T_4}$ and $\mathbf{T_5}$). Of course, other T-norms could also be used. These five T-norms are simply good representatives of the infinite number of functions that satisfy the T-norm properties.

From the associativity of the T-norms we can derive the associativity of the fusion

$$F(I^1,F(I^2,I^3)) = F(F(I^1,I^2),I^3) \tag{6}$$

providing that equation (4) only contains one term. This is the case when: a) there is *no lack of commitment* in the classifiers [$A(i,N+1)=A(N+1,i)=0$ *for* $i=1,...,N$]; b) the *lack of commitment is complete* in one classifiers [$A(i,i)=0$ *for* $i=1,...,N$ and either $A(i,N+1)=I^1(i)$ & $A(N+1,i)=0$ or $A(N+1,i)=I^2(i)$ & $A(i,N+1)=0$]; c) the *lack of commitment is complete* in both classifiers [*eq(4)* $=0$ & *eq(5)* $=1$]. In any other case, the preservation of associativity requires the distributivity of the T-norm over the addition operator of equation (4). This distributivity is satisfied only by T-norm $\boldsymbol{T_2}$ (scalar product). For any other T-norm the lack of associativity means that the outer-product A defined by equation (2) must be computed on all dimensions at once, using as operator a dimensionally extended Schweizer & Sklar T-norm:

$$T_p(x_1,...,x_k) = \left[\sum_{i=1}^{k} x_i^{-p} - k + 1\right]^{-1/p} \quad \text{if } (p>0) \text{ or } (p<0) \text{ and } \left[\sum_{i=1}^{k} x_i^{-p}\right] \geq (k-1)$$

Given our premise that the classes are disjoint, we have four possible situations:

(a) when $i=j$ *and* $i<(N+1)$ then $r_i \cap r_j = r_j \cap r_i = r_i$ (7a)

(b) when $i=j$ *and* $i=(N+1)$ then $r_i \cap r_j = U$ (the universe of classes) (7b)

(c) when $i \neq j$ *and* $i<(N+1)$ and $j<(N+1)$ then $r_i \cap r_j = \phi$ (the empty set) (7c)

(d) when $i \neq j$ *and either* $i=(N+1)$ then $U \cap r_j = r_j$ *or* $j=(N+1)$ then $r_i \cap U = r_i$ (7d)

Figure 1 illustrates the result of the intersections of the classes and the universe U.

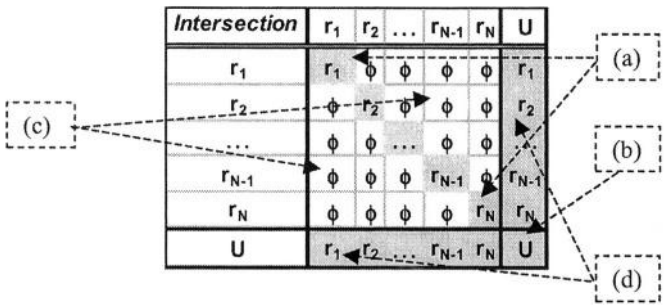


Fig. 1. Intersection of disjoint classes and the universe U.

Note that when DS theory is applied to subsets containing only singletons or the entire universe of classes U, its fusion rule becomes a special case of our proposed methodology.

2.3 Measure of Confidence

In cases where classes are ordered, we can compute the confidence in the fusion by defining a measure of the scattering around the main diagonal of the confusion matrix. The more weights are assigned to elements outside the main diagonal, the less is the measure of the consensus among the classifiers. We can represent this concept by defining a penalty matrix $\boldsymbol{P} = [P(i,j)]$, of the form:

$$P(i,j) = \begin{cases} \max(0, (1 - W * |i - j|))^d & \text{for } 1 \le i \le N \text{ and } 1 \le j \le N \\ 1 & \text{for } i = (N+1) \text{ or } j = (N+1) \end{cases} \quad (8)$$

This function rewards the presence of weights on the main diagonal, indicating agreement between the two classifiers, and penalizes the presence of elements off the main diagonal, indicating conflict. The conflict increases in magnitude as the distance from the main diagonal increases. For example, *for* ***W=0.2*** and ***d=5*** we have the following penalty matrix:

P	r_1	r_2			r_N	U
r_1	1	0.3	0.1	0	0	1
r_2	0.3	1	0.3	0.1	0	1
	0.1	0.3	1	0.3	0.1	1
	0	0.1	0.3	1	0.3	1
r_N	0	0	0.1	0.3	1	1
U	1	1	1	1	1	1

Fig. 2. Penalty matrix P for W=0.2 and d=5.

Of course any other function penalizing elements off the main diagonal (any suitable non-linear function of the distance from the main diagonal, i.e. the absolute value $|i-j|$) could also be used. The reason for this penalty function is that ***conflict is gradual***, since the classes have an ordering. Therefore, we want to capture the fact the discrepancy between classes r_1 and r_2 is smaller than the discrepancy between r_1 and r_3. The shape of the penalty matrix $\boldsymbol{P}$ captures this concept, as $\boldsymbol{P}$ shows that the confidence decreases non-linearly with the distance from the main diagonal.

A measure of the normalized confidence $\hat{C}$ is the sum of element-wise products between $\hat{A}$ and $\boldsymbol{P}$, e.g.:

$$\hat{C} = \text{Normalized Confidence}(\hat{A}, P) = \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} \hat{A}(i,j) * P(i,j) \quad (9)$$

where $\hat{A}$ is the normalized fusion matrix.

We could interpret the confidence factor $\hat{C}$ as the weighted cardinality of the normalized assignments around the main diagonal, after all the classifiers have been fused.

In the special case of Dempster Shafer, the measure of confidence $\hat{C}$ is the complement (to one) of the measure of conflict K, i.e.: $\hat{C} = 1 - K$, where K is the sum of weights assigned to the empty set. We would like to point out that the fusion rule based on Dempster-Shafer corresponds to the selection of:

(a) T-norm operator $T(x,y) = x*y$
(b) Penalty function using $d = \infty$

Constraint b) implies that the penalty matrix P will be:

P	r_1	r_2			r_N	U
r_1	1	0	0	0	0	1
r_2	0	1	0	0	0	1
	0	0	1	0	0	1
	0	0	0	1	0	1
r_N	0	0	0	0	1	1
U	1	1	1	1	1	1

Fig. 3. Penalty matrix P for the special case of DS confidence computation: $d=\infty$.

Therefore the two additional constraints a) and b) required by Dempster-Shafer theory imply that

(a) The classifiers to be fused must be uncorrelated (evidentially independent) and
(b) There is no ordering over the classes, and any kind of disagreement (weights assigned to elements off the main diagonal) can only contribute to a measure of conflict and not (at least to a partial degree) to a measure of confidence. In DS, the measure of conflict K is the sum of weights assigned to the empty set. This corresponds to the elements with a 0 in the penalty matrix P illustrated in Figure 3.

Clearly, our proposed method is more general, since it does not require these additional constraints, and includes the DS fusion rule as a special case.

2.4 Decisions from the Fusion Process

The decisions for each class can be gathered by adding up ***all*** the weights assigned to them, as indicated in equations (4) and (5).

3 Examples

3.1 Example (1): Agreement between the Classifiers without Discounting

Let $I^1 = [0.8, 0.15, 0.05, 0, 0, 0]$ and $I^2 = [0.9, 0.05, 0.05, 0, 0, 0]$

This indicates that both classifiers are showing a strong preference for the first class as they are assigned 0.8 and 0.9, respectively. We fuse these classifiers using

each of the T-norm operators defined above and normalize the results so that ***the sum of the entries is equal to one.*** Note that during the process we will use the unnormalized matrices A to preserve the associative property. Using the expressions for weights of a class, we can compute the final weights for the N classes and the universe. Table 2 shows the results of the fusion of classifiers S1 and S2, using each of the five T-norms, with the associated normalized confidence measure.

Table 2. Fusion of two classifiers and associated confidence measure with agreement.

	C1	C2	C3	C4	C5	U	Norm. Conf.
$T_6 = \min(x, y)$	0.615	0.038	0.038	0	0	0	0.774
$T_5(x,y) = (x^{-1} + y^{-1} - 1)^{-1}$	0.633	0.034	0.022	0	0	0	0.770
$T_3 = x * y$	0.720	0.008	0.003	0	0	0	0.797
$T_2 = Max(0, x^{0.5} + y^{0.5} - 1)^2$	0.807	0.000	0.000	0	0	0	0.858
$T_1 = \max(0, x + y - 1)$	0.933	0.000	0.000	0	0	0	0.955

3.2 Example (2): Disagreement between the Classifiers, without Ignorance

In a situation in which there is a *discrepancy* between the two classifiers, this fact will be captured by the confidence measure. For instance, let's assume that the two classifiers are showing strong preferences for two *different* classes, e.g.

$$I^1 = [0.15, 0.85, 0.05, 0, 0, 0] \quad \text{and} \quad I^2 = [0.9, 0.05, 0.05, 0, 0, 0]$$

The results of their fusion are summarized in Table 3. None of the classes has a high weight and the normalized confidence has dropped, with respect to Table 2.

Table 3. Fusion of two classifiers and related confidence measure with disagreement.

	C1	C2	C3	C4	C5	U	Norm. Conf.
$T_6 = \min(x, y)$	0.115	0.038	0.038	0	0	0	0.438
$T_5(x,y) = (x^{-1} + y^{-1} - 1)^{-1}$	0.127	0.043	0.022	0	0	0	0.438
$T_3 = x * y$	0.135	0.040	0.003	0	0	0	0.434
$T_2 = Max(0, x^{0.5} + y^{0.5} - 1)^2$	0.128	0.016	0.000	0	0	0	0.416
$T_1 = \max(0, x + y - 1)$	0.067	0.000	0.000	0	0	0	0.373

4 Application and Results

Insurance underwriting is a classification problem. The underwriting (UW) process consists of assigning a given insurance application, described by its medical and demographic records, to one of the risk categories (referred to as rate classes). Tradi-

tionally, highly trained human individuals perform insurance underwriting. The granularization of risk into rate classes is dictated by industry regulation and by the impossibility of human underwriters to achieve consistency of decisions over real-valued risk granularity. A given application for insurance is compared against standards put forward by the insurance company and classified into one of the risk categories available for the type of insurance requested by the applicant. The risk categories then affect the premium paid by the applicant; the higher the risk category, the higher the premium. Such manual underwriting, however, is not only time-consuming, but also often inadequate in consistency and reliability. The inadequacy becomes more apparent as the complexity of insurance applications increases. The automation of this decision-making process has strong accuracy, coverage and transparency requirements. There are two main tradeoffs to be considered in designing a classifier for the insurance underwriting problem: *1) Accuracy versus coverage* - requiring low misclassification rates for high volume of applications; *2) Accuracy versus interpretability* - requiring a transparent, traceable decision-making process. Given the highly non-linear boundaries of the rate classes, continuous approaches such as multiple regressions were not successful and a discrete classifier was deployed for production.

Once this process is automated by a production engine, we need to perform *off-line* audits to monitor and assure the quality of the production engine decisions. Auditing automated insurance underwriting could be accomplished by fusing the outputs of multiple classifiers. The purpose of this fusion, then, is to assess the quality of the decisions taken by a production engine. In addition, this fusion will identify the best cases, which could be used to tune the production engine in future releases, as well as controversial or unusual cases that should be highlighted for manual audit by senior underwriters, as part of the Quality Assurance process.

We tested the approach against a case base containing a total of 1,875 cases, which could be assigned to one of five rate-classes. The fusion was performed using different T-norms. The classifiers to be fused were a classification and regression tree (MARS), a bank of binary MVP neural (NN) nets using back-propagation as the learning algorithm, a cased-based reasoning engine (CBE), and a bank of binary support vector machines (SVM). Table 4 shows the performance of the individual classifiers as expressed by the true positive rate.

Table 4. Classifier Performance.

Classifier	True Positive Rate
MARS	94.72%
NN	94.08%
CBE	91.52%
SVM	94.51%

4.1 Experiments Results Using Different T-Norms

Table 5 shows the results of using three *Tnorms* as the outer-product operators (using the same set of 1,875 non-nicotine users). When using T3 *(scalar product)* combined

with a strict determination of the conflict (e.g. for $d=\infty$ in the computation of penalty matrix P), we have an aggregation analogous to that of Dempster-Shafer's rule of combination. This aggregation relies on the *evidential uncorrelation* of the sources, a constraint that is not easy to satisfy in real situations. In our case, we used a pre-processing stage *(Tagging)* to annotate the input data with domain knowledge. Since this pre-processing stage is applied to a common set of inputs for *all* models, it is possible to introduce a common bias; hence the models could exhibit partial positive correlation. We sampled the space of T-norms corresponding to positive correlation and we found that *T4* is the ***Pareto-best*** to account for this positive correlation, according to the metrics in Table 5. We also compared the T-norm fusion against a baseline fusion using an averaging scheme. The strict or liberal errors used in Table 5 are extensions of type 1 and type 2 errors in a classification matrix of the rate classes.

Table 5. Performance of Average and Fusion (using three T-norms).

Metrics	Average	Fusion using T3 ($p \rightarrow 0$)	Fusion using T4 ($p = 0.5$)	Fusion using T5 ($p = 1$)
True Positive Rate	94.24%	94.08%	**94.72%**	94.08%
Incorrect Rate (too strict)	3.47%	3.47%	**3.09%**	3.47%
Incorrect Rate (too liberal)	2.29%	2.35%	**2.19%**	2.40%
No Decisions	0%	0.11 %	**0%**	0.05 %

5 Conclusions

We introduced a fusion mechanism based on an outer-product of the outputs, using a Triangular norm as the operator. The output of the fusion was a class distribution and a measure of conflict among all the classifiers. We normalized the final output, identified the strongest selection of the fusion, and qualified the decision with an associated confidence measure. We considered the output of each classifier as a weight assignment, representing the (un-normalized) degree to which a given class was selected by the classifier. As in DS theory, the assignment of weights to the universe all rate-classes represented the lack of commitment to a specific decision.

We applied the fusion to the quality assurance (QA) problem for automated underwriting. When the degree of confidence of a fused decision was below a lower confidence bound, that case became a candidate for *auditing*. When the degree of confidence of a fused decision was above an upper confidence bound, that case became a candidate for *augmenting the Standard Reference Decision set (SRD)*. We tested the fusion module with data sets for nicotine and non-nicotine users. In an analysis of 1,875 non-nicotine applications, we generated a distribution of the quality of the production engine. By focusing on the two tails of such distribution, we identified **~9%** of the most reliable decisions and **4.9%** of the least reliable ones.

The proposed fusion module plays a key role in supporting the quality assurance of the knowledge-based classifier, by selecting the most questionable cases for auditing. At the same time, the fusion module supports the standard reference decision lifecy-

cle by identifying the most reliable cases used for updating it [4]. Our next step will be to use backward search, guided by a diversity measure [12,14], to select the smallest subset of classifiers to be deployed in the production version of this QA process.

References

1. Al-Ani, A. and Deriche, M. A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence, *J. Artificial Intelligence Research,* 17:333-361, 2002.
2. Bonissone, P.P. and Decker, K.S. Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity, in Kanal and Lemmer (editors) *Uncertainty in Artificial Intelligence,* pages 217-247, North-Holland, 1986.
3. Bonissone, P.P. Summarizing and Propagating Uncertain Information with Triangular Norms, *International Journal of Approximate Reasoning,* 1(1):71-101, January 1987.
4. Bonissone, P.P. Automating the Quality Assurance of an On-line Knowledge-Based Classifier By Fusing Multiple Off-line Classifiers, *Proc. IPMU 2004,* Perugia (Italy), July 2004.
5. Cho, S. and Kim, J., Combining multiple neural networks by fuzzy integral for robust classification, IEEE Trans. on Systems, Man, and Cybernetics, 25(2):380-384,1995
6. Dempster, A.P. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics,* 38:325-339, 1967.
7. Ho, T.K., Hull, J.J., and Srihari, S.N. Decision Combination in Multiple Classifier Systems, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 16, No.1, pp.66-75, 1994.
8. Kim, J., Seo, K., and Chung, K. A Systematic approach to Classifier selection on combining multiple classifiers for handwritten digit recognition, Proceedings of the 4th International Conference on Document Analysis and Recognition, 2:459-462, 1997.
9. Kuncheva L.I. Switching between selection and fusion in combining classifiers: An experiment, *IEEE Transactions on SMC, Part B,* 32 (2), 2002, 146-156.
10. Lam L. and Suen, C. Y. Increasing experts for majority vote in OCR: Theoretical considerations and strategies. *Proc. 4th Int. Workshop on Frontiers in Handwriting Recognition,* Taipei, Taiwan, 245-254, 1994.
11. Niyogi, P Pierrot, J-B, and Siohan.O. On decorrelating classifiers and combining them, MIT AI Lab., September 2001.
12. Partridge, D., Yates, W.B. Engineering multiversion neural-net systems. *Neural Computation,* 8:869-893, 1996.
13. Petrakos, M., Kannelopoulos, I., Benediktsson, J., and Pesaresi, M. The Effect of Correlation on the Accuracy of the Combined Classifier in Decision Level Fusion, *Proceedings of IEEE 2000 International Geo-science and Remote Sensing Symposium,* Vol. 6, 2000.
14. Roli, F., Giacinto, G., and Vernazza, G. Methods for Designing Multiple Classifier Systems, *MCS 200, LNCS 2096,* 78-87, 2001.
15. Schweizer, B., and Sklar, A. Associative Functions and Abstract Semi-Groups,. *Publicationes Mathematicae Debrecen,* 10:69-81, 1963.
16. Shafer, G. *A Mathematical Theory of Evidence,* Princeton University Press, NJ, 1976.
17. Tumer, K.& Ghosh, J. Error correlation and error reduction in ensemble classifiers, *Connection Science,* 8:385-404,1996.
18. Xu, L., Krzyzak, A., & Suen, C.Y. Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition, *IEEE Syst. Man and Cybernetics,* 22(3), 1992.

Dynamic Integration of Regression Models

Niall Rooney[1], David Patterson[1], Sarab Anand[1], and Alexey Tsymbal[2]

[1] NIKEL, Faculty of Engineering, 16J27
University Of Ulster at Jordanstown
Newtonabbey, BT37 OQB, United Kingdom
{nf.rooney,wd.patterson,ss.anand}@ulster.ac.uk
[2] Alexey Tsymbal, Department Of Computer Science, Trinity College Dublin
{Alexey.Tsymbal}@cs.tcd.de

Abstract. In this paper we adapt the recently proposed Dynamic Integration ensemble techniques for regression problems and compare their performance to the base models and to the popular ensemble technique of Stacked Regression. We show that the Dynamic Integration techniques are as effective for regression as Stacked Regression when the base models are simple. In addition, we demonstrate an extension to both Stacked Regression and Dynamic Integration to reduce the ensemble set in size and assess its effectiveness.

1 Introduction

The purpose of ensemble learning is to build a learning model which integrates a number of base learning models, so that the model gives better generalization performance on application to a particular data-set than any of the individual base models [3]. Ensemble learning consists of two problems; ensemble generation: how does one generate appropriate base models? and ensemble integration: how does one integrate the base models' predictions to improve performance? Ensemble generation can be characterized as being homogeneous if each base learning model uses the same learning algorithm or heterogeneous if the base models can be built from a range of learning algorithms. Ensemble integration can be addressed by either one of two mechanisms, either the predictions of the base models are combined in some fashion during the application phase to give an ensemble prediction (*combination/fusion approach*) or the prediction of one base model is selected according to some criteria to form the final prediction (*selection approach*) [9].

Theoretical and empirical work has shown the ensemble approach to be effective with the proviso that the base models are diverse and sufficiently accurate [3]. These measures are however not necessarily independent of each other. If the prediction error of all base models is very low, then their learning hypothesis must be very similar to the true function underlying the data, and hence they must of necessity, be similar to each other i.e. they are unlikely to be diverse. In essence then there is often a trade-off between diversity and accuracy [2].

There has been much research work on ensemble learning for regression in the context of neural networks, however there has been less research carried out in terms

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 164–173, 2004.

of using homogeneous ensemble techniques to improve the performance of simple regression algorithms. In this paper we look at improving the generalization performance of nearest neighbours (k-NN) and least squares linear regression (LR). These methods were chosen as they are simple models with different approaches to learning in that linear regression is an eager model which tries to approximate the true function by a global linear function and k-nearest neighbours is a lazy model which tries to approximate the true function locally.

2 Ensemble Integration

The initial approaches to ensemble *combination* for regression were based on the linear combination of the base models according to the function:

$$\sum_{i=1}^{n} \alpha_i f_i(x) \tag{1.1}$$

where α_i is the weight assigned to the base models prediction $f_i(x)$. The simplest approach to determining the values of α_i is to set them to the same value. This is known as the Base Ensemble Method (BEM). More advanced approaches try to set the weights so as to minimize the mean square error of the training data. Merz and Pazzani [12] provide an extensive description of these techniques.

Model *selection* simply chooses the best "base" model to make a prediction. This can be either done in a static fashion using cross validation majority [15] where the best model is the one that has the lowest training error. Alternatively it can be done in a dynamic fashion [4,11,13] where based on finding "close" instances in the training data to a test instance, a base model is chosen which according to certain criteria is believed will give the best prediction. The advantage of this approach is based on the rationale that one model may perform better than other learning models in a localised region of the instance space even if, on average over the whole instance space, it performs no better than the others.

An alternative strategy to model integration is to build a meta-model to select/combine the outputs from base models. The original and most widely used meta-technique is referred to Stacking. Stacking was introduced by Wolpert [18] and was shown theoretically by LeBlanc and Tibshirani [9] to be a bias reducing technique. In Stacked Regression (SR), the base models produce meta-instances consisting of the target value and the base models' predictions, created by running cross validation over the training data. The meta-data is used to build a meta-model, based on a regression algorithm and the base models are built using the whole training data. Ensemble prediction for a test Instance is formed by passing a meta test instance (formed from the base models' predictions) to the meta-model. Typically the generation of the base models is heterogeneous or homogeneous but built with different training parameters. Breiman [1] investigated the use of Linear Regression to form the meta-model and found that Linear Regression is a suitable meta-model so long as the coefficients of regression are constrained to be non-negative.

More recent meta-approaches for classification are the Dynamic Integration techniques developed by Puuronen and Tsymbal [13,16] Similar to Stacking, these perform a cross-validation history during the training phase. However meta-instances are formed consisting of the training instance attribute values and the error for each model in predicting its target value. During the test phase a lazy meta-model based on weighted nearest neighbours uses the meta-data to either dynamically select or combine models for a test instance in the application phase. In the Methodology section we describe in detail the DI techniques and the modifications required to make them applicable for regression. In this paper, we compare the accuracy of ensemble techniques of SR and DI over a range of data-sets. It is particularly apposite to compare SR to the variants of DI as there strong similarities in their approach in that they accumulate meta-data based on a cross validation history which is then used to build a meta-model.

3 Methodology

In this section we describe the DI classification algorithms and their regression variants. DI consists of 3 techniques Dynamic Selection, Dynamic Voting and Dynamic Voting with Selection. We refer to their regression counterparts as Dynamic Selection, Dynamic Weighting and Dynamic Weighting with Selection. Dynamic Selection makes a localized selection of a model based on which model has the lowest cumulative error for the nearest neighbours to the test instance. The procedure for regression remains the same. Dynamic Voting assigns a weight to each base model based on its localized performance on the NN set and the final classification is based on weighted voting. Dynamic Weighting (DW) is similar to the Dynamic Voting in its calculation of weights but the final prediction is made by summing each of the base models predictions weighted by a normalized weight value. Dynamic Weighting with Selection (DWS) is a regression derivative of Dynamic Voting with Selection. The process is similar to Dynamic Weighting except that base model with cumulative error in the upper half of the error interval, $E_i > (E^{\max} - E^{\min})/2$, (where $E^{\max}$ is the largest cumulative error of any model and $E^{\min}$ is the lowest cumulative error of any model) are discarded from adding to the prediction.

In [16] the ensemble generation is improved upon by a feature selection method based on hill climbing. In this paper, we consider a method that tries to reduce the size of the ensemble set whilst maintaining its accuracy. During the training phase, we start with a set of N base models, and due to a consideration of their training accuracy and diversity determined by the cross-validation process (intrinsic to both the SR and the DI techniques) reduce the size of the set down to M base models. This process of filtering down the number of models is based on the pseudo-code described in Figure 1 and adds little algorithmic overhead to the techniques. Its goal is to remove members from the ensemble that are considered too inaccurate to be effective and then to consider the remaining members based on both their accuracy and diversity.

$E_i(x)$ = training error for instance x in training data

$E_i^{sum} = \sum_{x \in Train} E_i(x)$ - total training error for model i

E_{min} – minimum total training error for any model

$$accuracy_i = \frac{E_{min}}{E_i^{sum}}$$

```
for (i=1 to N)
    if  (accuracy_i > accuracy_threshold) then
            discard model i
    endif
endfor
n = the number of models remaining in the ensemble
//models are re-indexed from 1..n
if n>M then
// determine diversity
    for ( i = 1 to n)
            count = 0
            for (j=1 to n)
            if ( i<>j AND correl(E_i, E_j) > 0.6) then
                    count = count + 1;
            endif
    endfor
    diversity_i= (n-count)/n
    acc+div_i= accuracy_i+diversity_i
endfor
// take the top M base models based on those having highest acc+div_i
// measure
```

Fig. 1. Ensemble size reduction technique.

4 Experimental Setup

The base models and ensemble techniques were assessed using 10 fold cross validation and the mean absolute error (MAE) was recorded for each technique. 15 data-sets were chosen from the WEKA repository [18]. The data-sets were chosen as they represent real world data and not artificial regression examples. The data-sets were pre-processed to remove missing values using a mean or modal technique. The two base models used were 5-NN and Linear Regression. We assessed the improvement in accuracy or otherwise of the ensemble in comparison to the base model by using a two tailed paired t-test (p=0.05). For each technique, the ensemble set was generated using the Random sub-space method (RSM) first proposed by Ho [5,6] for classification problems and is a derivative of Stochastic Discrimination [7]. Random sub-space method is a generation method where each base model is built from the training data which has been transformed to contain different random subsets of the variables. We chose the model tree technique M5, which combines instance based learning with regression trees [14] as the meta-model for SR. We chose this as it has a larger hypothesis space than simple linear regression. In the experiments where the ensemble

size was reduced the initial ensemble set had size $N = 25$ and was reduced to $M = 10$ with an `accuracy_threshold` of 0.66. Each of the DI techniques used distance weighted 5-NN as their meta-model.

5 Experimental Results

This section is divided into two sections where each section consists of two experiments; the first is related to the accuracy of the ensembles for the whole ensemble set; the second assesses the accuracy of the ensemble to the experiments when the ensembles are reduced in size. Each section consists of the results of the comparison with the base model of LR and 5-NN respectively. The results of each experiment over the 15 data-sets is presented in the form of a table where the first column gives the name of the data-set, the second column the base models' MAE ± standard deviation for each data-set, and column 3-6 gives the MAE for each ensemble technique. The remaining column records the technique with the least MAE, if any of the techniques were able to significantly improve upon the performance of the base model, otherwise the entry is left blank. An ensemble MAE result which is significantly better than the base model is shown in bold, if it is significantly worse it is shown underlined. An adjunct table summarizes the results of the significance comparison in the form of *wins/draws/losses* where *wins* is the number of data-sets where the ensemble outperformed the base model, *draws* is the number of data-sets for which the base model showed no significant difference in accuracy to the base model, and *losses* is the number of data-sets where the ensemble accuracy was worse than the base model.

5.1 Whole Ensemble Set

This section refers to experimental results involving the whole ensemble set. Table 1 shows the results of the comparison when the base model was Linear Regression. DS and DWS reduced the error significantly for the greatest number of data-sets whereas DW reduced the error for the least number. However both SR and DS increased MAE significantly for two data-sets. Only DWS never increased the MAE significantly for any of the 15-data-sets. If one considers the "least MAE" column it is clear than for 7 data-sets none of the techniques were effective. For the other 8 data-sets, if we rank the order in which the ensemble technique gave the least error most frequently, DS came first with SR second.

Table 2 shows the results of the comparison of ensembles when the base model was 5-NN. Clearly the two outstanding ensemble techniques were DW and DWS, which both reduced the error signicantly for 13 out of the 15 data-sets. The technique which proved least effective was DS. The "least error" column shows that for every data-set at least one of the ensemble techniques was effective in signifcantly reducing the error. DWS came first in rank order of the techniques which gave the least error most frequently with DW coming second.

Table 1. The comparison of ensembles using LR as the base model.

Data-set	LR	SR	DS	DW	DWS	Least MAE
abalone	1.58±0.08	1.61±0.28	1.58±0.09	1.62±0.11	1.58±0.10	-
autohorse	7.99±4.17	**6.54±4.11**	7.18±5.16	**6.84±4.13**	**6.65±4.23**	SR
autompg	2.23±0.21	2.11±0.48	**2.05±0.26**	2.16±0.21	**2.05±0.25**	DS/DWS
autoprice	1974.23± 326.81	**1659.33± 290.68**	**1518.58± 339.96**	**1660.29± 367.28**	**1532.65± 357.1**	DS
auto93	3.79±1.3	4.11±1.6	4.02±1.17	3.20±1.23	3.25±1.28	-
bodyfat	0.53±0.23	0.53±0.22	**0.43±0.26**	0.60±0.21	**0.48±0.24**	DS
breastTumor	7.97±1.05	8.1±1.05	8.06±0.99	7.77±0.93	7.84±0.89	-
cholesterol	39.24±5.88	40.89±5.73	38.92±4.64	38.19±4.62	38.41±4.44	-
cloud	0.26±0.09	0.26±0.09	0.32±0.08	0.27±0.09	0.26±0.09	-
cpu	35.02±4.45	**14.22±6.75**	**22.25±7.31**	**21.24±8.05**	**19.36±7.13**	SR
housing	3.41±0.33	**2.82±0.58**	**2.68±0.47**	3.29±0.56	**2.96±0.54**	DS
lowbwt	364.48± 48.21	392.01± 57.4	397.93± 51.48	356.87± 62.64	363.03± 61.44	-
sensory	0.61±0.04	0.61±0.04	**0.59±0.05**	0.61±0.06	0.59±0.06	-
servo	0.63±0.273	**0.38±0.23**	**0.45±0.28**	0.63±0.22	**0.44±0.25**	SR
strike	221.43± 38.47	**209.79± 41.65**	**180.84± 45.72**	**203.64±38.33**	**189.01± 42.54**	DS

Method	SR	DS	DW	DWS
Wins/Ties /Losses	6/7/2	8/5/2	4/9/2	8/7/0

Table 2. The comparison of ensembles using 5-NN as the base model.

Data-set	5-NN	SR	DS	DW	DWS	Least MAE
abalone	1.61±0.09	**1.54±0.08**	1.73±0.07	**1.54±0.09**	**1.54±0.09**	*SR/DW/DWS*
autohorse	8.7±4.69	**7.11±3.71**	**5.79±4.57**	**6.44±4.93**	**6.06±4.92**	*DS*
autompg	2.31±0.38	**2.12±0.34**	2.41±0.35	**2.04±0.35**	**2.08±0.35**	*DW*
autoprice	1531.86± 404.24	1478.62± 460.89	1382.06± 336.79	**1438.39± 460.48**	**1397.63± 454.43**	*DWS*
auto93	3.81±1.4	3.76±1.11	4.27±1.32	**3.4±1.52**	**3.39±1.58**	*DWS*
bodyfat	2.3±0.49	**0.94±0.21**	**1.16±0.28**	**1.7±0.37**	**1.407±0.34**	*SR*
breastTumor	9.39±1.04	**8.38±0.64**	9.67±1.06	**8.01±0.91**	**8.12±0.97**	*DW*
cholesterol	43.0±4.13	43.39±4.03	46.17±6.04	**39.64±4.63**	**40.36±4.56**	*DW*
cloud	0.51±0.19	**0.38±0.13**	**0.39±0.11**	**0.39±0.17**	**0.36±0.14**	*DWS*
cpu	22.72±13.94	34.16±17.61	23.97±12.97	**19.68±13.46**	20.72±14.18	*DW*
housing	2.59±0.58	**2.30±0.41**	2.56±0.39	**2.39±0.55**	**2.27±0.5**	*DWS*
lowbwt	398.3±80.6	397.8±47.17	471.35± 67.56	365.88± 80.53	369.47± 74.71	*DWS*
sensory	0.6±0.06	**0.55±0.06**	0.66±0.07	**0.58±0.05**	**0.58±0.05**	*SR*
servo	0.56±0.19	**0.38±0.30**	**0.42±0.24**	0.62±0.22	**0.42±0.22**	*SR*
strike	194.62± 53.46	222.29± 46.7	196.71± 50.16	**182.25± 50.01**	**176.08± 50.15**	*DWS*

Method	SR	DS	DW	DWS
Wins/ Draws/losses	9/3/3	4/7/4	13/2/0	13/2/0

In summary, it can be seen that for either base model, at least one of the DI techniques is as effective as SR, if not more so in reducing the error. Also DWS seemed to be the most reliable ensemble approach, as it never significantly increased the error. The pattern of behaviour of the DI techniques for regression mirrors that of

classification [16] where the best integration method varied with the data-set and the base model.

5.2 Reduced Ensemble Set

In this section, we repeated the experiments of the previous section, but with the addition that the ensemble set had been reduced at the end of the training phase using the algorithm described in Figure 1 from N = 25 to M =10 . Table 3 shows the results of the comparison of the reduced size ensembles for LR. Comparing the ties/wins/losses of Table 3 to Table 1 shows that DW and DS improved in performance, DWS remained the same and SR remained approximately the same.

Table 3. Results of comparison of ensembles using LR.

Data-set	LR	SR	DS	DW	DWS
abalone	1.58±0.08	**1.52±0.06**	1.57±0.09	1.60±0.09	1.58±0.09
autohorse	7.99±4.17	7.42±3.4	7.00±4.75	**6.57±3.69**	**6.2±3.87**
autompg	2.23±0.21	2.09±0.43	**2.09±0.33**	**2.13±0.2**	**2.08±0.24**
autoPrice	1974.23± 326.81	**1687.68± 233.37**	**1550.61± 343.32**	**1723.99± 324.96**	**1567.55± 356.56**
auto93	3.79±1.3	3.521±1.1	3.91±1.24	3.43±1.33	3.41±1.39
bodyfat	0.53±0.23	0.48±0.23	**0.41±0.27**	**0.45±0.25**	**0.42±0.26**
breastTumor	7.97±1.05	8.08±1.09	8.06±0.99	7.92±0.95	7.97±0.89
cholesterol	39.24±5.88	38.94±5.76	39.01±4.96	39.05±4.63	39.03±4.49
cloud	0.26±0.09	0.28±0.09	0.28±0.10	0.27±0.10	0.27±0.10
cpu	35.02±4.45	**15.35±6.8**	**24.27±6.01**	**25.05±7.2**	**23.07±7.09**
housing	3.41±0.33	**2.76±0.37**	**2.67±0.45**	3.17±0.42	**2.79±0.47**
lowbwt	364.48±48.21	365.46±50.86	376.69±42.86	365.19±44.73	361.59±49.06
sensory	0.61±0.04	0.61±0.04	**0.59±0.04**	0.60±0.05	0.59±0.05
Servo	0.63±0.27	**0.44±0.2**	**0.5±0.23**	**0.53±0.27**	**0.48±0.27**
Strike	221.43±38.47	218.47±37.39	**205.04±36.68**	**212.62±33.99**	**205.39±35.35**

Method	SR	DS	DW	DWS
Wins/ Ties/losses	5/10/0	8/7/0	7/8/0	8/7/0

There is however more variation in the results than the summary in significance comparison alone would suggest. If we calculate the percentage change in MAE between the results in Table 1 and Table 3 and average it over all data-sets, the following average percentage changes are shown in Table 4. A positive value is recorded if the technique gave on average a percentage reduction in error.

It is clear that although the average change in MAE is quite small no larger than a 2% decrease , the standard deviation is relatively large indicating that for some data-sets there is a large percentage change in the MAE.

Table 4. Percentage change in MAE for ensemble size from N to M.

Technique	SR	DS	DW	DWS
Average percentage change in MAE	-0.45±8.3	-0.72±6.36	0.9±9.89	-1.41±7.41

However comparing the reduced ensemble set to the whole ensemble results in detail shows a general trend that for data-sets where the error increased it did not increase to change the level of significance, but where the error decreased then in some cases it did change the signifcance comparison. e.g. consider the technique DW , for the whole ensemble set, autohorse, autoprice, cpu, strike gave an MAE better than the base model whereas abalone, and bodyfat were significantly worse. For the reduced ensemble set, autohorse, autompg, autoprice, bodyfat, cpu, servo, strike gave an MAE significantly better than base model even though for some of these data-sets there was a relative increase in MAE.

Table 5. Comparison of Ensembles with the base model 5-NN.

Data-set	5-NN	SR	DS	DW	DWS
abalone	1.61±0.09	**1.56±0.09**	1.756±0.07	**1.57±0.09**	1.589±0.09
autohorse	8.7±4.69	**6.14±3.71**	**4.88±4.92**	**4.76±4.39**	**4.77±4.72**
autoMpg	2.31±0.38	**2.13±0.34**	2.34±0.39	**2.00±0.39**	**2.06±0.41**
autoPrice	1531.86± 404.24	1383.66± 460.89	**1320.41± 236.91**	**1313.78± 462.28**	**1326.41± 419.81**
auto93	3.81±1.4	3.61±1.11	*4.28±1.29*	3.5±1.6	3.56±1.58
bodyfat	2.3±0.49	**0.95±0.21**	**1.07±0.28**	**1.03±0.32**	**1.04±0.28**
breastTumor	9.39±1.04	**8.24±0.64**	9.71±0.99	**7.99±0.91**	**8.32±1.0**
cholesterol	43.0±4.13	**39.81±4.03**	46.77±6.89	**40.13±4.72**	41.21±4.77
cloud	0.51±0.19	**0.37±0.13**	0.37±0.10	**0.33±0.14**	**0.34±0.12**
cpu	22.72±13.94	30.97±17.61	21.10±13.33	**20.12±12.48**	21.33±12.95
housing	2.59±0.58	**2.33±0.41**	2.57±0.5	**2.33±0.48**	**2.24±0.41**
lowbwt	398.3±80.6	375.46±47.17	430.79±83.64	**359.39±58.65**	**367.66±64.31**
sensory	0.6±0.06	**0.56±0.06**	0.64±0.08	**0.57±0.05**	0.58±0.06
servo	0.56±0.19	**0.38±0.3**	**0.44±0.28**	**0.45±0.29**	**0.39±0.3**
strike	194.62±53.46	208.72±46.7	195.78±61.88	**187.14±52.46**	185.92±56.27

Method	SR	DS	DW	DWS
Ties/Wins/losses	10/3/2	4/9/3	14/1/0	9/6/0

Table 5 shows the comparison of the reduced ensemble sets when the base model was 5-NN. Comparing the results to Table 2 shows that SR, DS, DWS performed slightly better with the reduced sets. DWS showed a drop of 4 from 13 to 9 data-sets showing a significant improvement in MAE. The average percentage change in MAE for the whole ensemble set and the reduced ensemble set for 5-NN is shown in Table 6. However for those 4 data-sets which were no longer significantly better with DWS than the base model, the percentage change in MAE was at most 5.6%. The same pattern of average error change is similar to LR with a low average pecentage change but a higher level of variability in percentage change amongst the data-sets, as shown in Table 6. The main difference to the results for LR is that for all techniques there was a positive change in the average percentage change in error, with a relatively large change for DW.

Table 6. Percentage change in MAE for ensemble size = N to M.

Technique	SR	DS	DW	DWS
Average percentage change in MAE	3.49±4.69	3.53±5.61	7.42±13.23	3.04+9.12

In summary, the ensemble size reduction strategy maintains the effective-ness both of SR and the DI techniques. In the case DW, the results would suggest that in fact pruning the ensemble set actually improves accuracy, a likely consequence that it is more sensitive to in-accurate or redundant base models, than the DS and DWS approaches, which either select the best model or remove inaccurate models from the model combination.

6 Conclusions and Future Work

In this paper we have demonstrated that the classification ensemble techniques of Dynamic Integration can be adapted to the problem of regression. We have shown that for simple base models, these techniques are as effective as Stacked Regression for the range of data-sets tested. We have presented a extension to the SR and DI techniques which uses the accuracy and diversity measure captured in the training of the base models to prune the size of the ensemble thus removing models that are ineffective in the model combination. We intend to refine and improve on this simple technique as it provides little extra overhead to the algorithms and has shown promising results in reducing the ensemble size whilst maintaining its level of accuracy. In particular, we intend to investigate in more detail the appropriate choice of accuracy threshold and the size of the reduced ensemble set. Also, we shall compare our measure for diversity to the more commonly known measures for diversity such as the variance based measure developed in [8].

Acknowledgements

We gratefully acknowledge the WEKA framework [16] within which the ensemble methods were implemented and assessed. Dr Tsymbal acknowledges Science Foundation, Ireland.

References

1. Breiman, L. 1996. Stacked Regression. Machine Learning, 24:49-64.
2. Christensen, S. 2003. Ensemble Construction via Designed Output Distortion In *Proc. 4th International Workshop on Multiple Classifier Systems,* LNCS, Vol. 2709, pp. 286-295, Springer-Verlag.
3. Dietterich, T. 2000. Ensemble Methods in Machine Learning, *In Proc. 1st International Workshop on Multiple Classifer Systems,* LNCS, Vol 1857, pp. 1-10, Springer-Verlag.
4. Giacinto, G. and Roli, F. 2000. Dynamic Classifier Selection, In *Proc. 1st Int. Workshop on Multiple Classifier Systems,* LNCS, Vol 1857, pp. 177-189, Springer-Verlag.
5. Ho, T. K. 1998a. The random subspace method for constructing decision forests. *IEEE PAMI,* 20(8):832--844.

6. Ho, T.K. 1998b. Nearest Neighbors in Random Subspaces, LNCS: *Advances in Pattern Recognition,* 640-648.
7. Kleinberg, E.M. 1990. Stochastic Discrimination, *Annals of Mathematics and Artificial intelligence,* 1:207-239.
8. Krogh, A. and Vedelsby, J. 1995. Neural Networks Ensembles, Cross validation, and Active Learning, *Advances in Neutal Information Processing Systems,* MIT Press, pp. 231-238.
9. Kuncheva L.I. 2002. Switching between selection and fusion in combining classifiers: An experiment, *IEEE Transactions on SMC,* Part B, 32 (2),, 146-156.
10. LeBlanc, M. and Tibshirani, R. 1992. Combining estimates in Regression and Classification, Technical Report, Dept. of Statistics, University of Toronto.
11. Merz, C.J. 1996. Dynamical selection of learning algorithms. In *Learning from data, artificial intelligence and statistics.* .Fisher and H.-J.Lenz (Eds.) New York: Springer.
12. Merz, C. and Pazzani, M. 1999. A principal components approach to combining regression estimates, *Machine Learning,* 36:9-32.
13. Puuronen, S., Terziyan, V., Tsymbal, A. 1999. A Dynamic Integration Algorithm for an Ensemble of Classifiers. *Foundations of Intelligent Systems, 11th International Symposium ISMIS'99,* LNAI, Vol. 1609: 592-600, Springer-Verlag.
14. Quinlan, R. 1992. Learning with continuous classes, In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence,* World Scientific, pp. 343-348.
15. Schaffer, C. 1993. Overfitting avoidance as bias. *Machine Learning* 10:153-178.
16. Sharkey, A.J C. (Ed.) 1999. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems.* Springer-Verlag,.
17. Tsymbal, A., Puuronen, S., Patterson, D. 2003. Ensemble feature selection with the simple Bayesian classification, *Information Fusion* Vol. 4:87-100, Elsevier.
18. Witten, I. and Frank, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations,* Morgan Kaufmann.
19. Wolpert, D. 1992. Stacked Generalization, *Neural Networks* 5, pp. 241-259.

Dynamic Classifier Selection by Adaptive k-Nearest-Neighbourhood Rule

Luca Didaci and Giorgio Giacinto

Dipartimento di Ingegneria Elettrica ed Elettronica
Università degli Studi di Cagliari
Piazza D'Armi – 09123 Cagliari, Italy
{giacinto,luca.didaci}@diee.unica.it

Abstract. Despite the good results provided by Dynamic Classifier Selection (DCS) mechanisms based on local accuracy in a large number of applications, the performances are still capable of improvement. As the selection is performed by computing the accuracy of each classifier in a neighbourhood of the test pattern, performances depend on the shape and size of such a neighbourhood, as well as the local density of the patterns. In this paper, we investigated the use of neighbourhoods of adaptive shape and size to better cope with the difficulties of a reliable estimation of local accuracies. Reported results show that performance improvements can be achieved by suitably tuning some additional parameters.

1 Introduction

An alternative to using a single, highly optimised classifier is to "fuse" the outputs of multiple classifiers into a so-called multiple classifier system (MCS) [1-4]. If the classifiers make complementary decisions, the use of an MCS will improve the classification accuracy with respect to that of individual classifiers. Two main strategies for classifier fusion have been proposed in the literature, namely classifier *combination* and classifier *selection* [5-8].

The rationale behind the classifier combination approach can be roughly traced back to the strategies for collective decisions. As a result, the final classification output depends on the output of the combined classifiers. As all the classifiers contribute to the classification label, these kinds of techniques implicitly assume that combined classifiers are "competent" in the entire features space.

On the other hand, the classifier selection approach assume that each classifier has a region of competence in some area of the feature space, i.e. it exhibits the lower error rate for patterns belonging to that region. Classifier selection is thus aimed at assigning each classifier to its region of competence in the feature space, thus exploiting the complementarities among classifiers. Two types of classifier selection techniques have been proposed in the literature. One is called *static* classifier selection, as the regions of competences are defined prior to classifying any test pattern [5]. The other one is called *dynamic* classifier selection (DCS) as the regions of competence are determined on the fly, depending on the test pattern to be classified [7, 8]. In the DCS based on local accuracy, each test pattern $\mathbf{x}^*$ is labelled by the classifier with the highest accuracy computed in a "neighbourhood" of $\mathbf{x}^*$. In particular, the *k*-nearest

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 174–183, 2004.

neighbour technique is used to estimate the accuracy of each classifier in the neighbourhood of $\mathbf{x}^*$. These estimates can then be used to predict the expertise of each classifier, and thus the related regions of competence.

While a number of experimental results showed the effectiveness of this approach [5, 9], the performances of DCS are much lower than that of the *oracle,* i.e. of an ideal selector that always select the optimal classifier. Such a limitation may be the consequence of some characteristics of the basic *k*-nn rule that have been thoroughly studied in the past years [10]. In particular, we focussed on the choice of the more suited distance metric to compute the neighbourhood, and the choice of the suitable value of *k.* The metric determines the shape of the neighbourhood, while the suitable value of *k* is related to the local density of patterns.

In Section 2 we briefly recall the basic steps of DCS algorithm in order to highlight some critical points that motivated the present study. Section 3 is devoted to the proposal of two modifications of the DCS algorithm, one related to the distance measure used to compute the neighbourhood of the test pattern, the other related to the dynamic choice of the most suitable value of *k.* We performed a number of experiments on a hyperspectral remote-sensing data set, which are reported in Section 4. The results show that the proposed modifications of the basic DCS algorithm allow for improvements in accuracy. At the same time, they suggest that further study is needed to bridge the gap between the accuracy of DCS and that of the *oracle.*

2 DCS Algorithm

For each unknown test pattern $\mathbf{x}^*$, let us consider a local region $\Re_k(\mathbf{x}^*)$ of the feature space defined in terms of the *k*-nearest neighbours in the validation data. Validation data are extracted from the training set and are not used for classifier training. In order to define this *k*-nn region, the distance between $\mathbf{x}^*$ and patterns in the validation data are calculated by a metric Σ. As in the *k*-nearest neighbour classifier, the appropriate size of the neighbourhood is decided by experiment.

It is easy to see that the classifier local accuracy (CLA) can be estimated as the ratio between the number of patterns in the neighbourhood that were correctly classified by the classifier C_j, and the number of patterns forming the neighbourhood of $\mathbf{x}^*$ [7, 8]. We will use these estimates to predict which single classifier is most likely to be correct for a given test pattern.

Let $\mathbf{x}^*$ be the test pattern and N the number of classifiers

1. If all the classifiers assign $\mathbf{x}^*$ to the same class, then the pattern is assigned to this class
2. Compute $CLA_j(\mathbf{x}^*)$, $j=1, ..., N$ (see below for details)
3. Identify the classifier C_m exhibiting the maximum value of $CLA(\mathbf{x}^*)$
4. If $CLA_m(\mathbf{x}^*)>CLA_j(\mathbf{x}^*)\ \forall\ j \neq m$, select C_m; else select the most globally accurate classifier among those exhibiting the maximum value of CLA.

In the following section, we will refer to this DCS algorithm in order to investigate the effect on the classification performance of two design parameters, namely the dimension of the neighbourhood *k,* and the distance metric Σ used to define the *k*-nn neighbourhood.

2.1 Local Accuracy "a priori"

The CLA "a priori" - a.k.a. "overall local accuracy" [7, 8] – is the percentage of validation samples in the local region $\Re_k(\mathbf{x}^*)$ of $\mathbf{x}^*$ that are correctly classified by the classifier C_j. In this case, CLA is computed "a priori", that is, without taking into account the class assigned by the classifier C_j to the test pattern $\mathbf{x}^*$.

For each classifier C_j the probability of classifying correctly the test pattern can be computed as follows:

$$\hat{p}(correct_j) = k_j / k \tag{1}$$

where k_j is the number of "neighbouring" patterns that were correctly classified by the classifier Cj and k is the number of patterns in the neighbourhood $\Re_k(\mathbf{x}^*)$.

2.2 Local Accuracy "a posteriori"

The estimation of the CLA "a posteriori" - a.k.a. "local class accuracy" [7, 8] – exploits the information on the class assigned by the classifier C_j to the test pattern $\mathbf{x}^*$, that is, $C_j(\mathbf{x}^*) = \omega_k$. Thus equation (1) can be rewritten as follows:

$$\hat{p}\left(correct_j / C_j(\mathbf{x}^*) = \omega_k\right) = \hat{p}\left(\mathbf{x}^* \in \omega_k / C_j(\mathbf{x}^*) = \omega_k\right) = N_{kk} \Big/ \sum_{j=1}^{M} N_{jk} \tag{2}$$

where N_{kk} is the number of neighbourhood patterns of $\mathbf{x}^*$ that have been *correctly* assigned by Cj to the class ω_k, and $\sum_{j=1}^{M} N_{jk}$ is the total number of neighbourhood patterns that have been assigned to the class ω_k by C_j. In this case, CLA is computed "a posteriori", that is, after each classifier C_j produces the output on the test pattern $\mathbf{x}^*$.

3 A Modified DCS Algorithm

As CLA estimates are based on the k-nn rule, they are liable to the same limitations that stimulated a number of improvements to the basic k-nn algorithm [10]. In the present paper, we will focus our attention on two issues: i) the k-nn estimate assumes that the local accuracy is almost constant in the local region $\Re_k(\mathbf{x}^*)$, otherwise the k-nn estimate is only asymptotically correct; ii) the reliability of the local accuracy estimate depends on the number of patterns included in the neighbourhood $\Re_k(\mathbf{x}^*)$. These issues are influenced respectively by the shape and size of $\Re_k(\mathbf{x}^*)$. The shape of the neighbourhood $\Re_k(\mathbf{x}^*)$ depends on the distance metric used, while its size depends on the choice of the k parameter. It is easy to see that the size of $\Re_k(\mathbf{x}^*)$ - for a given value of k - depends on the density of patterns surrounding $\mathbf{x}^*$.

Among the two DCS approaches summarised in sections 2.1 and 2.2, the DCS based on the "a posteriori" CLA may be affected by the above issues more deeply

than the "a priori" CLA, because the "a posteriori" CLA is computed on the subset of patterns belonging to the neighbourhood of $\mathbf{x}^*$ that received the same label assigned to $\mathbf{x}^*$. On the other hand, reported experiments show that the "a posteriori" computation of CLA provides higher accuracies than those provided by the "a priori" CLA [7, 8]. For these reasons, it is worth investigating the effect of an adaptive *k*-nn rule on the performances of the "a posteriori" DCS in order to bridge the gap with respect to the performances of the oracle. In particular, we propose two enhancements to the basic DCS algorithm inspired by their counterparts in the *k*-nn literature, namely an adaptive metric for neighbour computation, and a dynamic choice of the value of *k*.

In order to explain the rationale behind the proposed enhancements, it is worth recalling that the goal of DCS by CLA is to sort out the "best" classifier in the neighbourhood of $\mathbf{x}^*$. To this end, the feature space can be considered as divided by each classifier into a number of "region of competence" and a number of "region of lack of competence". In the first one the classifier makes the correct decision, in the second one the classifier makes the wrong decision [8] (see fig 1).

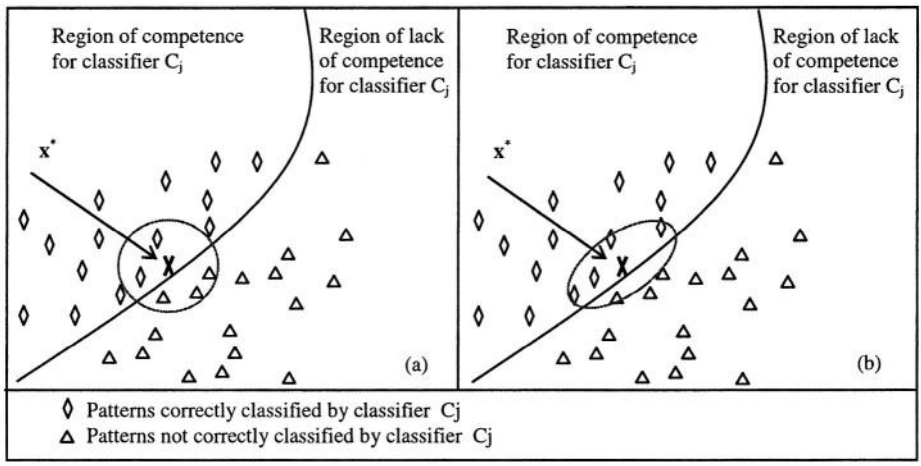


Fig. 1. Test pattern $\mathbf{x}^*$ and its 5-nn. (a) Use of the Euclidean metric. Patterns belonging to different regions are included in the 5-nn. (b) Use of an 'adaptive' metric. Most of the patterns included in the 5-nn belong to the same region which $\mathbf{x}^*$ belongs to.

The DCS by local accuracy estimate (Sect. 2.2) assumes that it is possible to estimate the accuracy of classifier C_j in $\mathbf{x}^*$ through the accuracy of C_j in a particular neighbourhood $\Re_k(\mathbf{x}^*)$ of $\mathbf{x}^*$. In order to ensure that this estimate is correct, it is important that patterns used in this estimate belong to the same region – "competence" or "lack of competence" – as $\mathbf{x}^*$. Therefore the neighbourhood of $\mathbf{x}^*$ should be prevalently contained in the region which $\mathbf{x}^*$ belongs to. In the *k*-nn rule for classification, this is analogous to assume that locally the posterior probabilities are almost constant.

Figure 1 shows a simple example, where a boundary between regions of competence is traced. Accordingly, the validation patterns are labelled as correctly classified or incorrectly classified. The test pattern $\mathbf{x}^*$ in Fig. 1(a) lies close to the boundary of

the region of competence. A k-nn neighbourhood $\Re_k(\mathbf{x}^*)$ using the Euclidean metric and k=5 is shown. The computation of the CLA in $\mathbf{x}^*$ may result in an inaccurate estimate because it cannot be assumed that CLA is approximately constant in $\Re_k(\mathbf{x}^*)$, as patterns in $\Re_k(\mathbf{x}^*)$ belong to a region of competence as well as a region of lack of competence. A more accurate estimate of CLA in $\mathbf{x}^*$ requires that $\Re_k(\mathbf{x}^*)$ is contained in the region of competence, as shown in Fig. 1(b).

To attain more accurate estimates of CLA, we propose three techniques for dynamically adjusting the shape and size of $\Re_k(\mathbf{x}^*)$.

3.1 DANN Metric for CLA Computation

The DANN metric, proposed by Hastie and Tibshirani [11] for the k-nn classification task, provided a solution to the above problem. This technique is based on the exploitation of the results of the Linear Discriminant Analysis (LDA) performed on a set of $\mathbf{K}_m$ patterns ($\mathbf{K}_m > k$) in a nearest neighbourhood of $\mathbf{x}^*$. This result is used to reshape the nearest neighbourhood $\Re_k(\mathbf{x}^*)$, so that class posterior probabilities are more homogenous in the modified neighbourhood. In order to use the DANN algorithm for CLA computation, the validation patterns are labelled according to the classification result, that is, correct or incorrect. In particular, we proposed a modified DANN algorithm for CLA computation, that can be summarised in the following steps:

1. For each classifier $\mathbf{C}_j$ and for each validation pattern $\mathbf{x}_i$, set the meta-label "correctly classified" or "incorrectly classified "
2. Spread out a nearest neighbourhood of $\mathbf{K}_m$ points around the test point $\mathbf{x}^*$.
3. Let $\mathbf{I}(\mathbf{x}^*, \mathbf{K}_m, \mathbf{C}_j)$ be the set of validation patterns in the $\mathbf{K}_m$ neighbourhood that satisfy $\mathbf{C}_j(\mathbf{x}_i) = \mathbf{C}_j(\mathbf{x}^*)$
4. For each $\mathbf{x}_i \in \mathbf{I}(\mathbf{x}^*, \mathbf{K}_m, \mathbf{C}_j)$, compute the following quantities:
   - the within-class covariance matrix
   $$\mathbf{W} = \sum_{x \in correct} w_i (x_i - \bar{x}_{correct})(x_i - \bar{x}_{correct})^T + \sum_{x \in incorrect} w_i (x_i - \bar{x}_{incorrect})(x_i - \bar{x}_{incorrect})^T$$
   - the between class covariance matrix
   $$\mathbf{B} = \pi_{correct} (\bar{x}_{correct} - \bar{x})(\bar{x}_{correct} - \bar{x})^T + \pi_{incorrect} (\bar{x}_{incorrect} - \bar{x})(\bar{x}_{incorrect} - \bar{x})^T$$
   where:
   - w_i are the weights assigned to the i-th observation, function of the distance $d(\mathbf{x}_i, \mathbf{x}^*)$,
   - $\bar{x}_{correct}$ and $\bar{x}_{incorrect}$ are the weighted means of patterns correctly and incorrectly classified respectively.
   - $\pi_{correct} = \sum_{correct} w_i / \sum w_i$; $\pi_{incorrect} = \sum_{incorrect} w_i / \sum w_i$
5. Define a new metric $\mathbf{\Sigma} = \mathbf{W}^{-1/2} [\mathbf{W}^{-1/2} \mathbf{B} \mathbf{W}^{-1/2} + \varepsilon \mathbf{I}] \mathbf{W}^{-1/2}$
6. Use the metric $\mathbf{\Sigma}$ to calculate a new k-nearest neighbourhood and the adaptive local classifier accuracy

The resulting metric $\mathbf{\Sigma}$ weighs the distances $d^2(\mathbf{x}_i, \mathbf{x}^*) = (\mathbf{x}_i - \mathbf{x}^*)^T \mathbf{\Sigma} (\mathbf{x}_i - \mathbf{x}^*)$ in agreement with the local densities of correctly classified and incorrectly classified

patterns. As a result, the $\Re_k(\mathbf{x}^*)$ is stretched in the directions of low variability of CLA, and shrunk in the directions of high variability of CLA. Thus, $\Re_k(\mathbf{x}^*)$ should be contained in a region with almost constant CLA. Further details on the DANN algorithm can be found in [11].

3.2 Simplified DANN for CLA Computation

Given that the task at hand is a two-class problem, where the classes are related to correct or incorrect classification, we propose a simplified version of the DANN metric. Our aim is to obtain a metric whose "unit sphere" is an ellipsoid whose minor axis is approximately orthogonal to the boundary between regions of competence and lack-of-competence. We can define a metric Σ with eigenvectors $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_d$, and corresponding eigenvalues $\{\alpha_i\}$, $\alpha_1>1$, $\alpha_2=\alpha_3=\ldots=\alpha_d=1$. Let us consider again a $\mathbf{K}_m$ ($\mathbf{K}_m>k$) nearest neighbourhood of $\mathbf{x}^*$. The eigenvectors can be constructed as an orthonormal basis such that $\mathbf{V}_1$ lies in the direction linking the barycentres of "correctly classified" and "incorrectly classified" patterns. The remaining eigenvectors are computed according to the Gram-Schmidt technique. In particular, the proposed metric is built according to the following steps:

1. For each classifier $\mathbf{C}_j$ and for each validation pattern $\mathbf{x}_i$, set the meta-label "correctly classified" and "incorrectly classified "
2. Spread out a nearest neighbourhood of $\mathbf{K}_m$ points around the test point $\mathbf{x}^*$.
3. Let $\mathbf{I}(\mathbf{x}^*,\mathbf{K}_m,\mathbf{C}_j)$ be the set of validation patterns in the $\mathbf{K}_m$ neighbourhood such that $\mathbf{C}_j(\mathbf{x}_i)=\mathbf{C}_j(\mathbf{x}^*)$
4. Let $\mathbf{b}_1$ and $\mathbf{b}_2$ be respectively the barycentre of "correctly classified" patterns and the barycentre of "incorrectly classified " patterns in $\mathbf{I}(\mathbf{x}^*,\mathbf{K}_m,\mathbf{C}_j)$. Let $\mathbf{V}_1$ be the normalized vector $\mathbf{V}_1 = (\mathbf{b}_1 - \mathbf{b}_2)/\|(\mathbf{b}_1 - \mathbf{b}_2)\|$
5. Define a new metric $\Sigma = \alpha\, \mathbf{V}_1 \mathbf{V}_1^T + \sum_{j=2}^{d} \mathbf{V}_j \mathbf{V}_j^T$, where
   - $\{\mathbf{V}_i\}, i = 2...d$ are computed by the Gram-Schmidt technique, so that $\{\mathbf{V}_i\}, i = 1...d$ is an orthonormal basis for the feature space
   - $\alpha >1$ is the eigenvalue of the eigenvector $\mathbf{V}_1$; the other eigenvalues being equal to 1, so that the "unit sphere" is an ellipsoid whose minor axis is collinear to $\mathbf{V}_1$
6. Use the metric Σ to calculate a modified *k*-nearest neighbourhood.

We can choose a dynamically, $\alpha = a \,\|\, \mathbf{b}_1 - \mathbf{b}_2 \,\| + b$, where *a* and *b* are suitable parameters, or we can use a constant value.

3.3 Dynamic Choice of *k*

In the "a posteriori" method it is possible that, for a given classifier $\mathbf{C}_j$ and for a given *k,* there is no validation pattern $\mathbf{x}$ assigned to class $\mathbf{C}_j(\mathbf{x}^*)$, that is, to the same class assigned to the test pattern. In this case, it is not possible to calculate the CLA using Equation (2). In order to avoid this problem, we propose to choose *k* dynamically as follows:

1. Let N be the number of classifiers, and let $\hat{k}$ be the initial value of k.
2. Compute the CLA "a posteriori" for all classifiers; let N_t be the number of classifiers for which it is not possible to compute the CLA according to equation (2).
3. If $N_t < N/2$ or $k \geq k\_max$, set to zero the CLA for those classifiers, otherwise set $k = k + k\_step$ and iterate steps 2, 3.

In this way, we choose a $k > \hat{k}$ such that the number of classifiers for which is not possible to calculate the CLA is less than N/2. Typical values of *k_step* are in the range from 1 to 14.

4 Experimental Results

In order to test the performances of the above techniques, we used a hyperspectral remote sensing data set, extracted from an AVIRIS image taken over NW Indiana's Indian Pine test site in June 1992 [12]. This data set contains 9345 patterns subdivided into 9 spectral classes. Each pattern is described by a feature vector of 202 components. This is an interesting application for testing the proposed DCS technique because of two main reasons:

- the available data allowed to extract a validation set (2885 patterns), a training set (3266 patterns), and a test set (3194 patterns) with comparable sizes, so that all the spectral classes are represented in either sets;
- the high dimensionality of the data suggests the use of an MCS made up of classifiers trained on different subspaces.

For the task at hand, we divided the entire dataset into five subspaces, the i-th subspace containing the features i, $i+5$, $i+10$, etc. This subdivision ensures that each subspace contains a number of uncorrelated features, as adjacent channels exhibit high correlation.

For each subspace we trained a k-nn classifier using $k = 10$. It is worth noting that any other base classifier can be used to test the performance of DCS.

In the following tables, results attained on the test set are reported. In order to emphasize the effects of the proposed DCS modifications, we also report the results attained on a reduced test set (797 patterns), that is, those patterns for which at least one classifier gives the correct classification result, and at least one classifier provide an incorrect classification. In other words, the reduced test set include only those patterns for which a selection mechanism is needed.

Table 1 shows the overall accuracy provided by each classifier on the test set and on the reduced test set. As each classifier is almost 50% correct on the reduced test set, it is worth to use a selection mechanism to improve the classification accuracy.

Table 1. Overall accuracy on the complete test set and on the "reduced" test set.

	C1	C2	C3	C4	C5	Oracle
Complete test set	70.79%	70.79%	**72.38%**	71.65%	69.97%	82.97%
Reduced test set	51.19%	51.19%	**57.59%**	54.96%	47.93%	100%

Table 2 summarises the seven experiments that have been carried out in order to make a thorough evaluation of the proposed mechanisms by varying the design parameters. In particular, experiments 1 to 4 are aimed at comparing the performances of DCS when different Minkowski metrics are used. For all the experiments, we used different values of the parameters k, K_m, α, in order to evaluate the sensitivity of the accuracy.

Table 2. List of experiments.

Name	Metric	Parameters	Note
EXP1	Euclidean metric		
EXP2	Euclidean metric	k_step from 1 to 14	Dynamic choice of k (sect 3.3)
EXP3	City block distance		
EXP4	City block distance	k_step from 1 to 14	Dynamic choice of k (sect 3.3)
EXP5	Simplified DANN metric	K_m from 51 to 324 α from 4 to 100	Sect 3.2
EXP6	Simplified DANN metric	K_m from 51 to 324	Dynamic choice of α (sect 3.2)
EXP7	DANN metric	K_m from 51 to 324	Sect 3.1
Common parameters		k from 1 to 51	

Table 3 shows the best overall accuracy of the proposed mechanisms for the 7 experiment settings. For the sake of comparison, the performances of the best individual classifier and the "oracle" are also shown.

Table 3. Best overall accuracy on the entire test set and on the reduced test set. The difference in respect to the best classifier is also reported.

	Entire Test Set		Reduced Test Set		
	Accuracy	Δ accuracy	Accuracy	Δ accuracy	Best parameters
Best classifier	72.38%	0	57.59%	0	
EXP1	74.11%	1.73%	64.49%	6.90%	k=12
EXP2	74.74	2.36%	67%	9.41%	k=3; k_step=9
EXP3	75.27%	2.89%	69.13	11.54%	k=10
EXP4	**75.52%**	3.14%	**70.14%**	12.55%	k=10;k_step=10, 14
EXP5	75.02%	2.64%	68.13%	10.54%	k=12, K_m=51, α=50
EXP6	74.89%	2.51%	67.63%	10.04%	k=9, K_m=51
EXP7	**75.48%**	3.10%	**70.01%**	12.42%	k=16, K_m=324
Oracle	82.97%	10.59%	100%	42.41%	

We can notice that an accurate choice of the DCS parameters allows outperforming the best classifier of the MCS. In addition, the proposed modifications of the DCS mechanism allow outperforming the basic DCS algorithm (EXP1). In particular the highest accuracies have been attained in EXP4 and EXP7, where the size (EXP4) and the shape (EXP7) of the neighbourhood have been dynamically chosen.

In order to evaluate the distribution of the results with respect to the choice of the design parameters, Tables 4 shows the maximum, medium, minimum, and standard deviation of the accuracy attained in a .number of experiments. In particular, Table 4 shows the results attained using either a fixed value of k = 12, or a varying value of k in the range [1,51]. The value of k = 12 represented the best value for the DCS based on the Euclidean metric. As it would be expected, when the value of k is fixed, EXP2 and EXP4 exhibit the lowest value of standard deviation as the only design parameter is *k_step*. On the other hand, when the value of k is not fixed the use of an adaptive metric allows attaining smaller values of standard deviation.

Table 4. Maximum, mean, minimum and standard deviation of the accuracy on the reduced test set for varying values of the parameters.

		k=12				*k*=1..51			
		Max	**Mean**	**Min**	**Std dev**	**Max**	**Mean**	**Min**	**Std dev**
Best classifier		57.59%	57.59%	57.59%	0	57.59%	57.59%	57.59%	0
EXP1		64.49%	64.49%	64.49%	0	64.49%	62.85%	60.23%	0.753
EXP2	k_step=1..14	65.12%	64.82%	64.62%	0.175	67.00%	63.14%	61.61%	0.95
EXP3		67.63%	67.63%	67.63%	0	69.13%	65.66%	62.74%	1.584
EXP4	k_step=1..14	68.38%	**68.29%**	**68.13%**	0.077	**70.14%**	66.02%	63.31%	1.91
EXP5	K_m=51..324; α=4..100	68.13%	65.54%	64.24%	1.005	68.13%	64.66%	58.85%	1.225
	K_m=51(best); α=4..100	68.13%	66.66%	65.12%	1.256	68.13%	65.07%	**64.60%**	1.292
	K_m=51..324; **α=50** (best)	68.13%	65.84%	64.24%	1.688	68.13%	64.82%	59.72%	1.331
EXP6	K_m=51..324	67.50%	65.46%	64.24%	1.41	67.63%	64.61%	58.97%	1.399
	K_m=51 (best)	67.50%	67.50%	67.50%	0	67.63%	65.14%	61.36%	1.498
EXP7	K_m=51..324;	**69.76%**	67.97%	65.75%	1.75	**70.01%**	67.02%	59.85%	1.605
	K_m=324 (best)	69.76%	69.76%	69.76%	0	**70.01%**	**67.97%**	62.48%	1.289

Table 4 also shows a number of experiment settings aimed at evaluating the influence on the performances of the different design parameters. It can be seen that the performances of the simplified DANN metric (EXP5 and EXP6) exhibit smaller values of the standard deviation than that of the original DANN metric. Thus, the DANN metric can outperform the other considered metrics thanks to a fine-tuning of the design parameters. On the other hand, the performances of the simplified DANN metric are less sensitive to a non-optimal choice of the values of the design parameters.

Table 5. Accuracy on the entire test set attained with the optimal parameters estimated from the validation set, and the optimal parameters estimated from the test set.

	Parameters estimated from the validation set	Accuracy	Parameters estimated from the test set	Accuracy
Best classifier	-	**72.38%**	-	72.38%
EXP1	*k*=19	**73.64%**	*k*=12	74.11%
EXP2	*k*=19; k_step=1-14	**73.70%**	*k*=3; k_step=9	74.74
EXP3	*k*=6	**74.99%**	*k*=10	75.27%
EXP4	*k*=6;k_step=14	**75.36%**	*k*=10;k_step=10, 14	**75.52%**
EXP5	*k*=41, K_m=175, α=50 *k*=47, K_m=175, α=20	**73.82%** **73.92%**	*k*=12, K_m=51, α=50	75.02%
EXP6	*k*=35, 36, K_m=175	**74.15%**	*k*=9, K_m=51	74.89%
EXP7	*k*=8,10, K_m=175	**74.86%; 75.05%**	*k*=16, K_m=324	**75.48%**

It can be seen that the use of an adaptive metric allows improving the accuracy of a fixed metric, thus confirming the influence of the shape and size of the neighbourhood in the computation of the CLA. In addition, the values of the standard deviation show that a sub-optimal choice of the design parameters is still capable to provide improved results. As an example, such a sub-optimal choice can be provided by esti-

mating the optimal parameter values from the validation set. Table 5 shows that this choice of the parameters allows attaining performances higher than those provided by the "standard" DCS algorithm. In addition, these results are quite close to those related to the optimal choice of the parameters estimated directly from the test set. Thus, it can be concluded that the use of an adaptive metric can improve the accuracy of the DCS based on local accuracy estimates.

References

1. L.Xu, A. Krzyzak, and C.Y. Suen, Methods for combining multiple classifiers and their applications to handwriting recognition, *IEEE Trans. on Systems, Man, and Cybernetics,* 22, pp. 418-435, 1992.
2. T.K. Ho, J.J. Hull, and S.N. Srihari, Decision combination in multiple classifiers systems,, *IEEE Trans. on* Pattern *Analysis and Machine Intelligence,* 16(1), pp. 66-75, 1994.
3. J. Kittler, M. Hatef, R.P.W. Duin and J. Matas, On Combining Classifiers, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 20, pp. 226-239,1998.
4. T.Windeatt and F.Roli (Eds.), Multiple Classifier Systems, Springer-Verlag, *Lecture Notes in Computer Science,* Vol. 2709, 2003.
5. Kuncheva L.I. Switching between selection and fusion in combining classifiers: An experiment, *IEEE Transactions on SMC,* Part B, 32 (2), 2002,146-156
6. G. Giacinto and F. Roli,, Dynamic Classifier Selection, *Proc. ofthe First Int. Workshop on Multiple Classifier Systems,* June 21-23, 2000, Cagliari, Italy, *Lecture Notes In Computer Science* 1857, Springer, Berlin, pp. 177-189.
7. Woods, K., Kegelmeyer, W.P., and Bowyer, K. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 1997, 19(4), pp. 405-410.
8. G. Giacinto and F. Roli, Adaptive Selection Of Image Classifiers, *ICIAP '97, 9th International Conference on Image Analysis and Processing,* Florence, Italy, Sept 17 - 19, 1997, pp.38-45, Lecture Notes in Computer Science 1310, Springer Verlag Ed
9. Smits, P.C., Multiple Classifier Systems for Supervised Remote Sensing Image Classification Based on Dynamic Classifier Selection, *IEEE Transactions on Geoscience and Remote Sensing,* Vol. 40, No. 4 (2002) 801-813
10. Dasarathy B. V. (Ed), Nearest neighbor(nn) norms: NN pattern classification techniques, IEEE Computer Society Press, Los Alamitos, 1991.
11. T. Hastie and R. Tibshirani. Discriminant Adaptive Nearest Neighbor Classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 18, No. 6, pp. 607-615,1996.
12. P. F. Hsieh and D. Landgrebe, Classification of high dimensional data, *Ph.D., School Elect. Comput. Eng.,* Purdue Univ., West Lafayette, IN, 1998.

Spectral Measure for Multi-class Problems

Terry Windeatt

Centre for Vision, Speech and Signal Proc., Department of Electronic Engineering
University of Surrey, Guildford, Surrey, United Kingdom GU2 7XH
t.windeatt@surrey.ac.uk

Abstract. Diversity among base classifiers is known to be a necessary condition for improved performance of a classifier ensemble. However, there is an inevitable trade-off between accuracy and diversity, which is known as the accuracy/diversity dilemma. In this paper, accuracy and diversity are incorporated into a single measure, that is based on a spectral representation and computed between pairs of patterns of different class. Although the technique is only applicable to two-class problems, it is extended here to multi-class through Output Coding, and a comparison made between various weighted decoding schemes.

1 Introduction

The idea of combining multiple classifiers is based on the observation that achieving optimal performance in combination is not necessarily consistent with obtaining the best performance for a single classifier. Although it is known that diversity among base classifiers is a necessary condition for improvement in ensemble performance, there is no general agreement about how to quantify the notion of diversity among a set of classifiers. The desirability of using negatively correlated base classifiers in an ensemble is generally recognised, and in [1] the relationship between Diversity and majority vote accuracy is characterized as classifier dependency is systematically changed. Diversity Measures can be categorised into pair-wise and non-pair-wise, but to apply pair-wise measures to finding overall diversity it is necessary to average over the classifier set. These pair-wise diversity measures are normally computed between pairs of classifiers independent of target labels. As explained in [2], the accuracy-diversity dilemma arises because when base classifiers become very accurate their diversity must decrease, so that it is expected that there will be a trade-off.

A possible way around this dilemma is to incorporate diversity and accuracy within a single measure, as suggested in this paper. The measure is based on a spectral representation that was first proposed for two-class problems in [3], and later developed in the context of Multiple Classifier Systems in [4]. It was shown for two-class problems in [5] that over-fitting of the training set could be detected by observing the spectral measure as it varies with base classifier complexity. Since realistic learning problems are in general ill-posed [6], it is known that any attempt to automate the learning task must make some assumptions. In this paper, in comparison with [5], one of the assumptions is relaxed (equ. (3) and equ. (5) in Section 2), and the only assumption required is that a suitable choice be made for the range over which base classifier complexity is varied. In this paper, the technique is applied to multi-class problems through Output Coding (Section 3), and a comparison of various weighted decoding schemes is included.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 184–193,2004.

2 Diversity/Accuracy over Pattern Pairs

Consider the following MCS framework in which a two-class supervised learning problem is solved by B parallel base classifiers whose outputs are combined by voting or summation. It is assumed that there are μ training patterns with the label given to each pattern X_m denoted by $\omega = f(X_m)$, where $m = 1,\ldots\mu$. The *mth* pattern may then be represented by the B-dimensional vector formed from the (real-valued) base classifier outputs given by

$$X_m = (\xi_{m1}, \xi_{m2}, \ldots, \xi_{mB}), \quad \xi_{mi} \in [0,1], \quad \omega \in \{0,1\}, \quad i = 1 \ldots B \tag{1}$$

If one of two classes is assigned to each of B base classifiers, the decisions may be treated as binary features. Then the *mth* pattern X_m in (1) can be represented as a vertex in the B-dimensional binary hypercube, resulting in a binary-to-binary mapping

$$X_m = (x_{m1}, x_{m2}, \ldots, x_{mB}) \quad x_{mi} \text{ and } \omega \in \{0,1\}, \; i = 1 \ldots B \tag{2}$$

In [3], a spectral representation of $f(X)$ is proposed for characterising the mapping in equ. (2). A well known property of the transforms that characterise these mappings (e.g. Rademacher-Walsh transform) is that the first order coefficients represent the correlation between $f(X)$ and x_i [7]. In [5] an estimate of first order coefficients, based on Hamming Distance D_H between pattern pairs of patterns of different class, is proposed for noisy, incompletely specified and perhaps contradictory patterns. The *mth* pattern component x_{mj} is assigned σ_{mj} $(j=1,2,\ldots B)$

$$\sigma_{mj}^{\;c} = \sum_{n=1}^{\mu} \frac{x_{mj} \oplus x_{nj}}{D_H(X_m, X_n)}, \quad f(X_n) \neq f(X_m) \tag{3}$$

where c = + if $x_{mj} = f(X_m)$, c = − if $x_{mj} = f(X_n)$, and $\oplus$ is logic XOR.

Examples of applying (3) to simple Boolean functions are given in [3]. The *jth* component x_{mj} of a pattern pair has associated σ_{mj}^{-} only if the *jth* base classifier misclassifies both patterns. Therefore we expect that a pattern with relatively large σ_{mj}^{-} is likely to come from regions where the two classes overlap. The measure σ_n' for *nth* pattern looks at the relative difference between excitatory and inhibitory contributions, normalised so that $-1 \le \sigma_n' \le 1$, defined by

$$\sigma_n' = \frac{1}{K_\sigma} \sum_{j=1}^{B} \left(\frac{\sigma_{nj}^{+}}{\sum_{m=1}^{\mu} \sigma_{mj}^{+}} - \frac{\sigma_{nj}^{-}}{\sum_{m=1}^{\mu} \sigma_{mj}^{-}} \right), K_\sigma = \sum_{j=1}^{B} \left(\frac{\sigma_{nj}^{+}}{\sum_{m=1}^{\mu} \sigma_{mj}^{+}} + \frac{\sigma_{nj}^{-}}{\sum_{m=1}^{\mu} \sigma_{mj}^{-}} \right) \tag{4}$$

In [4] plots of σ_n' are given and interpreted as a measure of class separability, indicating how well the *nth* pattern is separated from patterns of the other class by the set of B classifiers. It may be compared with the Margin (M) for the *nth* pattern, which represents the confidence of classification. Cumulative Distribution graphs can be defined for σ_n' similar to Margins, that is $g(\sigma_n')$ versus σ_n' where $g(\sigma_n')$ is the fraction of patterns with value at least σ_n'. Areas under the distribution are compared in [5], but here we plot mean over positive σ_n' and M

$$\Gamma = \frac{1}{\mu}\sum_{n=1}^{\mu}\Gamma_n, \Gamma_n > 0, \qquad \Gamma \in \{\sigma', M, \sigma'^{(H)}\} \tag{5}$$

where $\sigma_n'^{(H)}$ is a variant of σ_n' that sets D_H to 1 in (3), that is removing the assumption that the contribution between pattern pairs is inversely proportional to D_H.

Pair-wise diversity measures, such as Q statistic, correlation coefficient ρ, Double Fault F and Agreement A (1-Disagreement) measures [1], are computed over classifier pairs, using four counts, defined between *ith and jth* classifiers

$$N_{ij}^{ab} = \sum_{m=1}^{\mu}\psi_{mi}^{a} \wedge \psi_{mj}^{b} \qquad a,b \in \{0,1\},\ \psi^1 = \bar{y}, \psi^0 = y, \tag{6}$$

where $\wedge$ is logical AND, and $y_{mj} = 1$ if $x_{mj} = f(X_m)$.

For conventional diversity measures the mean is taken over B base classifiers

$$\Delta = \frac{2}{B(B-1)}\sum_{i=1}^{B-1}\sum_{j=i+1}^{B}\Delta_{ij}, \qquad \Delta \in \{Q, \rho, A, F\} \tag{7}$$

where Δ_{ij} represents the Diversity Measure between *ith and jth* classifiers.

For comparison, we also calculate mean Diversity Measures over patterns of different class (Δ') using counts as follows

$$\tilde{N}_{mn}^{ab} = \sum_{j=1}^{B}(\psi^{a}{}_{mj} \wedge \psi_{nj}^{b}), f(X_m) \neq f(X_n) \tag{8}$$

$$\text{from which } \Delta' = \frac{1}{K_\Delta}\sum_{m=1}^{\mu}\sum_{n=1}^{\mu}\Delta_{mn}, f(X_m) \neq f(X_n) \tag{9}$$

Note that equ. (4) may be re-arranged to express $\sigma_n'^{(H)}$ in the notation used in (8) since $\sum_{j=1}^{B}\sigma_{mj}^{+} = \sum_{n=1}^{\mu}\tilde{N}_{mn}^{11}$ and $\sum_{j=1}^{B}\sigma_{mj}^{-} = \sum_{n=1}^{\mu}\tilde{N}_{mn}^{00}$

3 Output Coding and Multi-class Problems

The idea of using codes is based on modelling the multi-class learning task as a communication problem in which class information is transmitted over a channel. The k x

B code word matrix Z has one row (code word) for each of *k* classes, with each column defining one of B sub-problems that use a different labelling. Assuming each element of Z is a binary variable *x,* a training pattern with target class ω_l $(l = 1... k)$ is re-labelled as class Ω_1 if $Z_{ij} = x$ and as class Ω_2 if $Z_{ij} = \bar{x}$. The two super-classes Ω_1 and Ω_2 represent, for each column, a different decomposition of the original problem. In the test phase the *pth* pattern is assigned to the class ω_i that is represented by the closest code word, where distance of the *pth* pattern to the *ith* code word is defined as

$$D_{pi} = \sum_{j=1}^{B} \alpha_{jl} \left| Z_{ij} - \upsilon_{pj} \right| \qquad l = 1,...k \tag{10}$$

where α_{jl} allows for *lth* class and jth classifier to be assigned a different weight. Hamming decoding is denoted in equ. (10) by $\{\alpha_{jl}=1,\ \upsilon_{pj}=x_{pj}$ *equ (2)}* and L^1 norm decoding by $\{\alpha_{jl}=1,\ \upsilon_{pj}=\xi_{pj}$ *equ. (1)}*. In addition to the Bayes error, errors due to individual classifiers and due to the combining strategy can be distinguished. This can be further broken down into errors due to sub-optimal decomposition and errors due to the distance-based decision rule. If it is assumed that each classifier provides exactly the probability of respective super-class membership, with posterior probability of *lth* class represented by q_{pl} $(l = 1 ... k)$, from equation (10), assuming $\alpha_{jl}=1$, it is shown in [8] that

$$D_{pi} = \sum_{j=1}^{B} \left| \left(\sum_{l=1}^{k} q_{pl} Z_{lj} \right) - Z_{ij} \right| = (1 - q_{pi}) \sum_{j=1}^{B} \left| Z_{lj} - Z_{ij} \right| \tag{11}$$

Equation (11) tells us that D_{pi} is the product of $(1-q_{pi})$ and Hamming Distance between code words, so that when all pairs of code words are equidistant, minimising D_{pi} implies maximising posterior probability, which is equivalent to Bayes rule. Therefore any variation in Hamming distance between pairs of code words will reduce the effectiveness of the combining strategy. Further criteria governing choice of codes are discussed in [8], but finding optimal code matrices satisfying the criteria is a complex problem, and longer random codes have frequently been employed with almost as good performance. Many types of decoding are possible but theoretical and experimental evidence indicates that, providing a problem-independent code is long enough, performance is not much affected. There is recent interest in adaptive, problem-dependent and non-binary codes [9], but in this paper a random code matrix with near equal split of classes (approximately equal number of 1's in each column) is chosen [10].

When base classifiers are sub-optimal and vary in accuracy (unbalanced [11]) it may be that the decoding strategy can assist in reducing error and in Section 5 various weighted decoding schemes are compared. All weights proposed in this study are fixed in the sense that none change as a function of the particular pattern being classified, which is sometimes referred to as implicit data-dependence or constant weighting. It is generally recognized that a weighed combination may in principle be superior, but it is not easy to estimate the weights. Fixed weighting coefficients $(\alpha_{jl}$ $j=1...B,\ l = 1 ... k)$, estimated from training data are proposed as follows

$$\alpha_{jl}^{\sigma'} = \frac{1}{K_{\sigma'}}\left(\sum_{m=1}^{\mu}\sigma_{mj}^{+} - \sum_{m=1}^{\mu}\sigma_{mj}^{-}\right) \tag{12}$$

$$\alpha_{jl}^{\sigma'(H)} = \frac{1}{K_{\alpha^{\sigma}}}\sum_{m=1}^{\mu}\sum_{n=1}^{\mu}\{(y_{mj} \wedge y_{nj}) - (\bar{y}_{mj} \wedge \bar{y}_{nj})\}\,,\ f(X_m) \neq f(X_n) \tag{13}$$

$$\alpha_{j}^{A'} = 1 - \frac{1}{K_{\alpha^{A}}}\sum_{m=1}^{\mu}\sum_{n=1}^{\mu}\{(y_{mj} \wedge \bar{y}_{nj}) - (\bar{y}_{mj} \wedge y_{nj})\}\,,\ f(X_m) \neq f(X_n) \tag{14}$$

where y_{mj} is defined in equ. (6)

These weighting functions are intended to give more weight to classifiers that separate the two classes well, in contrast to the following weighting, which is based on the Agreement measure

$$\alpha_{j}^{A} = \frac{1}{K_A}\sum_{i=1}^{B} A_{ij}, i \neq j \tag{15}$$

Normalization constants K in equ. (12) to (15) are chosen so that weights sum to 1, and negative weights in (12) and (13) are set to zero. Note that decodings in (12) and (13) have different weights for each class, which come by computing $\sigma'^{(H)}$ and σ' from the k binary-to-binary mappings defined by one-vs-rest coding with respect to base classifier output decisions.

4 Experimental Evidence

Multi-class benchmark problems have been selected from [12], and the experiments use random 50/50 or 20/80 training/testing splits. The datasets, including number of patterns, are Segment (2310), Iris (150), Ecoli (336), Yeast (1484), Vehicle (846), Vowel (990). All experiments are performed with one hundred single hidden-layer MLP base classifiers, except where the number of classifiers is varied. The algorithm is the Levenberg-Marquardt training algorithm with default parameters, and the number of training epochs is systematically varied for different runs of the MCS. Random perturbation of the MLP base classifiers is caused by different starting weights on each run, and each point on the graphs is mean over ten runs.

For the datasets tested in this paper, none over-fitted for the range of base classifier complexity values considered. To encourage over-fitting the experiments were carried out with varying classification noise, in which a percentage of patterns of each class were selected at random (without replacement), and each target label changed to a class chosen at random (from patterns of the remaining classes). Figure 1 gives test and train error rates for Segment 50/50 + 20% classification noise with [2,4,8,16] hidden nodes, and Figure 2 shows various measures defined in equ. (5) (7) (9). Figure 3 shows test error, margin M and $\sigma'^{(H)}$ (equ. (5)) for Iris 50/50 at 8 nodes with [0,10,20,30,40] % classification noise. By comparing Figure 1 and Figure 2 at 16 nodes, and observing Figure 3, it appears that $\sigma'^{(H)}$ may be a good predictor of test error and detect over-fitting. Note also in Figure 3, for more than 27 epochs, that over-fitting reaches the point where $\sigma'^{(H)}$ starts to increase. This is not unexpected since as number of nodes and epochs becomes very large, and all patterns become correctly classified $\sigma'^{(H)} \rightarrow 1$.

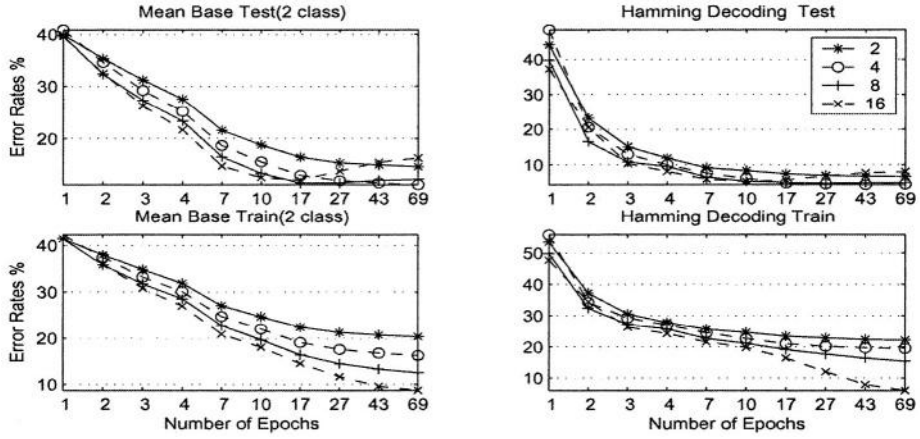


Fig. 1. Error rates for Segment 50/50 + 20% noise with [2,4,8,16] nodes.

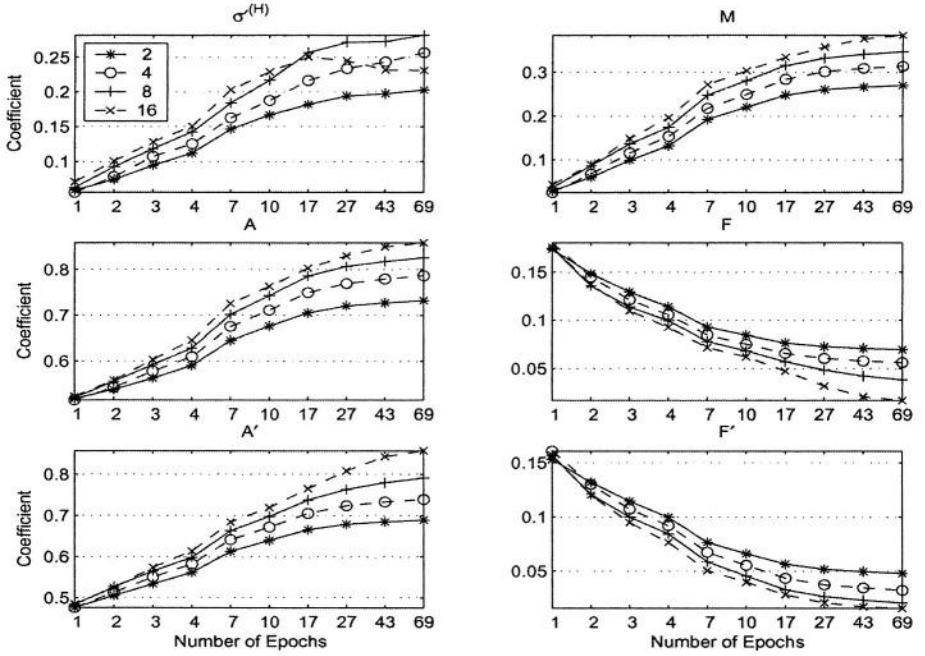


Fig. 2. Measures for Segment 50/50 + 20% noise for [2,4,8,16] nodes.

Table 1 shows mean correlation coefficient, over all 50/50 and 20/80 datasets+20% noise, of test error with respect to [1-69] epochs over [2,4,8,16] nodes. The correlation is shown for base classifier, Hamming and L1 norm decoding (defined equ. (10)) and demonstrates that $\sigma'^{(H)}$ is well correlated with base classifier test error. However, most measures appear well correlated, and the experiments were repeated for the restricted range of epochs 2-20 at 2 epoch intervals, with varying noise. Table 2 shows correlation coefficient for 50/50 datasets of base classifier test error at 8 nodes

averaged over [0,10,20,30,40] %noise. Table 3 has the same data as Table 2 but for 20/80 datasets, and Table 4 shows mean correlation coefficient, over all 50/50 and 20/80 datasets, of test error with respect to [2-20] epochs at 8 nodes averaged over [0,10,20,30,40] %noise. Table 2 to Table 4 indicates that $\sigma'^{(H)}$ is better correlated with base classifier test error than other measures, particularly for 20/80 datasets. Also $\sigma'^{(H)}$ was the only measure that was significantly correlated with base classifier test error for all datasets, 50/50 and 20/80 (95% level in comparison with random chance).

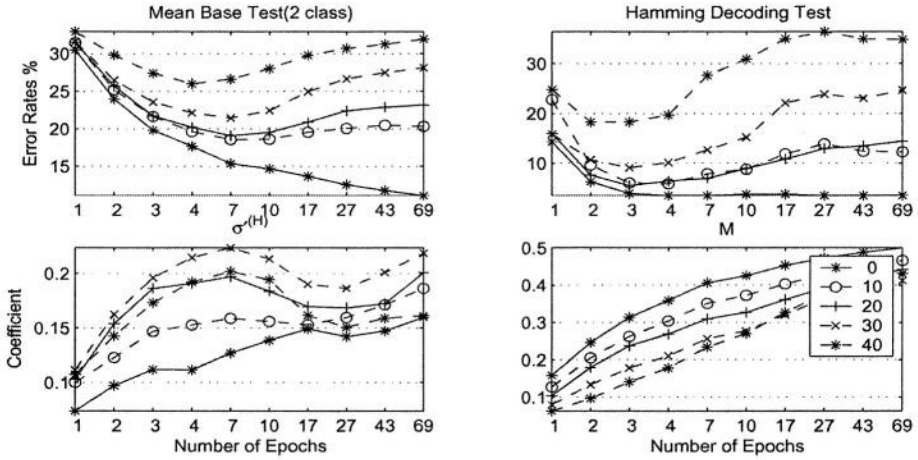


Fig. 3. Test error, M and $\sigma'^{(H)}$ for Iris 50/50 with [0,10,20,30,40] % noise.

Figure 4 shows the mean difference, over all 20/80 datasets, between Hamming and various other decoding schemes as number of classifiers (number of columns of code matrix Z) is increased. L^l, $\alpha^{\sigma'}$, $\alpha^{\sigma'(H)}$, $\alpha^{A'}$, α^{A} are defined in equ. (10), (12)-(15) and α^{ada} is chosen according to the Adaboost logarithmic function of training error. Both $\alpha^{\sigma'}$ and $\alpha^{\sigma'(H)}$ give dramatic improvement for 1-4 epochs, which is almost matched by α^{ada} but not by $\alpha^{A'}$ and α^{A}. As base classifiers become more unbalanced at small number of epochs, weighted decoding appears to be more effective. L^l is better than Hamming decoding although there is little difference when number of classifiers increases above 40.

Other datasets tested were Dermatology, Waveform and Soybean, none of which showed a good correlation between $\sigma'^{(H)}$ and test error. These three datasets have discrete features, in contrast to the other six datasets which all have continuous features. It is believed that the effect of discrete features on the probability distribution in the overlap region may account for this result, but further work is required to corroborate the observation.

5 Conclusion

The proposed measure, calculated over the training set, is capable of detecting overfitting of base classifier test error, and can be used to design a weighted vote combiner. Although the measure is only applicable to two-class problems, experimental

results in this paper demonstrate that it may also be applied to the artificial two-class decompositions induced by Output Coding.

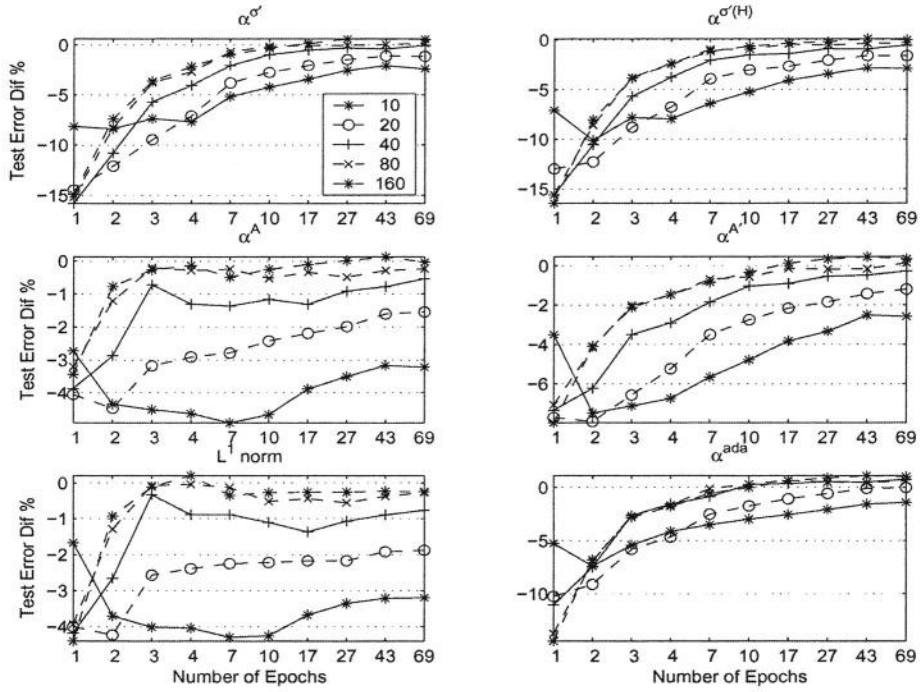


Fig. 4. Test Error minus Hamming Decoding Error over 20/80 datasets for various decoding schemes with [10 20 40 80 160] classifiers.

Table 1. Mean correlation coefficient (x100), over 50/50 and 20/80 datasets +20% noise, of test error with respect to [1-69] epochs over [2,4,8,16] nodes.

	BASE CLASSIFIER		HAMMING DECODING		L^1 NORM DECODING	
MEAS	50/50	20/80	50/50	20/80	50/50	20/80
-Q	75	25	68	33	71	36
-ρ	88	58	79	62	82	64
-A	82	76	65	56	64	51
F	87	84	72	67	71	63
-Q′	75	68	57	48	57	44
-ρ′	73	67	55	46	55	43
-A′	80	75	64	56	63	52
F′	90	86	75	70	74	66
-σ′	82	71	68	61	68	57
$-\sigma'^{(H)}$	91	84	76	68	76	63
-M	73	69	57	49	56	45

Table 2. Correlation coefficient (x100) for 50/50 datasets, of base classifier test error with respect to [2-20] epochs at 8 nodes averaged over [0,10,20,30,40] % noise.

Dataset	- Q	- A	F	- Q′	- A′	F ′	- σ′	-σ′(H)	- M
Segment	97	96	97	97	95	99	98	97	93
Iris	82	74	73	50	71	81	97	88	64
Ecoli	75	75	78	67	69	83	85	86	62
Yeast	95	90	92	86	87	93	98	96	79
Vehicle	67	76	86	75	77	88	89	93	67
Vowel	72	90	97	87	90	98	22	98	83

Table 3. Correlation coefficient (x100), for 20/80 datasets, of base classifier test error with respect to [2-20] epochs at 8 nodes averaged over [0,10,20,30,40] %noise.

Dataset	- Q	- A	F	- Q′	- A′	F ′	- σ′	-σ′(H)	- M
Segment	68	91	90	89	88	94	96	98	84
Iris	39	31	33	12	30	39	60	70	22
Ecoli	26	13	19	2	8	29	58	75	-3
Yeast	86	63	65	51	57	70	95	93	44
Vehicle	-1	52	72	45	54	74	56	78	42
Vowel	-32	80	94	70	81	95	40	91	74

Table 4. Mean correlation coefficient (x100), over all 50/50 and 20/80 datasets, of test error with respect to [2-20] epochs at 8 nodes averaged over [0,10,20,30,40] % noise.

·	BASE CLASSIFIER		HAMMING DECODING		L^1 NORM DECODING	
MEAS	50/50	20/80	50/50	20/80	50/50	20/80
-Q	75	31	68	41	71	42
-ρ	88	56	79	59	82	61
-A	82	55	65	31	64	33
F	87	62	72	39	71	41
-Q′	75	45	57	23	57	24
-ρ′	73	42	55	21	55	22
-A′	80	53	64	29	63	31
F′	90	67	75	44	74	45
-σ′	82	68	68	65	68	66
- σ′(H)	91	84	76	75	76	75
-M	73	44	57	21	56	23

References

1. L. I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles, Machine Learning 51, 2003,181-207.
2. L. I. Kuncheva, M. Skurichina, R.P.W. Duin. An experimental study on diversity for bagging and boosting with linear classifiers, Information Fusion, 3 (2), 2002, 245-258.

3. T. Windeatt, R. Tebbs, Spectral technique for hidden layer neural network training, Pattern Recognition Letters, Vol.18(8), 1997, 723-731.
4. T. Windeatt, Recursive Partitioning for combining multiple classifiers, Neural Processing Letters 13(3), June 2001, 221-236.
5. T. Windeatt, Vote Counting Measures for Ensemble Classifiers, Pattern Recognition 36(12), 2003, 2743-2756.
6. A. N. Tikhonov, V. A. Arsenin, Solutions of ill-posed problems, Winston & Sons, Washington, 1977.
7. S. L. Hurst, D. M. Miller, J. Muzio, *Spectral Techniques in Digital Logic,* Academic Press, 1985.
8. T. Windeatt T. R. Ghaderi., Coding and Decoding Strategies for multiclass learning problems, Information Fusion, 4(1), 2003, 11-21.
9. K. Crammer, Y. Singer, Improved Output Coding for Classification Using Continuous Relaxation, in T.G.Dietterich, S. Becker, and Z. Ghahramani (eds.) *Advances in Neural Information Processing Systems 14.* MIT Press, Mass., 2002.
10. R. E. Schapire, Using output codes to boost multiclass learning problems, 14th Int. Conf. of Machine Learning, Morgan Kaufman, 1997, 313--321.
11. G. Fumera, F. Roli, Linear Combiners for Classifier Fusion: Some Theoretical and Experimental Results, Proc. of 4th Int. Workshop on Multiple Classifier Systems, Eds: T.Windeatt, F. Roli, Guildford, UK, Lecture Notes in Comp. Science, Springer Verlag, June 2003, 74-83.
12. C.J. Merz, P. M. Murphy, UCI repository of machine learning databases, 1998 http://www.ics.uci.edu/~mlearn/MLRepository.html

The Relationship between Classifier Factorisation and Performance in Stochastic Vector Quantisation

David Windridge, Robin Patenall, and Josef Kittler

Centre for Vision, Speech and Signal Processing
Dept. of Electronic & Electrical Engineering, University of Surrey
Guildford, GU2 5XH Surrey, United Kingdom
d.windridge@eim.surrey.ac.uk
Tel: +44 1483 876043

Abstract. We seek to address the issue of multiple classifier formation within Luttrell's stochastic vector quantisation (SVQ) methodology. In particular, since (single layer) SVQs minimise a Euclidean distance cost function they tend to act as very faithful encoders of the input: however, for sparse data, or data with a large noise component, merely faithful encoding can give rise to a classifier with poor generalising abilities. We therefore seek to asses how the SVQs' ability to spontaneously factorise into independent classifiers relates to overall classification performance. In doing so, we shall propose a statistic to directly measure the aggregate 'factoriality' of code vector posterior probability distributions, which, we anticipate, will form the basis of a robust strategy for determining the capabilities of stochastic vector quantisers to act as unified classification/classifier-combination schemes.

1 Introduction

1.1 Stochastic Vector Quantisation

Stochastic vector quantisation (SVQ) [eg 1-3] can be considered to serve to bridge the conceptual gap between topographic feature mapping [5] and standard vector-quantisation [6]. It does so by utilising a 'folded' Markov chain topology to statistically relate input and output vectors conceived as occupying the same vector space via the minimisation of a positional reconstruction error measure. That is, for the input vector x, we seek to minimise the aggregate Euclidean distance:

$$D \equiv \int dx\, Pr(x) \sum_{y_1=1}^{M} \sum_{y_1=2}^{M} \ldots \sum_{y_n=1}^{M} Pr(y|x) \int dx' Pr(x'|y)||x - x'||^2 \qquad (1)$$

where x' is the output vector, and $y = (y_1, y_2, \ldots y_n) : 1 \leq y \leq M$ the code-index vector encoding of x.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 194–203, 2004.

Although soluble non-parametrically for certain theoretical cases, in practise, a number of constraints are required to achieve this minimisation [3], the most significant of which for the present purposes being the limitation of $Pr(y|x)$ to a sigmoid form:

$$Pr(y|x) = \frac{Q(y|x)}{\sum_{y'=1}^{M} Q(y'|x)} \tag{2}$$

where

$$Q(y|x) \equiv \frac{1}{1 + \exp(-\omega(y).x - b(y))} \tag{3}$$

Here b represents a bias offset in the input space and ω a weight vector with behaviour characteristics familiar from the study of artificial neural networks (although the normalisation factor can considerably modify the intrinsic sigmoid morphology). It is possible, within this framework, to concatenate chains of SVQs together to form a multistage network with the previous code-vector space becoming the input space of the subsequent SVQ, in which case the objective function defaults to a weighted sum of the reconstruction errors of each stage in the chain. It is also the case that the topographic aspects inherent in equation 1 can be *explicitly* pre-specified through the incorporation of a 'leakage current' term: $\Sigma_{y'} P(y|y')p(y'|x)$ in equation 2.

The method by which we shall attempt to attain the global minimum of the quantity D for both single and multistage SVQs proceeds, in the current paper, via a minor adaptation of Luttrell's original method [3], whereby a pseudo-annealing regime is employed within which step-size updates between gradient-descent iterations are adjusted logarithmically until convergence is attained. In this way we mimic the ideal logarithmic temperature reduction of simulated annealing, albeit without a thermalisation step [cf eg 7]. When we later come to apply additional constraints on the convergence, it will be assumed that the underlying convergence mechanism is still that of a logarithmic gradient descent minimisation of D.

An important consequence arises from this uniquely probabilistic form of vector quantisation: the act of maximising the error resilience of the necessarily bandwidth-limited transmission of stochastically sampled pattern-vector code information over the folded Markov chain structure gives rise to a completely natural mechanism for imposing an appropriately-dimensioned topology on the training vectors. In particular, it becomes possible to perform a factorial decomposition of the input when strong factor independence is indicated by the data structure (and the SVQ 'bandwidth' is not significantly in excess of that required to represent the factorised code-vector distribution). That is, the SVQ methodology is able to serve as a factor analyser as required, *without* any a priori imposition at the conceptual level, the topological constraints on the SVQ arising solely as a consequences of the data morphology and the requirement of representing it in the most efficient and noise-robust manner possible (efficient representation, in effect, dictating factorial-decomposition, and the error-robustness constraint imposing a proximity-based topology on the code-vector priors). In the absence of well defined independent factors in the data (or with an excess parametric

freedom in the SVQ), the proximetric aspect of the code-transmission error estimation robustness tends to dominate, and the SVQ code-vectors act to faithfully encode only cluster proximities within the training data.

1.2 Objective of the Current Investigation

Beyond being an indicator of SVQ representative efficiency, the concept of factoriality is, we expect, also significant as an indicator of potential classification performance. Thus taking a factorial perspective on the mechanics of SVQ convergence significantly differs from (but is in no way exclusive of) the characterisation of the SVQ state solely via the objective function, which describes the closeness of fit to the training vectors, but does not, however, *in itself* give an indication of the extent to which the SVQ has captured all of the morphological aspects of the classification problem under consideration.

We should, in the following paper, therefore like to test the intuition that the maximally factorial behaviour in relation to the SVQ information bandwidth parameters (sampling frequency and code vector cardinality) coincides, in general, with the peak of classification performance on typical test data-sets, and consequently with the actual underlying pattern-vector probability density distribution in so far as it is representable by the SVQ.

Our motivation for this stems from a general observation (experimentally formalised in section 3), that when the number of code indices is increased linearly, SVQs undergo a behavioural transition as the mechanism firstly characterises individual localities of the pattern space without co-ordinatising any particular sub-manifold, before making a transition to the factorial encoding regime at a point at which there is sufficient parametric freedom available to the SVQ for it to efficiently describe the predominating sub-manifold of the training data. As further code indices are made available to the SVQ, the method once again tends toward joint encoding, this behaviour apparently correlating with a *reduction* in the generalising ability of the classifier as the SVQ starts to characterise localised peculiarities of the training data that are not shared by the test set. The criterion of 'maximum factoriality' might therefore be supposed to serve to determine the appropriate amount of informational bandwidth required to efficiently characterise the underlying class PDFs from which the pattern data are drawn, beyond which the effect of over-classification begins to predominate, as too much parametric freedom is allocated in relation to the training set, and below which the classification suffers from insufficient data description bandwidth.

It is therefore this hypothesis, namely that maximal factoriality with respect to the various information bandwidth parameters corresponds to the most efficient descriptor of the pattern space, corresponding in turn to the most effectively generalising classifier, that we shall seek to test in the following paper.

1.3 Paper Structure

In addressing, then, the issues surrounding SVQs as general classifiers, our first specific objective, after having derived an appropriate measure of SVQ factoriality in section 2, will be to provide experimental evidence of the assertion that

factoriality correlates with classifier generalising ability, firstly, by demonstrating the correlation between SVQ 'information bandwidth' and factorisation of the code vectors with respect to simulated data, and secondly, via a demonstration that the maximum of SVQ factoriality corresponds to the optimal classification performance (constituting sections 3 and 4, respectively). Following this, we shall turn in section 5 to a brief consideration how maximised factoriality might be accomplished within the existing framework of Euclidean error objective function minimisation, findings suggesting that an appropriate upper-layer weighting will encourage factorial encoding of the initial layer as well as effectively acting as a combination scheme for the factorised classifiers.

2 Aggregate Weight Collinearity as a Factoriality Measure

The most appropriate statistic of classifier performance has thus been hypothesised to be one that gives a measure of the degree to which SVQs factorise in relation to a particular training set. Any such realisation of the proposed statistic would, necessarily, have to be robustly independent of both the bias factors and the feature dimensionality, as well as being simple to compute. Of the strategies to achieve this that most readily suggest themselves, two broad types may be distinguished within the terms of our analysis: the 'morphological' (ie concerning the input-space of the folded Markov chain) and the 'topological' (concerned with the interconnections of the code indices at the higher levels of the sequence of folded Markov chains). The former thus seeks to determine factoriality by measuring certain aggregate parameters of the code index probability *distributions* in the input space, whilst the latter determines the pattern of interconnectivity of the code indices themselves. However, topological measures, while being arguably the truest measure of 'factoriality' (given that the term is not absolutely defined), have the disadvantage of not directly reflecting underlying metrical disposition, as well as, in some cases, requiring a global optimisation algorithm to enumerate (for instance, a block-diagonal factorisation). Morphological measures, on the other hand, are capable of *implicitly* capturing the topological aspect of factoriality, while more directly capturing its metrical aspects. Furthermore, morphology measures can naturally encompass the form of generalisation that is required to occur in response to additive noise, in particular, the elongations of the code-vector kernels along the noisy dimensions (see [eg 3] for an illustration of this behaviour).

Favouring, then, the morphological approach to factoriality quantification, the most straightforward strategy is to take a density spectrum of the orientations of the individual weights, which exhibits a very clear bimodality in relation to the two distinct behavioural classes of factorial and joint encoding open to the SVQ. We define this quantity formally as follows:

2.1 Derivation of Factoriality Measure

We denote a specific weight-vector by ω_{ij}, where the first index, i, denotes a particular ordinate of the input space and the second index, j, denotes the code

index with which the weight is associated. The hyper-length-normalised weight vector is therefore given as:

$$\overline{\omega_{ij}} = \frac{\omega_{ij}}{\left[\sum_{i=1}^{i=d}(\omega_{ij})^2\right]^{\frac{1}{2}}} \tag{4}$$

Only this latter quantity shall be considered throughout the following; a consequence of our being primarily interested in the aggregate weight *orientations.*

The weight orientations are thus distributed around a $(d-1)$-dimensional angular hyper-space, orientation specification requiring one fewer free parameter than the specification of hyper-position. We wish to derive an appropriate model for this distribution in terms of the d weight ordinates with which we are presented by the SVQ. The first task is therefore to determine a density-bin size in the ordinate space appropriate to angularly distributed vectors.

We have from [4] that points uniformly distributed on the hypersphere S^{N-1} normalised to unity, embedded within R^N and then projected into R^M $(M < N)$ have radial probability distributions $P_{N,M}(r)$ given by:

$$P_{N,M}(r) = c_{N,M} r^{M-1}(1-r^2)^{(N-M-2)/2} \tag{5}$$

where:

$$c_{N,M} = \frac{2\Gamma(\frac{N}{2})}{\Gamma(\frac{M}{2})\Gamma(\frac{N-M}{2})} \tag{6}$$

So, for example, for $N = 3, M = 2$ we obtain:

$$P_{3,2}(r) = \frac{r}{2(1-r^2)^{\frac{1}{2}}} \tag{7}$$

Consequently, the appropriate bin size for our unidimensional $(M = 1)$ sample of the d-dimensionally-embedded hypersphere is given, for small Δi, by:

$$\begin{aligned} \Delta i(r) &= \Delta x \frac{1}{P_{d,1}(r)} \\ &= \Delta x \frac{(1-r^2)^{(3-d)/2}\pi^{\frac{1}{2}}\Gamma(\frac{d-1}{2})}{2\Gamma(\frac{d}{2})} \end{aligned} \tag{8}$$

with Δx some constant such that $\Delta x < 1$ (smaller Δx's giving greater accuracy at the expense of a lower linear sampling-rate). Thus, for uniformly distributed angular vectors, the $1-D$ projection with bin-size Δx gives rise to a uniform distribution.

Beyond this, we also require a coordinate transformation T between the density-normalised unidimensional space, x, and the radial-projection space, r. This is straight-forwardly given by the integral of the above as $\Delta x \to 0$:

$$T(x) = \frac{2\pi^{(d-1)/2} r\left[{}_2F_1(\frac{1}{2}, \frac{1-D}{2}; \frac{3}{2}; r^2)\right]}{\Gamma(\frac{1-D}{2})} \tag{9}$$

with ${}_2F_1$ the inductively-derived generalised hypergeometric function.

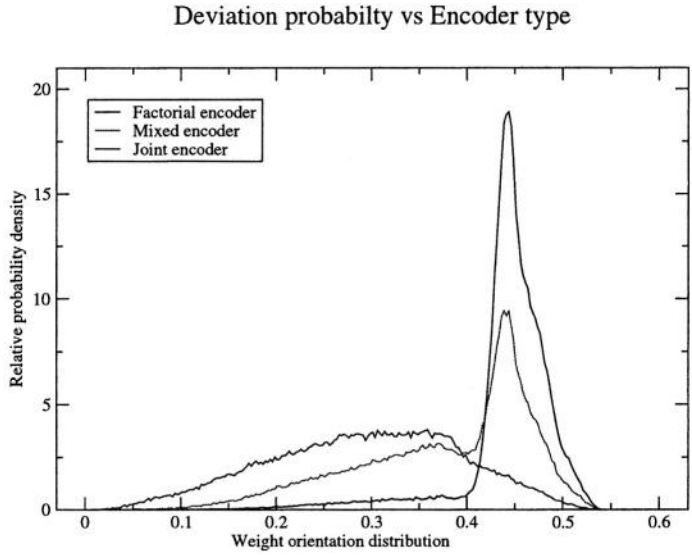


Fig. 1.

The regularised density spectrum for weight-orientations with respect to the feature-ordinate X_i sampled at intervals Δx is thus:

$$D_i(X_i) = \sum_{j=1}^{M} \int_{x=X_i}^{x=X_i+\Delta x} \delta(x - T^{-1}(\overline{\omega_{ij}}))dx \tag{10}$$

$$= \sum_{j=1}^{M} \int_{r=T^{-1}(X_i)}^{r=T^{-1}(X_i)+\Delta x/P_{d,1}(r)} \delta(r - \overline{\omega_{ij}})dr \tag{11}$$

We illustrate the bimodal behaviour of this distribution with respect to joint and factorial encoding in figure 1 on our real-world data-set, selected to be broadly representative of the sorts of data-set commonly brought before the pattern recognition community for automated classification. It consists in a set of expertly-segmented geological survey images, with features determined via a battery of cell-based processes for texture characterisation, chosen without regard to the particular nature of the classification problem. Hence the data exhibits an approximate manifold structure of considerably smaller intrinsic dimensionality than the total feature set cardinality. Of the original 26 features, a subset of three are consequently chosen to exemplify this manifold structure for the testing of the SVQ methodology. In particular, the data is sufficiently sparse for factorial and joint encoding to produce significantly different code vector probability distributions. In practical terms, to utilise the regularised collinearity distribution $D_i(X_i)$ as a factoriality measure, it is necessary to contract the information to a single value by taking the mean quantity:

$$\sum_i \frac{M}{d}\left[\frac{\sum_{x=0}^{|X_i|/\Delta x}\left\{D(x_i) : D(x_i) > \frac{M}{|X_i|/\Delta x}\right\}}{\sum_{x=0}^{|X_i|/\Delta x}\left\{1 : D(x_i) > \frac{M}{|X_i|/\Delta x}\right\}}\right]^{-1} \tag{12}$$

which would correspond to manifold dimensionality in the case where equal numbers of code vectors are allocated per ordinate, and the manifold achieves optimal factorisation.

In general, maximal factoriality requires that we minimise this quantity *with respect to a fixed Euclidean error* (Euclidean error solutions being degenerate with respect to factoriality).

3 Factoriality as a Function of SVQ Bandwidth Parameterisation

Having obtained an appropriate factoriality statistic, it is necessary to establish its relation to the various bandwidth parameters of the SVQ (which, when concatenated, would correspond loosely to the 'information content' of the resulting pattern space description). The two critical measures of this SVQ 'bandwidth' are the number of code indices, M, and the number of stochastic samples, n, employed in the derivation of $Pr(y|x)$, giving, respectively, the number of free morphological descriptors of the space, and an indication of their 'resolution'. A plot of these parameters against the factoriality measure for a canonical 2-toroid manifold results in a graph of the type shown in figure 2a, which clearly demonstrates a maximum at 8 code indices that is free of dependency on the number of samples, provided that $n > 10$ (the method generally factorialising maximally at a lower figure when the codes are badly under-sampled). A graph of the full spectrum of weight orientation densities is shown in figure 2b, plotted against the code index parameter that we have thus established to be the predominating information constraint (on the proviso that a sufficient number of samples have been allocated). This result, critically, correlates with the analytic findings of [1] for the same data set, demonstrating that factoriality is the characteristic quantity defining the effectiveness with which the SVQ code disposition is successful in characterising the intrinsic qualities of the data. Identical results (not reported in this paper) are obtained for comparable empirical data.

4 Investigation of Relationship between Factoriality and Classification Performance

It is now possible to address the central issue of classification performance, and instigate an investigation via the two-stage supervised procedure. We shall therefore test our assertion of the relation of the factoriality statistic to classification performance with reference to the simulated 2-toroid data-set. In particular, we shall specify a two-class case consisting in a pair of displaced 2-toroids separated by one radial unit across their major axis, performing cross-validation via an equivalently-sized set of test and training vectors.

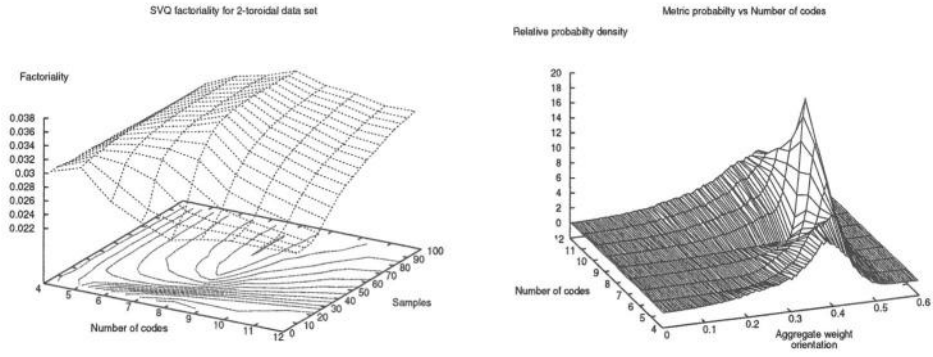


Fig. 2.

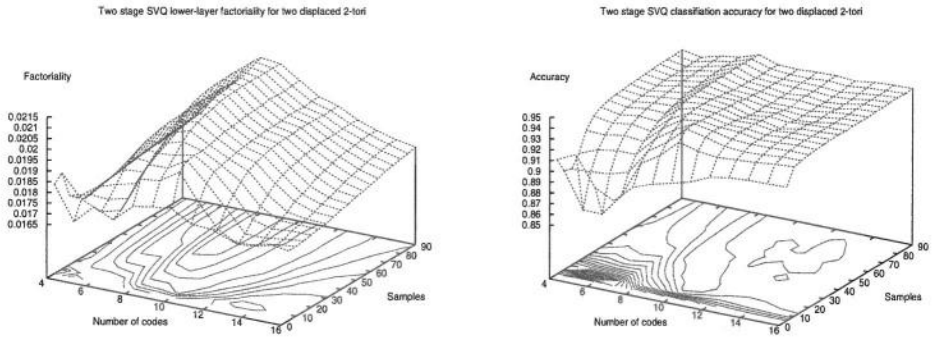


Fig. 3.

Results for this approach are set out in figures 3a and 3b, which collectively demonstrate an extremely strong (although non-linear) correspondence between the measures of classification performance and (lower-layer) factoriality, as measured against the two information-bandwidth parameters.

5 Non-parametric Factoriality: The Prospects for Employing Multi-layer SVQs as Generalised Bayesian Classifiers/Classifier-Combiners

We have thus far modified SVQ behaviour by adjusting the bandwidth parameter to match that of the underlying pattern-data's manifold information content. We should ultimately like to address the problem of maximising factoriality from the perspective of fixed parameter SVQs. How, then, might factoriality be induced without *a priori* tuning? The most straight-forward solution is to introduce an additional layer: we have already seen how a two-stage system can act as a supervised classification system, with the upper layer determining class allocations; however, it is also the case that the presence of an upper layer will act to encourage factoriality (which is to say, spontaneous classifier subdivision)

by virtue of decreasing the overall hyper-volume of the code-vector probability distribution. Although beyond the scope of this paper to elucidate fully, we should consequently like to set the stage for addressing the question of how the connection strengths both within and between layers in a multi-layer SVQ system could be made to respond to factorial analysis in a manner appropriate to maximising overall classification performance. That is to say, we would specifically like to assess how the second (or higher) SVQ layers could act as both classifier diversifiers (via the connection strength parameter), as well as classifier *combiners* (via the specific code vector topologies). In this way, it should be possible to re-envisage SVQs as a method for unifying feature-selection, classifier design and classifier combination within a single framework, and hence, by using our factoriality measure as a diagnostic tool, gaining an understanding of the appropriate balance between these component subsystems free of the *a priori* structural constraints inherant in other multi-classifier systems.

6 Conclusion

We have set out to give an indication of how the classification abilities of the stochastic vector quantisation methodology might be enhanced in terms of its capacity to factorise. Doing so involved, firstly, the construction of a robust and economic aggregate measurement of factoriality in terms of the net weight collinearity, and secondly, an investigation across the range of SVQ bandwidth parameters of the relationship between the degree of factoriality and the Bayesian classification performance with respect to simulated data sets. We thereby confirmed both the presence of an intrinsic 'manifold information content' within the data (via the *singular* peak of the factoriality statistic) with respect to the single-layer SVQ bandwidth parameters, and also the correspondence of this peak with the maximal generalising ability of the classifier.

We then went on to briefly consider a bandwidth-independent multilayer SVQ implementation, wherein the level of factoriality is achieved with respect to the connection strength parameter of a second (or higher) SVQ stage. In doing so, we were able to begin to re-envisage SVQs as a mechanism capable of *spontaneously* evolving towards a combined classifier framework, it being argued that SVQs thus represent a uniquely natural test-bed for studying expert fusion and diversity issues. In particular, now that a suitable diagnostic measure has been derived with the potential to determine the interaction between classifier diversification and combination *in situ,* we are in a position to give substance to the notion that SVQs are sufficiently parametrically-free classifiers as to effectively be considered constrained only by training data morphology.

Acknowledgment

The authors would like to gratefully acknowledge the support and assistance of Dr Stephen P. Luttrell in the compilation of this paper. The research itself was carried out at the University of Surrey, UK, supported by, and within the framework of, QinetiQ sub-contract number CU016-013391.

References

1. Luttrell S P, Self-organised modular neural networks for encoding data, in Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, 1997, pp. 235-263, Springer-Verlag, ed. Sharkey A J C.
2. Luttrell S P, 1997, A theory of self-organising neural networks, in Mathematics of Neural Networks: Models, Algorithms and Applications (Operations Research/Computer Science Interfaces, Kluwer, ed. Ellacott S W, Mason J C and Anderson I J), pp. 240-244
3. Luttrell S P, A user's guide to stochastic encoder/decoders. DERA Technical Report, DERA/S&P/SPI/TR990290 (1999)
4. K. Zyczkowski and H-J Sommers, J. Phys. A: Math. Gen. **33** (2000) 2045-2057
5. Kohonen T, "Self organisation and associative memory", 1994, Springer-Verlag.
6. Linde T, Buzo A and Gray R M, An algorithm for vector quantiser design, IEEE Trans. COM, 1980, **28**, pp 84-95
7. K. Binder (Editor), A. Baumgartner, The Monte Carlo Method in Condensed Matter Physics (Topics in Applied Physics, Vol 71), 1996, Springer-Verlag.

A Method for Designing Cost-Sensitive ECOC

Claudio Marrocco and Francesco Tortorella

Dipartimento di Automazione, Elettromagnetismo
Ingegneria dell'Informazione e Matematica Industriale
Università degli Studi di Cassino
03043 Cassino (FR), Italy
{marrocco,tortorella}@unicas.it

Abstract. Error Correcting Output Coding is a well established technique to decompose a multi-class classification problem into a set of two-class problems. However, a point not yet considered in the research is how to apply this method to a cost-sensitive classification that represents a significant aspect in many real problems. In this paper we propose a novel method for building cost-sensitive ECOC multi-class classifiers. Starting from the cost matrix for the multi-class problem and from the code matrix employed, a cost matrix is extracted for each of the binary subproblems induced by the coding matrix. As a consequence, it is possible to tune the single two-class classifier according to the cost matrix obtained and achieve an output from all the dichotomizers which takes into account the requirements of the original multi-class cost matrix. To evaluate the effectiveness of the method, a large number of tests has been performed on real data sets. The first experimental results show that the proposed approach is suitable for future developments in cost-sensitive application.

1 Introduction

A diffused technique to face a classification problem with many possible classes is to decompose the original problem into a set of two-class problems. The rationale of this approach relies on the stronger theoretical roots and better comprehension characterizing two class classifiers *(dichotomizers)* such as Perceptrons or Support Vector Machines. Moreover, with this method it becomes possible to employ in multi class problems some dichotomizers which are very effective in two-class problems but are not capable to directly perform multi-class classification.

In this framework, Error Correcting Output Coding (ECOC) has emerged as a well established technique for many applications in the field of Pattern Recognition and Data Mining, mainly for its good generalization capabilities. In short, ECOC decomposition labels each class with a bit string *(codeword)* of length L, higher than the number of classes. The codewords are arranged as rows of a *coding matrix,* whose columns define each a two-class problem; thus, for each problem, the set of the original classes parts into two complementary superclasses. On such problems induced by the coding matrix, L dichotomizers have to be trained in the learning phase. In the

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 204–213, 2004.

operating phase, the dichotomizers will provide a string of L outputs for each sample to be classified. The Hamming distance of such string from each of the codewords of the coding matrix is then evaluated and the class that corresponds to the nearest codeword is chosen. Usually, the codewords are chosen so as to have a high Hamming distance between each other; in this way, ECOC is robust to potential errors made by the dichotomizers.

The reasons for the classification efficiency exhibited by ECOC seem to be the reduction of both bias and variance [1] and the achievement of a large margin [2,3]. After the seminal paper by Dietterich and Bakiri [4], many studies have been proposed which have analyzed several aspects of ECOC such as the factors affecting the effectiveness of ECOC classifiers [5], techniques for designing codes from data [6], evaluations of coding and decoding strategies [3,7].

However a point not yet considered is how to devise an ECOC system for cost-sensitive classification. This is a significant point in many real problems such as automated disease diagnosis, currency recognition, speaker identification, and fraud detection in which different classification errors frequently have consequences of very different significance. For example, in systems for the automatic diagnosis, classifying a healthy patient as sick is much less critical than classifying a sick patient as healthy or misrecognizing the disease, because the first error can be successively corrected at the cost of a further analysis, while there could be no chance to recover the second error. This is also true for the correct classifications: the benefit obtained when recognizing the correct disease for a sick patient is much higher than the identification of a healthy patient. For this reason, the classification systems used in such situations must take into account the different costs and benefits (collected in a cost matrix) which the different decisions can provide and thus should be tuned accordingly. In multi-class classifiers, such task is usually accomplished by modifying the learning algorithm used during the training phase of the classifier or by tuning the classifier after the learning phase.

In this paper we propose a method for building cost-sensitive ECOC multi-class classifiers. Starting from the cost matrix for the multi-class problem and from the code matrix employed, a cost matrix is derived for each of the binary problems induced by the columns of the code matrix. In this way it is possible to tune the single dichotomizer according to the cost matrix obtained and achieve an output from the dichotomizers which takes into account the requirements of the original multi-class cost matrix.

In the rest of the paper we present, after a short description of the ECOC approach, some issues about two-class cost sensitive classification; then we describe how to decompose the original multi-class cost matrix in more two-class cost matrices. A conclusive section describes the results obtained from experiments performed on real data set.

2 The ECOC Approach

The Error Correcting Output Coding has been introduced to decompose a multi-class problem into a set of complementary binary problems. Each class label is represented

by a bit string of length L, called codeword, with the only requirement that distinct classes are represented by distinct bit strings. If n is the number of the original classes, a code is a $n \times L$ matrix $M = \{M_{hk}\}$ where $M_{hk} \in \{-1,+1\}$. Each row of M corresponds to a codeword for a class, while each column corresponds to a binary problem. In this way, the multi-class problem is reduced to L binary problems on which L dichotomizes have to be trained. An example of coding matrix with $n = 5$ and $L = 7$ is shown in table 1.

Table 1. A coding matrix for 5 classes and 7 dichotomizers

Classes \ dichotomizers	1	2	3	4	5	6	7
A	+1	-1	+1	-1	+1	+1	-1
B	-1	+1	+1	-1	+1	-1	-1
C	+1	-1	+1	+1	-1	+1	-1
D	-1	+1	+1	-1	+1	+1	-1
E	-1	-1	-1	+1	+1	-1	+1

However, each dichotomizer is learned from a finite set and thus, when classifying a new sample, its prediction could be wrong. This does not necessarily lead to an irrecoverable error in the multi-class problem since the code matrix is built by n distinct binary strings of length $L > n$, so as to make the Hamming distance between every pair of strings as large as possible. In fact, the minimum Hamming distance d between any pair of codewords is a measure of the quality of the code, because the code is able to correct at least $\lfloor (d-1)/2 \rfloor$ single bit errors. In this way, a single bit error does not influence the result, as it can happen when using the usual one-per-class coding, where the Hamming distance between each pair of strings is 2.

To classify a new sample x, a vector of binary decisions is computed by applying each of the learned dichotomizers to x; to decode the resulting vector, i.e. to pass from the binary to the multi-class problem, the most common approach consists in evaluating the Hamming distances between the vector and the codewords of the matrix and choose for the nearest code word, i.e. for the minimum Hamming distance. Other decoding rules have been proposed which are based on a Least Squares approach [7] or on the loss function employed in the training algorithm of the dichotomizer [3], but we will not consider them in this paper.

3 Cost-Sensitive Dichotomizers

Before analyzing cost-sensitive classification in the multi-class case, it is convenient to focus preliminarily on the two-class problem.

In this case, a sample can be assigned to one of two mutually exclusive classes that can be generically called *Positive (P)* class and *Negative (N)* class. The set of samples classified as "positive" by the dichotomizer will contain some actually-positive sam-

ples correctly classified and some actually-negative samples incorrectly classified. Hence, two appropriate performance figures are given by the *True Positive Rate (TPR),* i.e. the fraction of actually-positive cases correctly classified, and by the *False Positive Rate (FPR),* given by the fraction of actually-negative cases incorrectly classified as "positive". In a similar way, it is possible to evaluate the *True Negative Rate (TNR)* and the *False Negative Rate (FNR).* It is worth noting that only two indices are actually necessary because the following relations hold:

$$FNR=1-TPR \quad TNR=1-FPR\,. \tag{1}$$

In cost sensitive applications, every decision taken by the classifier involves a cost which estimates the penalty (benefit) produced by an error (by a correct decision). In many applications the two kinds of error (false positive and false negative) are not equally costly as well as the value of the benefit obtained can depend on the class of the sample correctly identified. Hence, we have to consider a cost matrix similar to the one described in table 2. It is worth noting that, while *CFN* and *CFP* have positive values, *CTP* and *CTN* are negative costs since they actually represent a benefit.

Table 2. Cost matrix for a two class problem

		True class: *N*	True class: *P*
Predicted Class	*N*	*CTN*	*CFN*
	P	*CFP*	*CTP*

With such assumptions, an estimate of the effectiveness of a dichotomizer working in a cost sensitive application can be given by the expected classification cost *(EC)* defined as:

$$\begin{aligned} EC = {} & p(P)\cdot CFN\cdot FNR + p(N)\cdot CFP\cdot FPR + \\ & p(P)\cdot CTP\cdot TPR + p(N)\cdot CTN\cdot TNR \end{aligned} \tag{2}$$

where $p(P)$ and $p(N)$ are the a priori probabilities of the positive and negative classes.

Eq. (2) shows the most general formulation for the expected cost. It can be simplified taking into account that results of the selection of the optimal decision are unchanged if each entry of the cost matrix is multiplied by a positive constant and/or is added by a constant [8]. Hence an equivalent form for the cost matrix in table 2 is given in table 3:

Table 3. A cost matrix equivalent to the cost matrix in tab. 2

		True class: *N*	True class: *P*
Predicted Class	*N*	0	$\frac{CFN-CTN}{CFP-CTN}$
	P	1	$\frac{CTP-CTN}{CFP-CTN}$

The corresponding expression for the expected cost is:

$$EC = p(P)\cdot\frac{CFN-CTN}{CFP-CTN}\cdot FNR + p(N)\cdot FPR + p(P)\cdot\frac{CTP-CTN}{CFP-CTN}\cdot TPR \tag{3}$$

Actually, since FNR and TPR depend on each other, the expression can be further simplified:

$$\begin{aligned} EC &= p(P)\cdot\frac{CFN-CTN}{CFP-CTN}\cdot FNR + p(N)\cdot FPR + p(P)\cdot\frac{CTP-CTN}{CFP-CTN}\cdot(1-FNR) = \\ &= p(P)\cdot\frac{CFN-CTP}{CFP-CTN}\cdot FNR + p(N)\cdot FPR + p(P)\cdot\frac{CTP-CTN}{CFP-CTN} \end{aligned} \tag{4}$$

For the classification purposes, the last term can be neglected since it represents only an offset which does not affect the choice of the decision rule minimizing the expected cost. Therefore the corresponding cost matrix becomes:

Table 4. Cost matrix for a two class problem

		True class: N	True class: P
Predicted Class	N	0	$\frac{CFN-CTP}{CFP-CTN}$
	P	1	0

And, consequently, the expressions of the expected cost associated to the cost matrices shown in table 4 is:

$$EC = p(N)\cdot FPR + p(P)\cdot FNR\cdot\rho \tag{5}$$

We can thus realize that, in two-class problems, the cost matrix has actually only one degree of freedom given by the ratio $\rho = \frac{CFN-CTP}{CFP-CTN}$. As a consequence, all problems having equal ρ are equivalent, i.e. they have the same optimal decision rule.

Some conditions must hold for the cost matrix to be realistic: ρ must be positive, higher than zero and finite. In fact, if $\rho \le 0$ the minimization of the expected cost (see eq. 5) will lead to the trivial decision rule which assigns all the samples to the negative class. On the contrary, if $\rho \to +\infty$ the optimal decision rule will tend to classify all samples as positive. This implies that sign(*CFN-CTP*)=sign(*CFP-CTN*), $CFN \neq CTP$ and $CFP \neq CTN$.

4 Evaluating the Two-Class Costs from the Multi-class Costs

Let us now consider a multi class problem with n classes to be reduced to L binary problems by using a $n \times L$ coding matrix $M=\{M_{hk}\}$ where M_{h*} is the codeword for the

class h and M_{hk} is the label assumed by a sample belonging to the class h in the binary problem induced by the k-th column. Moreover, let us assume that the costs of the multi-class problems are described by a $n \times n$ cost matrix $C = \{C_{ij}\}$ where $C_{ij} > 0$ represents the cost produced by assigning to the class j a sample actually belonging to the class i; the cost for a correct classification is null, i.e. $C_{ii} = 0 \ \forall i$.

For each column k, the n original classes are clustered into two classes, labelled with -1 and +1, which can be identified with the class N and the class P introduced in the previous section. For this reason, let us define $N^{(k)} = \{h \mid M_{hk} = -1\}$ the set of classes labelled with -1 and $P^{(k)} = \{h \mid M_{hk} = +1\}$ the set of classes labelled with +1 in the k-th binary problem. In an analogous way, let us call $C_{FP}^{(k)}$ and $C_{FN}^{(k)}$ the cost produced in the operative phase by the dichotomizer trained on the k-th problem when it erroneously assigns to $P^{(k)}$ a sample belonging to $N^{(k)}$ and vice versa.

To establish the values of $C_{FP}^{(k)}$ and $C_{FN}^{(k)}$, let us consider which are the consequences on the multi-class problem of an error made by the k-th dichotomizer. A false positive error moves one unit away from the true codewords containing a -1 in the k-th position toward the erroneous codewords containing a +1 in the same position. In particular, if r and s are two classes such that $r \in N^{(k)}$ and $s \in P^{(k)}$, a false positive error made by the k-th dichotomizer on a sample belonging to r will move one unit from the correct codeword of r, M_{r*}, toward the codeword of s, M_{s*}. Let us call $d(M_{r*},M_{s*})$ the Hamming distance existing between M_{r*} and M_{s*}; if there were errors also on the other $d(M_{r*},M_{s*})$-1 bits in which the two codewords differ, an error (with a cost equal to C_{rs}) would be generated in the multi-class problem. The contribution to such error given by the false positive produced by the k-th dichotomizer can be hence estimated as $\frac{1}{d(M_{r*},M_{s*})}$; as a consequence, the cost of the false positive related to the possible misclassification between r and s can be estimated as $\frac{C_{rs}}{d(M_{r*},M_{s*})}$.

Fig. 1 shows an example with reference to the coding matrix given in table 1. In particular, we are considering a false positive produced by the 4th dichotomizer when assigning to class E a sample of class A; to this aim, let us remark that $N^{(4)}$ collects classes A, B and D, while $P^{(4)}$ contains classes C and E.

Actually, the false positive moves the M_{r*} toward all the codewords belonging to $P^{(k)}$ and thus the cost related to all the possible misclassifications involving the class r can be estimated as (see fig. 2):

$$\sum_{s \in P^{(k)}} \frac{C_{rs}}{d(M_{r*},M_{s*})} \tag{6}$$

Eventually, we have to extend such evaluation to all the classes belonging to $N^{(k)}$. The conclusion is that the cost for a false positive made by the k-th dichotomizer is related to the risk of misclassifying one of the classes belonging to $N^{(k)}$ with one from $P^{(k)}$ and an estimate of its value is:

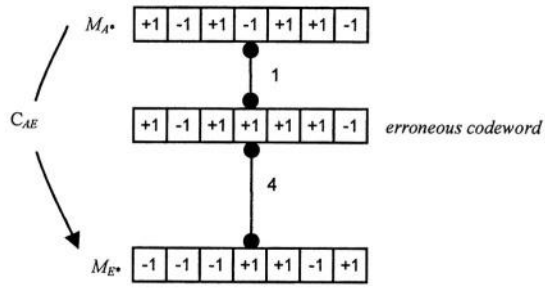


Fig. 1. Estimating the cost of a false positive produced by the 4th dichotomizer of cost matrix shown in table 1 when assigning a sample of class *A* to class *E*. The cost is evaluated as $C_{AE}/5$ where C_{AE} is the cost in the multi-class cost matrix and $\mathrm{d}(M_{A^*},M_{E^*}) = 5$

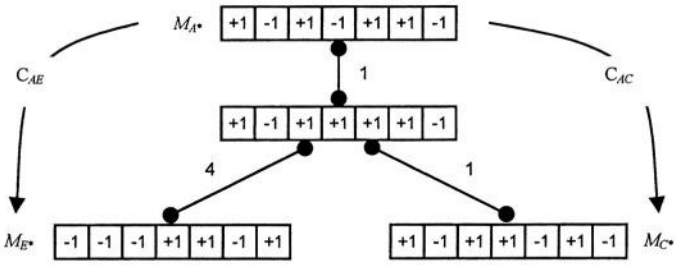


Fig. 2. The false positive produced by the 4th dichotomizer on a sample of class *A* makes the codeword M_{A^*} move toward $P^{(4)}$, with a cost estimated as $C_{AE}/5 + C_{AC}/2$.

$$C_{FP}^{(k)} = \sum_{r \in N^{(k)}} \sum_{s \in P^{(k)}} \frac{C_{rs}}{\mathrm{d}(M_{r^*}, M_{s^*})} \tag{7}$$

Likewise, it is possible to estimate the cost for a false negative made by the *k*-th dichotomizer since it is related to the risk of misclassifying one of the classes belonging to $P^{(k)}$ with one from $N^{(k)}$:

$$C_{FN}^{(k)} = \sum_{u \in P^{(k)}} \sum_{v \in N^{(k)}} \frac{C_{uv}}{\mathrm{d}(M_{u^*}, M_{v^*})} \tag{8}$$

In this way, we can define for the *k*-th dichotomizer a cost matrix similar to the one shown in table 4 with cost ratio:

$$\rho^{(k)} = \frac{\sum_{u \in P^{(k)}} \sum_{v \in N^{(k)}} \frac{C_{uv}}{\mathrm{d}(M_{u^*}, M_{v^*})}}{\sum_{r \in N^{(k)}} \sum_{s \in P^{(k)}} \frac{C_{rs}}{\mathrm{d}(M_{r^*}, M_{s^*})}} \tag{9}$$

It is easy to see that the conditions for a realistic cost matrix (i.e. $0 < \rho^{(k)} < +\infty$) are satisfied since $C_{ij} > 0 \ \forall \ i \neq j$ and $\mathrm{d}(M_{r^*},M_{s^*}) \neq 0 \ \forall r \neq s$.

5 Experimental Results

To evaluate the effectiveness of the proposed approach we have made several experiments on some data sets with different numbers of classes and dichotomizers with different architectures; moreover, a comparison technique has been devised to assure that the outcomes obtained were statistically significant.

The data sets used are publicly available at the UCI Machine Learning Repository [9]; all of them have numerical input features and a variable number of classes. All the features were previously rescaled so as to have zero mean and unit standard deviation. More details of data sets are given in table 5. The table provides also the type of coding matrix used for each data set. We choose an exhaustive code [4] for those data sets that have a number of classes lower than 8 and a BCH code [4] for those having a number of classes greater or equal to 8. In particular, for Vowel and Letter Recognition data sets, we adopted, respectively, ECOC codes 15-11 and 63-26 available at http://web.engr.oregonstate.edu/~tgd.

Table 5. Data sets and coding matrices used in the experiments

data Set	classes	feat.	samples	train. set	test set	valid. set	coding matrices	code length
Ann-thyroid	3	21	7200	5040	1080	1080	Exhaustive	3
Dermatology.	6	34	358	252	54	52	Exhaustive	31
Glass	6	9	214	149	32	33	Exhaustive	31
Sat Image	6	36	6435	4505	965	965	Exhaustive	31
Segmentation	7	18	2310	1617	350	343	Exhaustive	63
Optdigits	10	62	5620	3935	844	841	BCH 31-21	31
Pendigits	10	16	10992	7696	1647	1649	BCH 31-21	31
Vowel	11	10	990	693	154	143	Diett 15-11	15
Letter Rec.	26	16	20000	14001	2998	3001	Diett 63-26	63

The dichotomizers employed were Support Vector Machines implemented by means of SVMlight tool [10] available at http://svmlight.joachims.org.

In order to build dichotomizers tuned on the cost matrices determined according the method described in Section 4, we have adopted a *post-learning scheme* [11] which evaluates a threshold t to be imposed on the output of the SVM, so as to attribute the sample to be classified to the class N if the SVM output is less than t and to the class P otherwise. The threshold is chosen so as to minimize the expected classification cost on a validation set; this is an approach more general than the standard SVM setting which uses a zero threshold.

The aim of the experiments was to verify if our method can give a real improvement and thus we have compared the classification costs obtained when the described method is applied (i.e. when the dichotomizers are tuned according to the cost matrices built as seen in Section 4) with the classification costs obtained by using the standard ECOC architecture (i.e. the dichotomizers are employed without any cost-sensitive tuning). With reference to the value assumed for thresholding the SVM output, hereafter we will denote the first case as CST (Cost Sensitive Threshold) and the second one as ZT (Zero Threshold).

To avoid any bias in the comparison, 12 runs of a multiple hold-out procedure were performed on all data sets. In each run, the data set was split in three subsets: a training set (containing 70% of the samples of each class), a validation set and a test set (each containing 15% of the samples of each class); the final size of each of these sets is given in table 5. The validation set was used to evaluate the optimal threshold of the CST, while it was considered as part of the training set in the ZT method.

After the encoding phase, we obtained a different training set for each SVM. Since in such training sets the distribution between the two classes was frequently very skewed, the sets were balanced [12] before training so as to have a more effective learning for the SVMs. The majority class was randomly undersampled so as to equate the number of samples in the two classes.

The two classification costs to be compared were evaluated on the test set, thus obtaining, for a given data set, 12 different values for each of the costs required. To establish if the classification cost obtained by the CST was significantly better than the cost of ZT, we used the Wilcoxon rank-sum test [13], that verifies if the mean of the CST is higher than, lower than or undistinguishable from the mean of the costs of ZT. All the results were provided with a significance level equal to 0.05.

To obtain a result unbiased with respect to the particular cost values, we apply the approach proposed in [14]: a hundred of different cost matrices have been used whose elements were randomly generated according to a uniform distribution over the range [1,10]. For each cost matrix, the test before described has been repeated.

In table 6 are presented the results of the evaluation of the CST scheme compared with ZT; the experiments were performed on SVMs with three different kernels: linear, 2^{nd} degree polynomial and RBF. Each row of the table corresponds to a data set, while a group of three columns refers to the particular kernel used. For each kernel, the columns contains a value which indicates the number of runs (out of 100) for which the CST classification has produced a classification cost respectively undistinguishable from, higher than or lower than the classification cost obtained with ZT.

Table 6. Result of comparison between CST and ZT for different kernels

	Linear			2-degree Polynomial			RBF		
Ann-thyroid	28	5	67	0	0	100	0	0	100
Dermatology	8	92	0	74	23	3	0	0	100
Glass	62	38	0	36	9	55	16	1	83
Sat Image	0	100	0	22	77	1	15	3	82
Segmentation	42	58	0	57	43	0	6	0	94
Optdigits	1	99	0	19	81	0	6	0	94
Pendigits	51	44	5	44	56	0	30	18	52
Vowel	53	16	31	35	65	0	30	38	32
Letter Rec.	16	5	79	45	55	0	53	24	23

The results appear to be quite dependent on the kernel adopted for the SVM. In fact, while for the linear kernel the percentage of cases in which CST performs

equally or better than ZT is only 49.22%, this result grows to 54.56% for the 2-degree polynomial kernel and to 90.67% for the RBF kernel. Such dependence on the particular dichotomizer used is a more general property of ECOC, as pointed out in [5]. The outcomes produced by the 2^{nd} degree polynomial and by the RBF kernel can be explained by taking into account the higher complexity of such dichotomizers which are able to discriminate better than the linear dichotomizer. However, there are two situations, i.e. Vowel and Letter Recognition, in which this is not true. This cases could be due to overfitting phenomena which affect the performance of the 2^{nd} degree polynomial and RBF kernels.

In summary, the experiments show that the proposed method can achieve an improvement in terms of classification cost when using an ECOC-based classification system in cost-sensitive applications. It remains to be defined which kind of dichotomizer is best suited for a particular application, but this is a more general, open problem that involves the estimation of the complexity of the data characterizing the application [5].

References

1. Kong, E., Dietterich, T.G.: Error Correcting Output Coding Corrects Bias and Variance. Proc. 12th Int. Conf. on Machine Learning (1995), San Francisco, CA.
2. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.: Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics 26 (1998) 1651-1686
3. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. Journal of Machine Learning Research 1 (2000) 113-141
4. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error Correcting Output Codes. Journal of Artificial Intelligence Research 2 (1995) 263-286
5. Masulli, F., Valentini, G.: Effectiveness of Error Correcting Output Coding Methods in Ensemble and Monolithic Learning Machines. Pattern Analysis and Applications to appear
6. Alpaydin, E., Mayoraz, E.: Learning Error Correcting Output Codes from Data. Proc. Int. Conf. on Artificial Neural Networks (1999), Edinburgh, UK
7. Windeatt, T., Ghaderi, R.: Coding and Decoding Strategies for Multi Class Learning Problems. Information Fusion 4 (2003) 11-21
8. Elkan, C.: The Foundations of Cost-Sensitive Learning. Proc. 17th IJCAI (2001)
9. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases. (1998) [www.ics.uci.edu/~mlearn/MLRepository.html]
10. Joachims, T.: Making Large-Scale SVM Learning Practical, in Schölkopf, B., Burges, C.J.C., Smola, A.J., eds., Advances in Kernel Methods, MIT Press (1999) 169-184
11. Tortorella, F.: An Empirical Comparison of In-Learning and Post-Learning Optimization Schemes for Tuning the Support Vector Machines in Cost-Sensitive Applications. Proc. 12th Int. Conf. on Image Anal. and Proc., IEEE Computer Society Press (2003), 560-565
12. Weiss, G., Provost, F.: The Effect of Class Distribution on Classifier Learning: An Empirical Study. Tech. Rep. ML-TR-44, Dept. of Comp. Sci., Rutgers University (2001)
13. Walpole, R.E., Myers, R.H., Myers, S.L.: Probability and Statistics for Engineers and Scientists. 6th ed., Prentice Hall Int., London, (1998)
14. Margineantu, D.D., Dietterich, T.G., Bootstrap Methods for the Cost-Sensitive Evaluation of Classifiers. Proc. Int. Conf. Machine Learning ICML-2000 (2000), 582-590

Building Graph-Based Classifier Ensembles by Random Node Selection

Adam Schenker[1], Horst Bunke[2], Mark Last[3], and Abraham Kandel[1,4]

[1] University of South Florida, Tampa FL 33620, USA
[2] University of Bern, CH-3012 Bern, Switzerland
[3] Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[4] Tel-Aviv University, Tel-Aviv 69978, Israel

Abstract. In this paper we introduce a method of creating structural (i.e. graph-based) classifier ensembles through random node selection. Different k-Nearest Neighbor classifiers, based on a graph distance measure, are created automatically by randomly removing nodes in each prototype graph, similar to random feature subset selection for creating ensembles of statistical classifiers. These classifiers are then combined using a Borda ranking scheme to form a multiple classifier system. We examine the performance of this method when classifying a web document collection; experimental results show the proposed method can outperform a single classifier approach (using either a graph-based or vector-based representation).

1 Introduction

Classifiers are machine learning algorithms which attempt to assign a label (a classification) to objects based on comparisons to previously seen training examples. One application of such a system might be to automatically categorize documents into specified categories to allow for later browsing or retrieval [1][2][3]. The performance of classifiers is measured by their ability to accurately assign the correct class label to previously unseen data. In order to improve classifier accuracy, the idea of creating classifier ensembles has been proposed [4] [5]. This methodology involves combining the output of several different (usually independent) classifiers in order to build one large, and hopefully more accurate, multiple classifier system. Several approaches have been proposed to create classifier ensembles. Bagging, for instance, creates classifiers by randomly selecting the group of training examples to be used for each classifier [6]. A similar idea is that of random feature subset selection [7]. In this method, we randomly select the features (dimensions) to be used for each feature vector to create a group of classifiers.

Common to many classification algorithms, including those used in classifier ensembles, is that they are designed so as to work on data which is represented by vectors (i.e. sets of attribute values). However, using such vector representations may lead to the loss of the inherent structural information in the original data. Consider, for example, the case of classifying web documents based on

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 214–222, 2004.

their content. Under the vector space model of document representation [8] each term which may appear on a document is represented by a vector component (or dimension). The value aasociated with each dimension indicates either the frequency of the term or its relative importance according to some weighting scheme. A problem with representing web documents in this manner is that certain information, such as the order of term appearance, term proximity, term location within the document, and any web specific information, is lost under the vector model. Graphs are a more robust data structure which are capable of capturing and maintaining this additional information.

Until recently, there have been no mathematical frameworks available for dealing with graphs in the same fashion that we can deal with vectors in a machine learning system. For example, the k-Nearest Neighbors (k-NN) classification algorithm requires the computation of similarity (or distance) between pairs of objects. This is easily accomplished with vectors in a Euclidean feature space, but until recently it has not been possible with graphs [9] [10] [11]. Given these new graph-theoretic foundations, a version of the k-Nearest Neighbors algorithm which can classify objects which are represented by graphs rather than by vectors has been proposed [12][13]. Experimental results comparing the classification accuracy of the graph-theoretic method with traditional vector-based k-NN classifiers showed that the performance when representing data by graphs usually exceeds that of the corresponding vector-based approach [12] [13].

In this paper, we introduce a technique for creating graph-based classifier ensembles using random node selection. To our knowledge, this is the first time such an approach has been taken to build structural classifier ensembles. In this paper we will also consider ensembles that include both structural, i.e. graph-based, and statistical, i.e. feature vector-based, classifiers. In [14] such an approach, using one statistical and one structural classifier, has been proposed. However, the classifiers used in [14] were both designed by hand. By contrast, our method allows for automatic ensemble generation out of one single structural base classifier. We will perform experiments in order to test the accuracy of the classifier ensembles created with our novel procedure. The data set we use is a web document collection; the documents will be represented by both graphs and vectors and we will measure the accuracy of assigning the correct category to each document using the proposed method.

The remainder of the paper is organized as follows. In Sect. 2 we will explain how the web documents are represented by graphs. The details of the graph-based k-Nearest Neighbors algorithm are given in Sect. 3. In Sect. 4 we describe the method used to combine the k-NN classifiers into an ensemble. We present the experimental results in Sect. 5. Finally, in Sect. 6, we give some concluding remarks.

2 Graph Representation of Web Documents

Content-based classification of web documents is useful because it allows users to more easily navigate and browse collections of documents [1][3]. However, such

classifications are often costly to perform manually, as this requires a human expert to examine the content of each web document and then make a determination of its classification. Due to the large number of documents available on the Internet in general, or even when we consider smaller collections of web documents, such as those associated with corporate or university web sites, an automated system which performs web document classification is desirable in order to reduce costs and increase the speed with which new documents are classified.

In order to represent web documents by graphs during classification, and thus maintain the information that is usually lost in a vector model representation, we will use the following method. First, each term (word) appearing in the web document, except for stop words such as "the", "of", and "and" which convey little information, becomes a node in the graph representing that document. This is accomplished by labeling each node with the term it represents. Note that we create only a single node for each unique word even if a word appears more than once in the text. Second, if word a immediately precedes word b somewhere in a "section"s of the web document, then there is a directed edge from the node corresponding to a to the node corresponding to b with an edge label s. We do not create an edge when certain punctuation marks (such as a period) are present between two words. Sections we have defined are: *title,* which contains the text related to the web document's title and any provided keywords; *link,* which is text appearing in clickable hyperlinks on the web document; and *text,* which comprises any of the readable text in the web document (this includes link text but not title and keyword text). Next, we remove the most infrequently occurring words for each document by deleting their corresponding nodes, leaving at most m nodes per graph (m being a user provided parameter). This is similar to the dimensionality reduction process for vector representations but with our method the term set is usually different for each document. Finally, we perform a simple stemming method and conflate terms to the most frequently occurring form by re-labeling nodes and updating edges as needed. An example of this type of graph representation is given in Fig. 1. The ovals indicate nodes and their corresponding term labels. The edges are labeled according to title (TI), link (L), or text (TX). The document represented by the example has the title "YAHOO NEWS", a link whose text reads "MORE NEWS", and text containing "REUTERS NEWS SERVICE REPORTS". Note there is no restriction on the form of the graph and that cycles are allowed. While this appears superficially similar to the bigram, trigram, or N-gram methods [15], those are statistically-oriented approaches based on word occurrence probability models. The method described above does not require or use the computation of term probabilities.

3 Graph-Theoretic k-Nearest Neighbors

In this section we describe the k-Nearest Neighbors (k-NN) classification algorithm and how we can easily extend it to work with graph-based data. The basic k-NN algorithm is given as follows [16]. First, we have a set of training examples

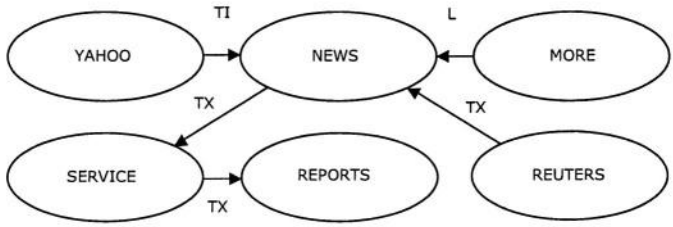


Fig. 1. Example of a graph representation of a document

(or *prototypes*). In the traditional k-NN approach these are numerical vectors in a Euclidean feature space. Each of these prototypes is associated with a label which indicates to what class the prototype belongs. Given a new, previously unseen instance, called an input instance, we attempt to estimate which class it belongs to. Under the k-NN method this is accomplished by looking at the k training instances closest (i.e. with least distance) to the input instance. Here k is a user provided parameter and distance is usually defined to be the Euclidean distance. However, in information retrieval applications, the cosine similarity or Jaccard similarity measures [8] are often used due to their length invariance properties.

Once we have found the k training instances nearest to the input instance using some distance measure, we estimate the class of the input instance by the majority held among the k training instances. This class is then assigned as the predicted class for the input instance. If there are ties due to more than one class having equal numbers of representatives amongst the nearest neighbors we can either choose one class randomly or we can break the tie with some other method, such as selecting the tied class which has the minimum distance neighbor. For the experiments in this paper we will use the latter method, which in our experiments has shown a slight improvement over random tie breaking.

In order to extend the k-NN method to work with graphs instead of vectors, we only need a distance measure which computes the distance between two graphs instead of two vectors, since both training and input instances will be represented by graphs. One such graph distance measure is based on the maximum common subgraph [9]:

$$dist(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_2|)}. \tag{1}$$

Here G_1 and G_2 are graphs, $mcs(G_1, G_2)$ is their maximum common subgraph, $max(\cdots)$ is the standard numerical maximum operation, and $|\cdots|$ denotes the size of the graph, which is taken in this context to be the number of nodes and edges in the graph. This distance measure has been shown to be a metric [9]. More information about graph-theoretic k-NN classifiers and applications to web document classification can be found in [12][13].

4 Creating Ensembles of Structural Classifiers

In this section we describe our proposed method of creating graph-based classifiers and combining them into an ensemble. As mentioned earlier, the goal of creating an ensemble of classifiers is to achieve a higher overall accuracy than any single classifier in the ensemble by combining the output of the individual classifiers [4] [5]. The classifiers used in our ensembles perform the k-Nearest Neighbors method described in Sect. 3. Different graph-based classifiers are generated by randomly removing nodes (and their incident edges) from the training graphs until a maximum number of nodes is reached for all graphs. We create several graph-based classifiers using this method, and each becomes a classifier in the ensemble.

For each classifier, we output the three top ranked classification labels. The ranked outputs from each classifier are then combined using a Borda count [17]. The first ranked class receives a vote of 3, the second a vote of 2, and the third a vote of 1. Using the Borda count we select the class with the highest total vote count as the predicted class for the ensemble, with ties broken arbitrarily.

5 Experiments and Results

In order to evaluate the performance of our proposed method of creating classifier ensembles using random node selection, we performed several experiments on a collection of web documents. Our data set contains 185 documents, each belonging to one of ten categories, and was obtained from ftp.cs.umn.edu/dept/users/boley/PDDPdata/. For our experiments, we created graphs from the original web documents as described in Sect. 2. For the parameter m, which indicates the maximum number of the most frequent nodes to retain in each graph, we used a value of 100 for our experiments. In addition to the graph-based classifiers, we also include a single vector-based k-NN classifier in the ensemble. For the vector-based classifiers, we used a standard term–document matrix of 332 dimensions that was provided with the data set; we used a distance based on the Jaccard similarity [8] for the vector-based classifiers.

The experimental results are given in Table 1. Each row indicates an experiment with different parameter values, which are shown in the three leftmost columns. The parameters are *NN,* the maximum number of nodes to randomly be included in each prototype graph; *NC,* the number of classifiers in the ensemble; and k, which is the parameter k used in the k-NN algorithm (the number of nearest neighbors). Note that in a given ensemble there will be $NC - 1$ graph-based classifiers created through random node selection and a single vector-based classifier.

The results of each experiment are shown in the next six columns as classification accuracy measured by leave-one-out. *Min* is the accuracy of the worst single classifier in the ensemble. Similarly, *Max* is the accuracy of the best single classifier in the ensemble. *Ens* is the accuracy of the combined ensemble using Borda count as described above. *Oracle* is the accuracy if we assume that when

at least one individual classifier in the ensemble correctly classifies a document the ensemble will also correctly classify the document; this gives us an upper bound on the best possible accuracy of the ensemble if we were able to find an optimal combination method but leave the classifiers themselves as they are. *BL (G)* (baseline graph-based classifier) gives the accuracy of a single graph-based classifier using the standard k-NN method with the full sized training graphs ($m = 100$) for a baseline comparison; similarly, *BL (V)* (baseline vector-based classifier) gives the accuracy of the vector-based k-NN classifier used in the ensemble. The final column, *Imp,* is the difference between *Ens* and *BL (G),* which indicates the performance improvement realized by the ensemble over the baseline graph-based classifier. The average of each column is shown in the bottom row of the table.

Table 1. Experimental results

NN	NC	k	Min	Max	Ens	Oracle	BL (G)	BL (V)	Imp
50	3	1	68.65%	73.51%	83.24%	91.35%	80.00%	73.51%	3.24%
50	3	3	70.27%	74.59%	82.16%	92.43%	81.62%	74.59%	0.54%
50	3	5	73.51%	74.05%	81.08%	92.43%	83.24%	74.05%	-2.16%
50	5	1	63.78%	74.05%	83.78%	93.51%	80.00%	73.51%	3.78%
50	5	3	72.43%	78.38%	82.16%	94.59%	81.62%	74.59%	0.54%
50	5	5	72.43%	78.92%	81.62%	95.14%	83.24%	74.05%	-1.62%
50	10	1	63.78%	73.51%	80.00%	94.05%	80.00%	73.51%	0.00%
50	10	3	68.11%	79.46%	81.08%	95.68%	81.62%	74.59%	-0.54%
50	10	5	70.81%	80.54%	81.62%	95.14%	83.24%	74.05%	-1.62%
75	3	1	72.97%	76.76%	84.32%	91.35%	80.00%	73.51%	4.32%
75	3	3	74.59%	78.38%	81.08%	91.35%	81.62%	74.59%	-0.54%
75	3	5	74.05%	78.92%	83.24%	92.43%	83.24%	74.05%	0.00%
75	5	1	70.81%	74.05%	82.16%	91.35%	80.00%	73.51%	2.16%
75	5	3	72.97%	78.92%	81.08%	94.05%	81.62%	74.59%	-0.54%
75	5	5	74.05%	81.08%	85.41%	94.59%	83.24%	74.05%	2.17%
75	10	1	68.65%	76.76%	80.00%	90.81%	80.00%	73.51%	0.00%
75	10	3	72.43%	78.38%	82.70%	95.14%	81.62%	74.59%	1.08%
75	10	5	74.05%	81.08%	83.24%	95.14%	83.24%	74.05%	0.00%
Average			71.02%	77.30%	82.22%	93.36%	81.62%	74.05%	0.60%

The first thing we observe from the results is that, in every case, the accuracy of the ensemble *(Ens)* was greater than the best single classifier in the ensemble *(Max).* Additionally, the ensemble accuracy *(Ens)* was always greater than the baseline vector classifier ($BL\ (V)$). We note also that the best accuracy attained by our ensemble method was 85.41% (for $NN = 75$, $NC = 5$, $k = 5$), while the best accuracy achieved by the graph-based baseline classifiers *(BL (G))* was 83.24% (for $k = 5$); out of *BL (G), BL (V),* and *Ens, Ens* attained both the highest average and highest maximum accuracy. Note that the performance of *BL (G)* and *BL (V)* is dependent only on the parameter k. The results furthermore show that the ensemble was an improvement over the baseline graph-based classifier in 8 out of 18 cases; conversely, *BL (G)* was better than *Ens* in 6 out

of 18 cases. However, *Oracle,* in all cases, was much better than *Ens* (the maximum oracle accuracy was 95.68%). This suggests one should look at improving the combination scheme. This could be done, for example, by altering our current method of Borda ranking to include more rankings or better tie breaking procedures. Alternatively we could introduce a weighted voting scheme, where classifier weights are determined by methods such as genetic algorithms [18]. One could also take the distances returned by our k-NN classifiers into account. The number of classifiers *(NC)* and the parameter k did not seem to affect ensemble accuracy in any specific way.

6 Conclusions

In this paper we introduced the novel concept of creating structural classifier ensembles through random node selection. This is a method similar to random subset feature selection, but applied to (structural) classifiers that deal with data represented by graphs rather than vectors. The accuracy of the ensembles created using this method was slightly better, on average, than the accuracy of a baseline single classifier when classifying a collection of web documents; the best accuracy achieved by our method was also greater than the best accuracy of any baseline classifier. In addition, the overall ensemble accuracy was an improvement over the best individual classifier in the ensemble in all cases, and the oracle accuracy (which measures accuracy achieved when using an optimal combination strategy) was an improvement over the ensemble and baseline accuracies in all cases.

The work described in this paper is, to the knowledge of the authors, the first on classifier ensembles in the domain of structural pattern recognition. Our future work will be directed toward examining the effect of other parameters which were not considered in the experiments presented here, such as the maximum number of nodes after dimensionality reduction (m) and the number of nodes used for random node selection. We also plan to vary the number of classifiers in the ensemble over a wider ranger of values. As we saw in our experiments, examining the accuracy of the ensemble as if it were an oracle showed a significant potential improvement in performance. This is a strong motivation to look at further refining the classifier combination method.

There are many other open issues to explore as well. For example, instead of random node selection, we could select nodes based on some criteria, such as their degree (i.e., the number of incident edges) or a particular structural pattern they form with other nodes in the graph (chains, cliques, etc.). We have previously experimented with various other graph representations of documents [19], such as those that capture frequency information about the nodes and edges; using these representations in the context of ensemble classifiers will be a subject of future experiments. Ensemble performance could perhaps be further improved by analyzing the documents on which the baseline graph classifier outperformed the ensemble. We can also create different graph-based classifiers for an ensemble by changing the graph-theoretic distance measures used or through more well-known techniques such as bagging.

Acknowledgments

This work was supported in part by the National Institute for Systems Test and Productivity at the University of South Florida under the U.S. Space and Naval Warfare Systems Command Contract No. N00039–02–C–3244 and by the Fulbright Foundation that has granted Professor Kandel the Fulbright Research Award at Tel-Aviv University, College of Engineering during the academic year 2003–2004.

References

1. Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proceedings of SIGIR–00, 23rd ACM International Conference on Research and Development in Information Retrieval. (2000) 256–263
2. McCallum, A., Nigam, K.: A comparison of event models for naive Bayes text classification. In: Proceedings of the AAAI98 Workshop on Learning for Text Categorization. (1998)
3. Apte, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Transactions on Information Systems **12** (1994) 233–251
4. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239
5. Dietterich, T.G.: Ensemble methods in machine learning. In Kittler, J., Roli, F., eds.: First International Workshop on Multiple Classifier Systems. Volume 1857 of Lecture Notes in Computer Science. Springer (2000)
6. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
7. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 832–844
8. Salton, G.: Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading (1989)
9. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters **19** (1998) 225–259
10. Fernández, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters **22** (2001) 753–758
11. Wallis, W.D., Shoubridge, P., Kraetz, M., Ray, D.: Graph distances using graph union. Pattern Recognition Letters **22** (2001) 701–704
12. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of web documents using graph matching. International Journal of Pattern Recognition and Artificial Intelligence (to appear)
13. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of web documents using a graph model. In: Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR2003). (2003) 240–244
14. Marcialis, G., Roli, F., Serrau, A.: Fusion of statistical and structural fingerprint classifiers. In Kittler, J., Nixon, M.S., eds.: Proceedings of the 4th International Conference on Audio- and Video-Based Biometric Person Authentication, (AVBPA 2003). Volume 2688 of Lecture Notes in Computer Science., Springer (2003) 310–317
15. Tan, C.M., Wang, Y.F., Lee, C.D.: The use of bigrams to enhance text categorization. Information Processing and Management **38** (2002) 529–546

16. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
17. Ho, T.K., Hull, J.J., Srihari, S.N.: On multiple classifier systems for pattern recognition. In: Proceedings of the 11th International Conference on Pattern Recognition, The Hague, The Netherlands (1992) 84–87
18. Lam, L., Huang, Y.S., Suen, C.: Combination of multiple classifier decisions for optical character recognition. In Bunke, H., Wang, P., eds.: Handbook of Character Recognition and Document Image Analysis. World Scientific (1997) 79–101
19. Schenker, A., Last, M., Bunke, H., Kandel, A.: Graph representations for web document clustering. In: Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA). (2003) 935–942

A Comparison of Ensemble Creation Techniques

Robert E. Banfield[1], Lawrence O. Hall[1], Kevin W. Bowyer[2], Divya Bhadoria[1], W. Philip Kegelmeyer[3], and Steven Eschrich[1]

[1] Department of Computer Science & Engineering
University of South Florida
Tampa, Florida 33620-5399
{rbanfiel,hall,dbhadori,eschrich}@csee.usf.edu
[2] Computer Science & Engineering
384 Fitzpatrick Hall Notre Dame, IN 46556
kwb@cse.nd.edu
[3] Sandia National Labs, Biosystems Research Department
PO Box 969, MS 9951 Livermore, CA 94551-0969, USA
wpk@ca.sandia.gov

Abstract. We experimentally evaluated bagging and six other randomization-based ensemble tree methods. Bagging uses randomization to create multiple training sets. Other approaches, such as Randomized C4.5, apply randomization in selecting a test at a given node of a tree. Then there are approaches, such as random forests and random subspaces, that apply randomization in the selection of attributes to be used in building the tree. On the other hand boosting incrementally builds classifiers by focusing on examples misclassified by existing classifiers. Experiments were performed on 34 publicly available data sets. While each of the other six approaches has some strengths, we find that none of them is consistently more accurate than standard bagging when tested for statistical significance.

1 Introduction

Bagging [1], Adaboost.M1W [2–4], three variations of random forests [5], one variation of Randomized C4.5 [6] (which we will call by the more general name "random trees"), and random subspaces [7] are compared. With the exception of boosting, each ensemble creation approach compared here can be distributed in a simple way across a set of processors. This makes them suitable for learning from very large data sets [8–10] because each classifier in an ensemble can be built at the same time if processors are available. The classification accuracy of the approaches was evaluated through a series of 10-fold cross validation experiments on 34 data sets taken mostly from the UC Irvine repository [11]. We used the open source software package "OpenDT" [12] for learning decision trees in parallel, which outputs trees very similar to C4.5 release 8 [13], and has added functionality for ensemble creation.

Our previous results [14] showed that each of the ensemble creation techniques gives a statistically significant, though small, increase in accuracy over a

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 223–232, 2004.

single decision tree. However, in head-to-head comparisons with bagging, none of the ensemble building methods was generally statistically significantly more accurate.

This work extends our previous work [14] by comparing against the results for boosting, using ANOVA to better understand the statistical significance of the results, and increasing the number and size of the data sets used. We have also increased the number of classifiers in each ensemble, with the exception of boosting, to 1000. By using a greater number of decision trees in the ensemble and larger size data sets, our conclusions about several of the methods have changed.

2 Ensemble Creation Techniques Evaluated

Ho's random subspace method of creating a decision forest utilizes the random selection of attributes or features in creating each decision tree. Ho used a randomly chosen 50% of the attributes to create each decision tree in an ensemble and the ensemble size was 100 trees.

The random subspace approach was better than bagging and boosting for a single train/test data split for four data sets taken from the stat log project [15]. Ho tested 14 other data sets by splitting them into two halves randomly. Each half was used as a training set with the other half used as a test set. This was done 10 times for each of the data sets. The maximum and minimum accuracy results were deleted and the other eight runs were averaged. There was no evaluation of statistical significance. The conclusion was that random subspaces was better for data sets with a large number of attributes. It was not as good with a smaller number of attributes and a small number of examples, or a small number of attributes and a large number of classes. This approach is interesting for large data sets with many of attributes because it requires less time and memory to build each of the classifiers.

Breiman's random forest approach to creating an ensemble also utilizes a random choice of attributes in the construction of each CART decision tree [16, 5]. However, a random selection of a subset of attributes occurs at each node in the decision tree. Potential tests from these random attributes are evaluated and the best one is chosen. So, it is possible for each of the attributes to be utilized in the tree. The number of random attributes chosen for evaluation at each node is a variable in this approach. Additionally, bagging is used to create the training set for each of the trees. We utilized random subsets of size 1, 2 and $\lfloor \log_2(n) + 1 \rfloor$, where n is the number of attributes.

Random forest experiments were conducted on 20 data sets and compared with Adaboost on the same data sets in [5]. Ensembles of 100 decision trees were built for the random forests and 50 decision trees for Adaboost. For the zip-code data set 200 trees were used. A random 10% of the data was left out of the training set to serve as test data. This was done 100 times and the results averaged. The random forest with a single attribute randomly chosen at each node was better than Adaboost on 11 of the 20 data sets. There was no evaluation

of statistical significance. It was significantly faster to build the ensembles using random forests.

Dietterich introduced an approach which he called Randomized C4.5 [6], which comes under our more general description of random trees. In this approach, at each node in the decision tree the 20 best tests are determined and the actual test used is randomly chosen from among them. With continuous attributes, it is possible that multiple tests from the same attribute will be in the top 20.

Dietterich experimented with 33 data sets from the UC Irvine repository. For all but three of them a 10-fold cross validation approach was followed. The best result from a pruned or unpruned ensemble was reported. Pruning was done with a certainty factor of 10. The test results were evaluated for statistical significance at the 95% confidence level. It was found that Randomized C4.5 was better than C4.5 14 times and equivalent 19 times. It was better than bagging with C4.5 6 times, worse 3 times and equivalent 24 times. From this, it was concluded that the approach tends to produce an equivalent or better ensemble than bagging. It has the advantage that you do not have to create multiple instances of a training set.

3 Algorithm Modifications

We describe our implementation of random forests and a modification to Dietterich's randomized C4.5 method. In OpenDT, like C4.5, a penalty is assessed to the information gain of a continuous attribute with many potential splits. In the event that the attribute set randomly chosen provides a "negative" information gain, our approach is to randomly re-choose attributes until a positive information gain is obtained, or no further split is possible. This enables each test to improve the purity of the resultant leaves. This approach was also used in WEKA [17].

We have made a modification to the Randomized C4.5[1] ensemble creation method in which only the best test from each attribute is allowed to be among the best set of twenty features from which one is randomly chosen. This allows a greater chance of discrete attributes being chosen for testing when there are a large number of continuous valued attributes. We call it random trees B.

4 Experimental Results

We used 34 data sets, 32 from the UC Irvine repository [11], credit-g from NIAAD (www.liacc.up.pt/ML) and phoneme from the ELENA project. The data sets, described in Table 1, have from 4 to 69 attributes and the attributes are

[1] On a code implementation note, we allow trees to be grown to single example leaves, which we call pure trees. MINOBJS is set to one (which means a test will be attempted any time there are two or more examples at a node), tree collapsing is not allowed and dynamic changes in the minimum number of examples in a branch for a test to be used are not allowed.

Table 1. Description of data sets attributes and size.

Data Set	# attributes	# continuous attributes	# examples	# classes
anneal	38	6	898	6
audiology	69	0	226	24
autos	25	15	205	7
breast-w	9	9	699	2
breast-y	9	0	286	2
credit-a	15	6	690	2
credit-g	20	7	1000	2
glass	9	9	214	7
heart-c	13	5	303	2
heart-h	13	5	294	2
heart-s	13	5	123	2
heart-v	13	5	200	2
hepatitis	19	6	155	2
horse-colic	22	8	368	2
hypo	25	7	3163	2
ion	34	34	351	2
iris	4	4	150	3
krkp	36	0	3196	2
labor	16	8	57	2
led-24	24	0	5000	10
letter	16	16	20000	26
lymph	18	3	148	4
page	10	10	5473	5
pendigits	16	16	10992	10
phoneme	5	5	5404	2
pima	8	8	768	2
primary	17	0	339	22
satimage	36	36	6435	7
sick	29	7	3772	2
sonar	60	60	208	2
soybean	35	0	683	19
vehicle	18	18	846	4
voting	15	0	435	2
waveform	21	21	5000	3

a mixture of continuous and nominal values. The ensemble size was 50 for the boosting approach, and 1000 trees for each of the other approaches. While 1000 trees are more than the original authors suggested, this number was chosen so that more than enough trees were present in the ensemble. Unlike boosting, Breiman has argued these other techniques do not overfit as more classifiers are added to the ensemble [5].

For the random trees B approach, we used a random test from the 20 attributes with maximal information gain. In the random subspace approach of Ho, half ($\lceil n/2 \rceil$) of the attributes were chosen each time. For the random forest approach, we tested using a single attribute, 2 attributes and $\lfloor log_2\ n + 1 \rfloor$ attributes (which will be abbreviated as Random Forests-lg in the following).

For each data set, a 10-fold cross validation was done. For each fold, an ensemble is built by each method and tested on the held out data. This allows for statistical comparisons between approaches to be made. Each ensemble consists solely of unpruned trees. ANOVA was first used to determine which data sets showed statistically significant differences at a specified confidence level. Subsequently, a paired t-test was used to determine, at the same confidence level, whether a particular ensemble approach is better or worse than bagging.

Table 2 shows the comparative results at the 99% confidence interval. All ensemble creation techniques to the right of the bagging column in the table utilized bagging in creating training sets. For 30 of the 34 of the data sets none of the ensemble approaches could produce a statistically significant improvement over bagging. On two data sets, all techniques showed accuracy improvement (as indicated by a boldface number). The best ensemble building approaches by a slight margin were random forests-lg and random forests-2 which are better than bagging four times. Both of these methods are worse than bagging only twice (indicated by a number in italics). Boosting had the least wins at two, while losing to bagging once. Random subspaces lost to bagging four times and registered only three wins. Random trees B and random forests-1 were better 3 times and worse 1 time and 2 times respectively.

We can create a summary score for each ensemble algorithm by providing 1 point for a win, and 1/2 point for a tie. At the 99% confidence level the top performing ensemble methods are random forests-lg, random forests-2 and Random Trees B (18 points). All other approaches score 17.5 points or 16.5 (random subspaces).

An interesting question is how would these approaches rank if the average accuracy, regardless of significance, was the only criterion. Once again that random forests-lg and random forests-2 appear the best (25.5 and 24 points respectively). Random subspaces (22.5 points) performs much better in this study, beating random forests-1 (21 points), boosting (18 points), and random trees B (17.5 points). It is worth noting that all scores are above 17 which means they are each better than growing a bagged ensemble on average. Clearly, utilizing statistical significance tests changes the conclusions that one would make given these experimental results.

To complete our investigation of the performance of these ensemble creation techniques we list the average accuracy results over ten folds and provide a Borda count. The Borda count [18] is calculated by assigning a rank to each of the proposed methods (first place, second place, etc.). The first place method obtains N points, second place takes $N - 1$ points, and so on, where N is the number of methods compared. The sum of those values across all data sets is the Borda count, as shown in Table 2, where greater values are better.

Again we see random forests-lg and random forests-2 taking the lead, with Borda counts of 167 and 166, respectively. Random trees B and random subspaces obtain the next best scores of 152 and 150. It is difficult to say how many points constitutes a "significant" win, however boosting (128), bagging (118), and random forests-1 (117) certainly have a non-trivial number of points less.

Table 2. The average raw accuracy results. Boldface indicates statistically significantly better than bagging at the 99% confidence interval and italics means significantly worse. Results of a Borda count are provided with higher values signifying better performance. Summary scores are also reported.

Data set	Boosting	Random Subspaces	Random Trees B	Bagging	Random Forests-lg	Random Forests-1	Random Forests-2
anneal	99.33	99.78	99.78	99.22	99.33	99.67	99.78
audiology	79.57	81.74	78.26	80.43	81.74	76.09	78.26
autos	87.76	89.74	86.79	86.76	85.86	81.38	84.83
breast-w	96.85	96.99	96.28	95.99	96.85	97.14	96.99
breast-y	68.60	73.17	71.05	74.53	73.14	74.23	74.18
credit-a	86.23	86.81	83.48	86.09	86.82	86.96	86.96
credit-g	75.10	76.70	74.20	74.60	76.90	73.70	75.90
glass	77.27	77.73	79.55	74.55	75.91	78.18	79.09
heart-c	81.53	83.17	81.83	77.85	82.80	83.80	83.45
heart-h	79.24	82.62	79.92	79.92	80.60	81.29	80.93
heart-s	90.26	93.53	89.43	90.26	91.93	92.69	91.93
heart-v	71.00	75.00	72.00	72.50	77.50	77.50	76.50
hepatitis	83.79	82.46	86.37	83.83	85.67	85.04	85.04
horse-colic	80.96	83.96	84.50	85.86	85.86	84.23	85.59
hypo	98.74	98.80	98.99	99.15	99.02	98.92	98.99
ion	95.28	93.89	93.61	93.61	93.61	93.89	93.89
iris	94.67	94.00	94.67	94.67	94.67	94.00	94.00
krkp	99.56	*95.75*	*98.72*	99.66	*99.47*	*97.94*	*99.13*
labor	85.67	84.00	89.00	84.00	89.33	94.33	94.33
led-24	*71.43*	*69.44*	72.41	73.57	**74.93**	74.27	**74.77**
letter	**96.74**	**97.03**	**96.44**	94.90	**96.84**	**95.66**	**96.81**
lymph	82.67	80.00	86.67	78.67	84.00	84.00	85.33
page	96.39	97.21	97.34	97.19	97.32	97.21	97.39
pendigits	**99.21**	**99.30**	**99.25**	98.59	**99.25**	**99.02**	**99.14**
phoneme	91.46	*83.70*	90.37	91.42	91.26	91.02	91.35
pima	74.29	74.55	76.49	76.75	76.75	75.45	75.97
primary	35.73	45.75	44.57	40.74	45.16	44.56	46.35
sat	91.89	92.19	92.24	91.06	92.08	91.26	91.72
sick	98.91	*96.29*	98.86	98.94	*98.49*	*97.96*	*98.17*
sonar	81.43	82.38	85.24	77.14	81.90	82.38	83.81
soybean	91.36	95.31	92.38	92.53	94.14	94.00	93.41
vehicle	76.94	75.76	74.59	74.35	74.59	73.53	74.47
vote	95.23	95.45	94.77	95.91	95.91	95.00	96.14
waveform	84.21	**85.27**	**85.55**	84.01	**85.01**	**85.59**	**85.41**
Borda count	108	144	134	114	167	131	166
Summary							
Better	2	3	3		4	3	4
Worse	1	4	1		2	2	2
Same	31	27	30		28	29	28
Score	17.5	16.5	18		18	17.5	18

5 Discussion

5.1 Random Forests and Bagging

Since the random forest approach uses bagging to create the training sets for the trees of their ensembles, one might expect that the two algorithms share the same wins and losses while the other methods do not. This turns out not to be the case. On the krkp data set, random forests are statistically significantly less accurate than bagging, as are random subspaces and random trees. Likewise in the three cases where each version of random forests is statistically significantly more accurate than bagging, random subspaces and random trees are also more accurate; on two of those data sets boosting is more accurate. An interesting experiment would be to measure the accuracy of random forests without bagging the training set since this could lead to a decrease in the running time. Random forests are already much faster than bagging since fewer attributes need to be tested at every possible node in the tree.

5.2 Comparison against Prior Results in the Literature

Our accuracy results compare with those published by Breiman in [5] for both boosting and random forests. Of the 12 data sets common to each work, our implementation of random forests-1 is more accurate six times and our implementation of boosting is more accurate seven times than what is shown in [5]. The accuracy differences on data sets are small, possibly influenced by the use of two different splitting criteria functions (the gini index for CART and information gain ratio for OpenDT).

In Dietterich's Randomized C4.5 experiments, he chose to report the best of the pruned (C4.5 certainty factor of 10) and unpruned ensembles, arguing that the decision to prune might always be correctly determined by doing cross validation on the training set. Of the 13 data sets for which Dietterich chose to use unpruned trees and which appear in our paper, the OpenDT implementation was more accurate ten times. This difference is likely a result of the OpenDT implementation splitting randomly amongst the top twenty best attributes, rather than among attribute tests.

In [19], Ho carried out extensive comparison experiments between random subspaces and bagging on two class data sets that have more than 500 examples. The classifier utilized was an oblique decision tree. The data sets were randomly split in half 10 times and 10 experiments building 100 trees for the ensemble were done to get different accuracy values for statistical comparison purposes. Ho characterized the data sets for which random subspaces was statistically significantly better and the data sets for which bagging was better using a set of metrics. The average ratio of examples to features was about 93 for the data sets for which random subspaces outperform bagging. For the data sets in which we found significant differences, the minimum ratio of examples to features was 88 and all others were above 125. However, for some of these data sets bagging was better than any of the other approaches. Evaluating these data sets on the

rest of the metrics used would be interesting future work (with the caveat that some of these data sets are multi-class).

Another result of note from this work is that significant differences between ensemble classifiers were only found for data sets with more than 3000 examples. Previously [10], it was found that building an ensemble of classifiers from disjoint subsets of the data could be as accurate or more accurate than bagging for large data sets. One might consider that classifiers would be more diverse [20] if built from disjoint subsets of data than built with bagging if the data was dense. A diverse set of classifiers can, sometimes, have a beneficial effect on accuracy.

5.3 Other Ensemble Methods vs. Bagging

Given previous results [7,5] where random forests and random subspaces were better than boosting on data sets without noise and often better than bagging, we expected one or more approach might be very often better in a very rigorous statistical test. This was not the case (even for 95% where RF-1 does somewhat better but there are no changes in other rankings vs. bagging). Of the 34 data sets examined, the maximum statistically significant wins was 4 (with 2 losses). While the Borda count shows that, given several data sets, techniques such as random forests can show an accuracy improvement over bagging, for any particular data set, this accuracy improvement is not reliable.

There are other potential benefits aside from increased accuracy performance though. Random forests, by picking only a small number of attributes to test, generates trees very rapidly. Random subspaces, which also tests fewer attributes, can also use much less memory because only the chosen percentage of attributes needs to be stored. Recall that since random forests may potentially split on any attribute, it must store all the data. Since random trees do not need to make and store new training sets, they save a small amount of time and memory over the other methods.

Boosting has the unique ability to specialize itself on the hard to learn examples. Unfortunately this makes the algorithm highly susceptible to noise. As several of the data sets used here are known to be noisy, boosting is at a disadvantage in these experiments. If led-24, a synthetic data set with artificial noise, is removed from the experiment, boosting would have zero losses at the 99% confidence interval.

Finally, random trees and random forests can only be directly used to create ensembles of decision trees. As with bagging, both boosting and random subspaces can be utilized with other learning algorithms such as neural networks.

6 Summary

This paper compares several methods of building ensembles of decision trees. A variant of the randomized C4.5 method introduced by Dietterich [6] (which we call random trees B), random subspaces [7], random forests [5], Adaboost.M1W,

and bagging are compared. All experiments used a 10-fold cross validation approach to compare average accuracy. The accuracy of the various ensemble building approaches was compared with bagging using OpenDT to build unpruned trees. The comparison was done on 34 data sets with 32 taken from the UC Irvine repository [11] and the others publicly available.

The ensemble size was 1000 trees for each of the ensemble creation techniques except boosting which used 50. The ensemble size of boosting was chosen to match what had been used in previous work [5]. The ensemble size of the remaining techniques was chosen to enable ensembles to reach maximum accuracy. Statistical significance tests in conjunction with ANOVA were done to determine whether each of the ensemble methods was statistically significantly more than or less accurate than bagging.

No approach was unambiguously always more accurate than bagging. Random forests generally have better performance, and are much faster to build. The accuracy of the random subspace approach fluctuated, showing mediocre results statistically, but fairly good results generally. It is notable that a random forest built utilizing only two randomly chosen attributes for each test in the decision tree was among the most accurate classification methods.

Acknowledgments

This research was partially supported by the Department of Energy through the ASCI Views Data Discovery Program, Contract number: DE-AC04-76DO00789 and the National Science Foundation under grant EIA-0130768.

References

1. L. Breiman. Bagging predictors. *Machine Learning,* 24:123–140, 1996.
2. G. Eibl and K.P. Pfeiffer. How to make AdaBoost.M1 work for weak base classifiers by changing only one line of the code. In *Proceedings of the Thirteenth European Conference on Machine Learning,* pages 72–83, 2002.
3. M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory,* pages 158–169, 2000.
4. R.E. Schapire. The strength of weak learnability. *Machine Learning,* 5(2): 197–227, 1990.
5. L. Breiman. Random forests. *Machine Learning,* 45(1):5–32, 2001.
6. T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning,* 40(2): 139–157, 2000.
7. T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(8):832–844, 1998.
8. G. Hulten and P. Domingos. Learning from infinite data in finite time. In *Advances in Neural Information Processing Systems 14,* pages 673–680, Cambridge, MA, 2002. MIT Press.

9. K.W. Bowyer, N.V. Chawla, Jr. T.E. Moore, L.O. Hall, and W.P. Kegelmeyer. A parallel decision tree builder for mining very large visualization datasets. In *IEEE Systems, Man, and Cybernetics Conference,* pages 1888–1893, 2000.
10. N.V. Chawla, T.E. Moore, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, and C. Springer. Distributed learning with bagging-like performance. *Pattern Recognition Letters,* 24:455–471, 2003.
11. C.J. Merz and P.M. Murphy. *UCI Repository of Machine Learning Databases.* Univ. of CA., Dept. of CIS, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.
12. Robert Banfield. The OpenDT project. Technical report, University of South Florida, http://www.csee.usf.edu/~rbanfiel/OpenDT.html, 2003.
13. J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992. San Mateo, CA.
14. L.O. Hall, K.W. Bowyer, R.E. Banfield, D. Bhadoria, W.P. Kegelmeyer, and Steven Eschrich. Comparing pure parallel ensemble creation techniques against bagging. In *The Third IEEE International Conference on Data Mining,* pages 533–536, 2003.
15. P. Brazdil and J.Gama. The statlog project- evaluation / characterization of classification algorithms. Technical report, The STATLOG Project- Evaluation / Characterization of Classification Algorithms, http://www.ncc.up.pt/liacc/ML/statlog/, 1998.
16. L. Breiman, J.H. Friedman, R.A. Olshen, and P.J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA., 1984.
17. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations.* Morgan Kaufmann, San Francisco, 1999.
18. J. C. de Borda. *Memoire sur les elections au scrutin.* Historie de lÁcademie Royale des Sciences, Paris, 1781.
19. T.K. Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis and Applications,* 5:102–112, 2002.
20. R.E. Banfield, L.O. Hall, K.W. Bowyer, and W. P. Kegelmeyer. A new ensemble diversity measure applied to thinning ensembles. In *Multiple Classifier Systems Conference,* pages 306–316, June 2003.

Multiple Classifiers System for Reducing Influences of Atypical Observations

Šarūnas Raudys and Masakazu Iwamura

Vilnius Gediminas Technical University, Saulėtekio 11, Vilnius, Lithuania
Tohoku University, Aoba 05, Aramaki, Aoba, Sendai, 980-8579 Japan
raudys@ktl.mii.lt, masa@aso.ecei.tohoku.ac.jp

Abstract. Atypical observations, which are called outliers, are one of difficulties to apply standard Gaussian density based pattern classification methods. Large number of outliers makes distribution densities of input features multimodal. The problem becomes especially challenging in high-dimensional feature space. To tackle atypical observations, we propose multiple classifiers systems (MCSs) whose base classifiers have different representations of the original feature by transformations. This enables to deal with outliers in different ways. As the base classifier, we employ the integrated approach of statistical and neural networks. This consists of data whitening and training of single layer perceptron (SLP). Data whitening makes marginal distributions close to unimodal, and SLP is robust to outliers. Various kinds of combination strategies of the base classifiers achieved reduction of generalization error in comparison with the benchmark method, the regularized discriminant analysis (RDA).

1 Introduction

In many real-world practical pattern recognition tasks including printed and handwritten character recognition, we often meet atypical observations, and also meet the classification problem of such observations with Gaussian classifiers. Outliers are the observations which follow another distribution. If the number of outliers is large, the distributions could be multimodal ones. Applying Gaussian model to multimodal distribution produces many outliers.

To deal with multimodal distributions, nonparametric (local) pattern recognition methods such as *k*-NN rule and Parzen window classifier could be used because they approach the Bayes classifier with large training samples. However, in high-dimensional and small sample cases, *sample size/complexity ratio* becomes low. In such situations, utilization of nonparametric methods is problematic [1-3].

To reduce influences of atypical observations, we suggest multiple classifier systems (MCSs) whose several base classifiers have different representations of the original feature. We perform different transformations of the original feature (including no transformation) in order to deal with outliers in different ways.

As the base classifier, we employ the integrated approach of statistical and neural networks. This approach is the combination of data whitening and training of single layer perceptron (SLP) to recognize patterns. In data whitening, we also perform data rotation to achieve good start, speed up the SLP training, and obtain new features

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 233–242, 2004.

whose marginal distribution densities are close to unimodal ones and often resemble Gaussian distribution. In one base classifier, for data rotation we utilized robust estimates of mean vectors and pooled covariance matrix. The SLP based classifier is inherently robust to outliers.

We considered various kinds of combination strategies of the base classifiers including linear and non-linear fusion rules. We compare their performances with the regularized discriminant analysis (RDA) [2-4] as a benchmark method. RDA is one of the most powerful statistical pattern classification methods.

To test our theoretical suggestions, we considered important task of recognition of handwritten Japanese characters. In handwritten Japanese character recognition, some of the classes can be easily discriminated. However, there are many very similar classes, and recognition of such similar classes is important but difficult problem. To improve this situation, our concern is to study most ambiguous pairs of pattern classes. For illustration, eight pairs of similar Japanese characters are shown in Fig. 1.

Fig. 1. Eight pairs of similar Japanese characters.

2 Sample Size/Complexity Properties

The standard Fisher discriminant function (we call "discriminant function" DF in short) is one of the most popular decision rules. Let $\bar{\boldsymbol{x}}^{(1)}$ and $\bar{\boldsymbol{x}}^{(2)}$ be sample mean vectors, and **S** be pooled sample covariance matrix. Allocation of a *p*-variate vector $\boldsymbol{x}=(x_1,\cdots,x_p)^T$ is performed according to sign of DF [2]

$$g(\boldsymbol{x})=\left(\boldsymbol{x}-\frac{1}{2}\left(\bar{\boldsymbol{x}}^{(1)}+\bar{\boldsymbol{x}}^{(2)}\right)\right)^T \mathbf{S}^{-1}\left(\bar{\boldsymbol{x}}^{(1)}-\bar{\boldsymbol{x}}^{(2)}\right). \tag{1}$$

Let *N* be the number of training sets which are used to obtain estimates $\bar{\boldsymbol{x}}^{(1)}$, $\bar{\boldsymbol{x}}^{(2)}$ and **S**. Asymptotic classification error of sample based DF is given as $P_{\mathrm{B}}=\Phi(-\frac{1}{2}\delta)$, where δ stands for Mahalanobis distance and $\Phi(\cdot)$ is cumulative distribution function of N(0,1) (see, e.g., [2]). As both sample size *N* and dimensionality *p* increase, distribution of sample based DF approaches Gaussian law. After calculation of conditional means and common variance of discriminant function (1), one can find expected probability of misclassification,

$$\mathrm{EP}_N=\Phi\left(-\frac{1}{2}\delta\left(\left(1+\frac{2p}{N\delta^2}\right)\frac{2N}{2N-p}\right)^{-1/2}\right) \tag{2}$$

[3, 5]. In Eq. (2), term 2N/(2N–*p*) arises due to inexact estimation of covariance matrix, and term $1+2p/N\delta^2$ arises due to inexact estimation of mean vectors. Equation (1) will be Euclidean distance classifier (EDC) if covariance matrix **S** is ignored. EDC has relatively good small sample properties. Similarly, if covariance matrix **S** is de-

scribed by small number of parameters, DF with better small sample properties could be obtained. An example is the first order decision tree model described in Sect. 4 (see, e.g., [3, 6]). An alternative way to improve small sample properties is RDA. Covariance matrix of RDA is given as $\mathbf{S}_{\mathrm{RDA}} = (1-\lambda)\mathbf{S} + \lambda\mathbf{I}$, where λ is positive constant defined in an interval [0 1]. Optimal value of λ, denoted by λ_{opt}, have to be chosen by taking into account the balance between complexity of pattern recognition task (structure of the true covariance matrix $\boldsymbol{\Sigma}$) and sample size N.

In our investigations, sample size N=100 and the original dimensionality p=196. Suppose Bayes error $P_{\mathrm{B}} = 0.1$ ($\delta = 2.56$). Then $1 + 2p/N\delta^2 \approx 1.6$ and $2N/(2N - p) \approx 800$. High values of these coefficients indicate that we work in serious deficit of training data. One way to improve the data deficit problem is to reduce dimensionality, i.e. perform feature selection. Another way is to use simpler estimate of covariance matrix. We will use both of them. They are described in Sect. 3 and 4.

3 Representations of Feature Vectors

3.1 The Original Feature Vector

In this paper, 196-dimensional directional element feature [7] was used to represent handwritten Japanese characters in database ETL9B [8]. Preliminary to extracting the feature vector, a character image was normalized nonlinearly [9] to fit in a 64×64 box. Then, skeleton were extracted, and line segments of vertical, horizontal and slanted at ±45 degrees were extracted. An image is divided into 49 sub-areas of 16×16 dots. Sum of each segment in a region is an element of feature vector.

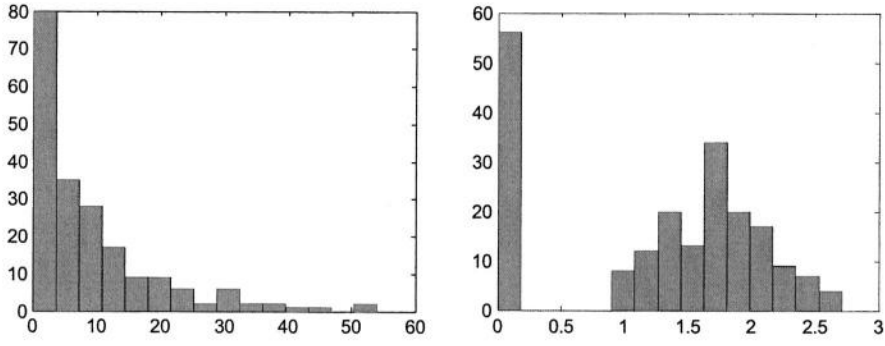
3.2 Three Representations of Feature Vector

In constructing three base classifiers for multiple classifier system, we performed three kinds of transformations of the feature vector:

A) original (without transformation),
B) transformed by $y_r = x_r^{1/s}$ for r-th element of the feature (r= 1, ..., 196, s is arbitrary) and
C) binarized (0 or 1) (non-zero valued components of the feature vector were equalized to 1).

We comment the reasons of using feature B and C. In Fig. 2a, we have a histogram of an element of the feature vector, which corresponds to the sub-area at a boundary of an image. The distribution density is highly asymmetric. It is well known that estimation of covariance matrix requires Gaussian density [10], and nonlinear transformations such as transformation (B) often helps reveal correlation structure of the data better. Thus, the histogram of nonlinearly transformed by $y_r = x_r^{1/4}$ is performed (Fig. 2b). We notice that the distribution of single feature is obviously bimodal and one peak is at zero. One possible way to tackle bimodality problem is to

ignore "outliers" (in our case "zero valued features"). As mentioned in Sect. 3.3, our dimensionality reduction strategy has similar effect to this for feature B. However, the deletion may cause loss of information. Therefore, the third expert classifier utilized binary vectors.



(a) Original measurement. (b) After transformation by $y_r = x_r^{1/4}$.

Fig. 2. Histograms of distribution of 5th feature, x_5.

3.3 Dimensionality Reduction

When the number of training vectors is unlimitedly large, local pattern recognition algorithms (such as Parzen window classifier and *k*-NN rule) could lead to minimal (Bayes) classification error if properly used. Unfortunately, in practice, the number of training vectors is limited, and the dimensionality of feature vector is usually high. Therefore, one needs utilize prior information available to build the classification rule. An optimal balance between complexity and training sample size has to be retained. If sample size is not very large, one has to restrict complexity of base classifiers. In small sample case, optimistically biased resubstitution error estimates of the base classifiers supplied to fusion rule designer could ruin performance of MCS [11].

When we have notably smaller coefficient in Eq. (2), i.e. for dimensionality $p^* = 20$, $1 + 2p^*/N\delta^2 \approx 1.06$. In order to improve small sample properties of base classifiers, for each kind of features transformation, we selected only twenty "better" features. Selection was performed on bases of sample Mahalanobis distances of each original feature. Since sample size was relatively small, we could not use complex feature selection strategy. Here, the written character occupies only a part of 7×7 area. Like in many of similar character recognition problem, some of the sub-areas are almost "empty". Thus, our feature selection strategy is: at first, the *r*-th elements of 196-dimensional feature vectors were divided by their standard deviation of each class s_r $(r = 1, 2, \ldots, 196)$; then, twenty features (dimensions) whose 1-dimensional sample means of two classes are more distant were selected. The experiments showed that this feature selection also had important secondary effect: many non-informative features which contain a large number of outliers (zero valued measurements) were discarded.

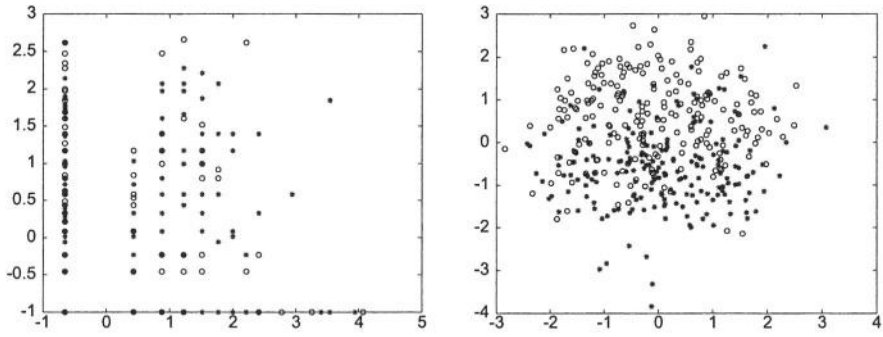
4 Three Base Classifiers

Three base classifiers were built in the three feature spaces (A, B, C) respectively. To construct robust base classifiers, we utilized the integrated approach of statistical and neural network [3]. This approach consists of data whitening and training of SLP. This can offer better linear DF by taking advantage of both statistical methods and neural networks.

In the data whitening, *one moves data mean vectors to the origin of coordinates,* and then performs *data whitening transformation* by use of $\boldsymbol{y} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{\Phi}^T\left(\boldsymbol{x} - 1/2\left(\bar{\boldsymbol{x}}^{(1)} + \bar{\boldsymbol{x}}^{(2)}\right)\right)$, where $\boldsymbol{\Lambda}$ and $\boldsymbol{\Phi}$ are the matrices of eigenvalues and eigenvectors of simplified covariance matrix $\mathbf{S}_{\text{RDA\&Tree1}}$. We used regularization and the first order tree dependence model [6] for the simplified covariance matrix $\mathbf{S}_{\text{RDA\&Tree1}} = \mathbf{S}_{\text{Tree1}}\left(1-\lambda_{\text{opt}}\right) + \lambda_{\text{opt}}\mathbf{I}$, where $\mathbf{S}_{\text{Tree1}}$ is covariance matrix of the first order tree dependence model described only by $2p$-1 independent parameters. This simplification makes the estimate of the covariance matrix more reliable in small sample case. Thus, in data whitening, we perform data rotation by means of orthogonal matrix $\boldsymbol{\Phi}$ and variance normalization by multiplying rotated data by matrix $\boldsymbol{\Lambda}^{-1/2}$. This transformation has a secondary effect which have not been discussed in the robust statistics literature. Linear transformation of multidimensional data produces weighted sums of the original features (see Fig. 3). For this reason, the distribution densities of the new features are closer to univariate and unimodal, and often resemble Gaussian distribution. In our experiments we noticed that time and again in whitened feature space, the first components give good separation of the data.

After data whitening, SLP was trained in space of $\boldsymbol{y}$. The training of SLP started with zero valued weight vector. After the first batch iteration, we obtain DF (1) whose $\mathbf{S}$ is replaced by $\mathbf{S}_{\text{RDA\&Tree1}}$. If assumptions about structure of covariance matrix are truthful, estimate $\mathbf{S}_{\text{RDA\&Tree1}}$ helps to have quite good DF with relatively small error rate and good small sample properties just at the very beginning.

If starting regularization parameter λ_{opt} is suitable, proper stopping could help to obtain the classifier of optimal complexity. To determine optimal number of batch iterations (epochs) to train SLP, we utilized independent pseudo-validation data sets with colored noise injection [12]. The pseudo-validation sets were formed by adding many (say n) randomly generated zero mean vectors to each training pattern vector. The detail is as follows. For each vector $\boldsymbol{x}_i$, its k nearest neighbors $\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \cdots, \boldsymbol{x}_{ik}$ are found in the same pattern class; then, k lines which connect $\boldsymbol{x}_i$ and $\boldsymbol{x}_{iq}$, (q=1, 2,..., k) are prepared; along the q-th line, one adds random variables which follow Gaussian distribution $\mathrm{N}\left(0, \left(\sigma \mid \boldsymbol{x}_i - \boldsymbol{x}_{iq} \mid\right)^2\right)$; after adding k components, a new artificial vector is obtained. This procedure is repeated n times. Three parameters have to be defined to realize a noise injection procedure. In our experiments, we used: k = 2, n = 10, and σ = 1. In fact, noise injection introduces *additional non-formal information*: it declares in an inexplicit way that the space between nearest vectors of one pattern class could be filled with vectors of the same category (for more details see [11, 12]).



(a) Original feature space. (b) Whitened feature space.

Fig. 3. Effect of whitening transformation; "*" and "o" stand for feature vectors of two classes respectively, which were transformed by $y_r = x_r^{1/2}$.

Second expert, B, was working in transformed by $y_r = x_r^{1/3}$ feature space where bimodality of the data was clearly visible. For robust estimation, an influence of outliers was reduced purposefully. We ignored measurements with zero feature values for *robust estimation of mean vectors and covariance matrix.* While estimating the mean values and variance of *j*-th feature, we rejected zero valued training observations. To estimate ρ_{ij}, a correlation coefficient between *i*-th and *j*-th elements of feature vector, we utilized training vectors with only nonzero *i*-th and *j*-th components.

5 Experiments with Handwritten Japanese Characters

5.1 Fusion Rules

We utilize a number of different fusion rules and compare classification performances of MCSs with RDA used as a benchmark method. From an abundance of known fusion rules (see, e.g., [13]), eight linear and non-linear rules below were considered to make final decision.

BestT) The best (single) base classifier is selected according to classification results using the test set. Actually, this is the ideal classifier which achieves the minimum error rates in use of the three base classifiers. This classifier and BestV (the next item) are weighted voting MCSs that only one weight is unequal to zero.

BestV) The best (single) base classifier is selected according to classification results using the pseudo-validation set. This classifier was used as a benchmark MCS.

MajV) Majority voting. This is a fixed (non-trainable) fusion rule.

WStv) Weighed sum of the outputs of the base classifiers. SLP was used as fusion rule. The original training data set was used to train SLP classifier and produce coefficients of weighted sum. Optimal stopping was performed according to classification error estimated from pseudo-validation set.

BKS) The original behavior knowledge space method (see, e.g., [3, 14]). Allocation is performed according to probabilities $P_{11},\ldots,P_{18}$, $P_{21},\ldots,P_{28}$ of eight combinations of binary outputs of three base classifiers; training set were used to estimate the probabilities mentioned.

BKSn) Modified BKS method aimed to reduce expert adaptation to training data [11]. An independent pseudo-validation set was used to estimate $P_{11},\ldots,P_{28}$.

G&R) Nonlinear classification in 3D space of the three outputs of the base classifiers. To make final allocation, the Parzen window classifier was utilized (this MCS utilizes of expert outputs. Its decision making procedure resembles that of Giacinto and Roli [15], therefore, it is marked by G&R).

R&E) Nonlinear fusion of the outputs of the base classifiers where a sample-based oracle uses the input vector $\boldsymbol{x}$ in order to decide which expert is the best competent to classify this particular vector $\boldsymbol{x}$. The fusion rule allocates vector $\boldsymbol{x}$ to one of three virtual pattern classes (experts). The competence of the j-th expert is estimated as a "potential"

$$\hat{p}_j(\boldsymbol{x}) = \sum_{s=1}^{K}\sum_{l=1}^{N_s} q_{jl}^{s} \exp\left\{-\frac{(\boldsymbol{x}-\boldsymbol{x}_l^{(s)})^T(\boldsymbol{x}-\boldsymbol{x}_l^{(s)})}{h^2}\right\}, \quad (j=1,2,3), \tag{3}$$

where $q_{jl}^{s}=1$ if l-th training vector of s-th class, $\boldsymbol{x}_l^{(s)}$, was classified by the j-th expert correctly, and $q_{jl}^{s}=-1$ if vector $\boldsymbol{x}_l^{(s)}$ was classified incorrectly, $\exp\{\cdot\}$ is a *kernel* and h is a smoothing constant. This approach corresponds to Rastrigin and Erenstein [16] fusion rule introduced three decades ago. We marked it by R&E.

RDA) RDA is one of the most powerful pattern classification tools, and was described in Sect. 4.

5.2 Experiment

Each pair of handwritten character contains two similar classes. Each class consists of 200 vectors. 100 vectors were randomly selected as the training set (N=100), and remaining 100 vectors were the test set. To reduce an influence of randomness, the experiments were performed 100 times for each pair of characters. Every time, random permutation of vectors was performed in each category.

For each data representation, individual feature selection and subsequent data rotation and normalization procedures were performed. After few preliminary experiments following parameters were determined: $p^{*}=20$, $\lambda_{\text{opt}}=0.2$, and $s=2$.

In all experiments, only training set and its "product", artificial pseudo-validation set described in Sect. 4, were used to design decision making rules. The SLPs which were used as the base classifiers were trained on training set. Optimal number of iterations was determined by the recognition results of pseudo-validation set. While building some of trainable fusion rules, we interchanged training and pseudo-validations sets: the fusion rules were trained on validation set, and optimal number of iterations was found according to error rates of the training set. The test set was used only once, for final evaluation of generalization errors.

Results obtained in 800 training sessions (100 independent experiments with eight pairs of similar handwritten Japanese characters) are summarized in Table 1. For every pair, averaged test error rates of three experts (1st E, 2nd E and 3rd E), BestT and BestV are presented in the left five columns of Table 1 (next to the index of character pair). Let $\bar{\hat{P}}_{\text{BestV}}$ be the averaged test error rate of BestV (printed in bold in the table) and $P_{\text{Method\_A}}$ be that of "Method A". Further, the ratio of the averaged test error rate of "Method A" (corresponding to remaining 6 fusion rules and RDA) to that of BestV are shown in the right seven columns of Table 1. Namely, the relative error rate is given as $P_{\text{Method\_A}} \Big/ \bar{\hat{P}}_{\text{BestV}}$. The very last row in the table contains averaged values of eight cells of the column.

Table 1. Average error rates of single experts, BestT and BestV are in the left five columns next to index, and relative efficacy of six fusion procedures and RDA are in the seven columns in the right. Relative error rates of the most effective fusion rules are underlined.

PAIR	1st E	2nd E	3rd E	BestT	BestV	MVot	WSum	BKS	BKSn	G&R	R&E	RDA
A	0.037	0.037	0.042	0.035	**0.042**	0.882	0.899	1.022	0.930	1.239	0.911	1.349
B	0.129	0.116	0.198	0.112	**0.122**	0.946	1.012	1.039	0.946	1.090	1.008	1.385
C	0.056	0.070	0.159	0.054	**0.066**	1.002	0.964	0.923	1.002	0.955	0.841	1.115
D	0.138	0.139	0.154	0.132	**0.144**	0.950	0.981	1.051	1.018	1.042	0.972	1.453
E	0.087	0.083	0.133	0.079	**0.103**	0.805	0.810	0.865	0.823	0.819	0.842	1.261
F	0.135	0.130	0.190	0.125	**0.138**	0.920	0.962	0.996	0.921	0.986	0.972	1.575
G	0.086	0.088	0.100	0.081	**0.091**	0.940	0.940	0.948	0.941	0.955	0.968	1.675
H	0.120	0.119	0.149	0.112	**0.125**	0.899	0.923	0.960	0.899	0.943	0.967	1.410
ALL	0.098	0.098	0.141	0.091	**0.104**	0.918	0.936	0.976	0.935	1.004	0.935	1.403

In spite of apparent similarity of eight kinds of Japanese character pairs, we have notable variations in experimental results obtained for diverse pairs: both separability of pattern classes (classification error rate) and relative efficacy of the experts differ by pairs.

Nevertheless, for all the pairs, RDA whose parameter λ is adjusted to complexity of the recognition problem and size of training set was outperformed by MCSs designed to deal with outliers and multimodality problems. By comparing RDA and MCS rules, the highest gain among MCS rules (1.675 times in comparison with BestV, and 1.78 times in comparison with Majority Voting) was obtained for pair **G** where all three experts were approximately equally qualified. The lowest gain (1.115 times in comparison with BestV, and 1.325 times in comparison with Rastrigin-Erenstein procedure) was obtained for pair **C** where the third expert was notably worse than two others.

The training set size of the current problem, 100+100 vectors in 196-variate, is rather small. Therefore, sophisticated trainable fusion rules were ineffective: for almost all eight Japanese character pairs considered, the fixed fusion rule, Majority Voting, was the best. Exception is pair **C** because of inefficiency of the third expert, that is, only two experts participated in final decision making. Detailed analysis shows that in general, all three experts are useful: rejection of one of them assists an increase in generalization error of MCS.

6 Concluding Remarks

In this paper, we considered problem of atypical observations in training set in high-dimensional situations where sample size is relatively small. Since Gaussian classifiers are not suitable for atypical observations, as a practical solution, we proposed multiple classifiers systems (MCSs) whose base classifiers have different data representations respectively. The base classifiers are constructed with the integrated approach of statistical and neural networks.

To test the proposed MCSs, we considered recognition task of similar pairs of handwritten Japanese characters. For all eight similar Japanese character pairs considered, all the proposed MCSs outperformed the benchmark classification method, the RDA, in the situation of small sample and high-dimensional problem. Utilization of MCSs with base classifiers working in differently transformed feature space contains supplementary information that nonlinear transformations are important in revealing atypical observations. Dealing with the outlier problem, dissimilarity of features allowed the MCSs to reduce generalization error.

With a simple feature selection procedure, all three base classifiers worked in reduced feature spaces. The feature selection procedure utilizes additional information: a part of the features are notably less important for the linear classification rules designed to operate with unimodal distributions. Analysis of histograms of rejected features showed that often rejected features had bimodal distribution density functions, i.e. substantial part of data contained zero-valued measurements. This means that our feature selection lightened outliers and multimodality problem.

Training sample size used to train the experts and the trainable fusion rules of MCSs is too small for given high-dimensional pattern recognition problem. Therefore, fixed fusion rule performed the best. No doubt that in situations with larger number of samples, more sophisticated fusion rules would be preferable and could lead higher gain in dealing with outlier and multimodality problems.

Acknowledgments

The authors thank Assoc. Prof. Shinichiro Omachi, Prof. Hirotomo Aso, Dr. Ausra Saudargiene and Giedrius Misiukas for useful discussions, shared with us data sets and Matlab codes.

References

1. S. Raudys. and A.K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Trans, on Pattern Analysis and Machine Intelligence,* PAMI-13:252–64, 1991.
2. R.O. Duda, P.E. Hart and D.G. Stork. *Pattern Classification.* 2nd ed. Wiley, NY, 2000.
3. S. Raudys. *Statistical and Neural Classifiers: An integrated approach to design.* Springer-Verlag, NY, p. 312, 2001.

4. J.M. Friedman. Regularized discriminant analysis. *Journal ofAmerican Statistical Association* 84:165–75, 1989.
5. S. Raudys. On the amount of a priori information in designing the classification algorithm. Proc. Ac. Sci. USSR, Ser. *Engineering Cybernetics,* N4:168–74, 1972 (in Russian).
6. S. Raudys and A. Saudargiene. Tree type dependency model and sample size - dimensionality properties. *IEEE Trans, on Pattern Analysis and Machine Intelligence* 23(2):233-239, 2001.
7. N. Sun, Y. Uchiyama, H. Ichimura, H. Aso and M. Kimura. Intelligent recognition of characters using associative matching technique. *Pacific Rim Int'l Conf. Artificial Intelligence (PRICAI'90),* 546-551,1990.
8. T. Saito, H. Yamada and K. Yamamoto. On the data base ETL9 of handprinted characters in JIS Chinese characters and its analysis. *Trans. IEICE,* J68-D(4):757-764, 1985 (in Japanese).
9. H. Yamada, K. Yamamoto and T. Saito. A nonlinear normalization method for handprinted Kanji character recognition: line density equalization. *Pattern Recognition,* 23(9): 1023-1029, 1990.
10. H. Ujiie, S. Omachi, and H. Aso. A Discriminant Function Considering Normality Improvement of the Distribution. *16th International Conference on Pattern Recognition (ICPR 2002),* 2:224-227, 2002.
11. S. Raudys. Experts' boasting in trainable fusion rules. *IEEE Trans. on Pattern Analysis and Machine Intelligence.* PAMI 25(9):1178-1182, 2003.
12. M. Skurichina, Š. Raudys and R.P.W. Duin. K-nearest neighbors directed noise injection in multilayer perceptron training, *IEEE Trans. on Neural Networks,* 11(2):504–511, 2000.
13. A.F.R. Rahman and M.C. Fairhurst. Multiple classifier decision combination strategies for character recognition: A review. *Int. J.Document Analysis and Recognition,* 5(4):166–194, 2003.
14. Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 17(1):90-94,1995.
15. G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition,* 34(9):1879-1881, 2001.
16. L.A. Rastrigin and R.Ch Erenstein. On decision making in a collective of decision rules. *Priborostroenie ,* LITMO, St-Petersburg. 16(11):31-35, 1973 (in Russian).

Sharing Training Patterns among Multiple Classifiers

Rozita Dara and Mohamed Kamel

Pattern Analysis and Machine Intelligence Laboratory
University of Waterloo, Waterloo, Ont.
Canada, N2L-3G1
{mkamel,rdara}@uwaterloo.ca

Abstract. Demand for solving complex problems has directed the research trend in intelligent systems toward design of cooperative multi-experts. One way of achieving effective cooperation is through *sharing* resources such as information and components. In this paper, we study classifier combination techniques from cooperation perspective. The degree and method by which multiple classifier systems *share* training resources can be a measure of cooperation. Even though data modification techniques, such as bagging and k-fold crossvalidation, have been extensively used, there is no guidance whether sharing or not sharing training patterns results in higher accuracy and under what conditions. We carried out a set of experiments to examine the effect of sharing training patterns on several architectures by varying the size of overlap between 0-100% of the size of training subsets. The overall conclusion is that sharing training patterns among classifiers is beneficial.

1 Introduction

Combination of multiple classifiers has been studied and compared from different perspectives. These categorizations are based on different design techniques ([1], [5], [9]), type of aggregation strategies ([6], [7], [13]), and topology of different architectures [11]. However, none of these studies have considered combination methods from *cooperation* viewpoint. Although Sharkey [13] differentiates between cooperative and competitive approaches, this distinction refers to the way in which outputs of base classifiers are combined to make the final decision. In this study, we are interested in the issue of cooperation from a different perspective. The degree and method by which multiple classifier systems (MCS) share the resources can be a measure of cooperation. A clear picture of the behaviour of MCS may emerge by identification of the components/resources and analysis of the gains and drawbacks of sharing these resources.

Sharing can be studied at four fundamental levels of MCS: training, feature representation, classifier architectures, and decision making. The basic level of sharing occurs at the training level. Sharing at this level may be examined by distinguishing three types of resources: training patterns, training algorithms, and training strategies. The key investigation, in this study, is how cooperation

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 243–252, 2004.

among classifiers through sharing training patterns affects system performance. More training data may result in higher accuracy for individual classifiers. However, many popular techniques in MCS have been designed based on partitioning of the training data. There are several reasons behind these approaches: *i)* to achieve diversity among classifiers ([8], [14], [15]), *ii)* to be able to work with large datasets [4], *iii)* to obtain classifiers that are experts on specific parts of the data space.

Bagging, boosting, and k-fold crossvalidation are the most well-known data modification approaches. In *Bagging* [3], at each run, each classifier is presented with a training set that is of the same size as the original training data. These training sets consist of samples of training examples that are selected randomly with replacement from the original training set. Each classifier is trained with one of these training sets and the final classification decision is made by taking the majority vote over the class predictions produced by these classifiers. Boosting, proposed by Freund and Schapire [12], is a technique for combining weak classifiers. In this technique, classifiers and training sets are obtained in a more deterministic way, in comparison to bagging. At each step of boosting, weights are assigned to data patterns in such a way that higher weight is given to the training examples that are misclassified by the classifiers. At the final step, outputs of classifiers are combined using weighted majority method for each presented pattern. In k-fold-crossvalidation, the training set is randomly divided into k subsets. Then, k-*1* of these sets are used to train a classifier. Subsequently, the resulted classifier is tested on the subset that has been left out of the training. By changing the left out subset in the training process, k classifiers are constructed.

Despite the growing number of publications on different data modification techniques and their usage, it is not clear what the advantages and disadvantages of sharing training patterns are, and which combining architectures make use of sharing data and which ones do not. In this work, we examine the effect of sharing patterns by developing simple partitioning schemes for the training data similar to Chawla and co-workers [4]. Since training in MCS is affected by different factors including type of architectures, we also studied the effects of sharing training data on several architectures. Throughout the experiments, for each different partitioning technique, we evaluated performance of the system using disjoint, different sizes of overlaps, and identical training sets. We were interested in finding a relationship between the overlap size and performance of several MCS architectures.

2 Sharing Training Patterns

Let $c = \{c_1, c_2, ..., c_l\}$ and $r = \{r_1, r_2, ..., r_k\}$ be sets of system components and resources respectively. Let f represent a mapping between a component and a resource (a resource used by a component). Then, S_{r^*} is the total number of components $c_i \in c$ that share a resource $r^* \in r$.

$$S_{r^*} = \bigcup_{i=1}^{l} f(c_i, r^*) \tag{1}$$

As a result, the total amount of system resources $r_j \in r$ shared by components $c_i \in c$ can be denoted by

$$S = \bigcup_{j=1}^{k} \{ \bigcup_{i=1}^{l} f(c_i, r_j) \}. \tag{2}$$

In this paper, we investigate the effect of sharing training patterns among classifiers. Therefore, we consider components of the system to be the classifiers and resources to be the training patterns. We denote a set of training data by $X = (X_1, X_2, ...X_n)$, where each training pattern X_i is an m dimensional vector. The training data is randomly divided into disjoint and overlapped partitions. The modified training set consists of $X^N = (X_1^N, X_2^N, ..., X_p^N)$, where p, $p < n$, is the size of partition randomly selected from n patterns and N represents the number of partitions. The partitioning methods used in this study are as follows:

Disjoint Partitions (DP): Training data is randomly partitioned into N disjoint partitions of size $\frac{1}{N}$th of the original data. The union of the N training set is identical to the original data (Figure 1).

Disjoint Partitions with Replications (DPR): For each disjoint partition, elements of each partition are independently selected in a random fashion and added to the partition until the size of each partition is equal to the size of the original data (Figure 1). Importantly, there is no overlap between partitions.

Overlapping Partitions (OP): A certain percentage of patterns are randomly selected from the disjoint partitions. The union of these randomly selected patterns constitutes the overlapping set. The overlapping set is added to each of the N partitions independently (Figure 1).

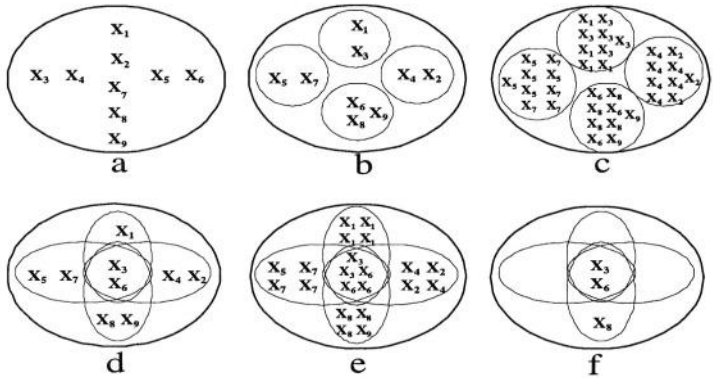


Fig. 1. Partitioning Techniques, (a) Original Data, (b) DP, (c) DPR, (d) OP, (e) OPR, (f) FOP

Overlapping Partitions with Replications (OPR): For each overlapping partitioning set, elements of each partition are randomly replicated until the size of each partition is equal to the size of the original data (Figure 1). The size of overlap varies depending on the randomly selected elements that are added to each partition.

Fixed Overlapping Partitions (FOP): This method is similar to *OP* with the exception of substituting overlapping set with a set of randomly selected patterns from disjoint partitions (Figure 1). In this technique, the union of partition is not identical to the original data.

Task-Oriented Partitions (TOP): Training data is partitioned into k or less disjoint partitions, where k is the number of classes. Each class or a number of classes belongs to one partition. This method is only used for training modular architectures.

Algorithm 1 The Partitioning Algorithm

1: Partition each class, or several classes, to disjoint sets (TOP).
2: Randomly partition the original training data into disjoint subsets (DP).
3: Randomly replicate patterns of each disjoint set (DPR).
4: For each overlap size, make an overlapping set by randomly selecting patterns from the disjoint partitions, generated in the step 2.
5: Add the overlapping sets to the disjoint sets (OP).
6: Randomly replicate patterns of the partitions generated in step 5 (OPR).
7: Substitute overlapping sets in each of the disjoint sets by randomly eliminating some of the patterns from the disjoint sets (FOP).

The training data partitions were prepared based on the following generation scheme (Algorithm 1). Each one of these partitioning schemes was developed for a different objective: *i)* DP, DPR and TOP to obtain disjoint partitions, *ii)* DPR and OPR to obtain partitions equal to the size of original training data, and *iii)* OP, OPR, and FOP to achieve partitions with overlap. Because of lack of studies from sharing point of view, it was not clear which method (shared or disjoint data representation) yields to a better generalization performance.

3 Description of Architectures

The significance of sharing patterns may depend on the architectures of MCS. In order to draw reliable conclusions, we implemented four architectures modular, ensemble, stacked generalization, and hierarchical mixtures of experts, and examined the effect of partitioning techniques and overlap size on each one of them. Description of architectures is as follows:

- *Ensemble:* classifiers were trained in parallel using the subsets obtained by DP, DPR, OP, OPR and FOP partitioning methods. The final decision was made using Majority Vote and Weighted Averaging aggregation methods.

- *Modular:* complete decoupling [1] and top-down competitive [13] architectures were implemented. For each of these architectures, Task-Oriented Partitioning (TOP) method was used in which each class or multiple disjoint classes were partitioned into one training set. By adding overlapping set (OP) to these disjoint sets, the effect of sharing was evaluated. In top-down competitive method, control switching to an appropriate module was done using a backpropagation classifier.
- *Stacked Generalization:* classifiers were arranged in layers [16]. Similar to ensemble architecture, DP, DPR, OP, OPR and FOP methods were used for partitioning. The final decision was made using a backpropagation network located at the higher level.
- *Hierarchical Mixtures of Experts (HME)*: classifiers were arranged in layers [10]. DP, DPR, OP, OPR and FOP methods were used for obtain disjoint and overlapped training subsets. Backpropagation classifiers were used at the gating level.

4 Experimental Setup

We used two benchmark datasets for this study. The *80-D Correlated Gaussian* is an 80 dimensional artificial data [8]. It consists of two Gaussian classes with equal covariance matrices and 500 vectors for each class. The second dataset, *20-class Gaussian,* is a two dimensional artificial dataset that contains 20 classes [2]. Each class has a total number of 100 patterns.

We studied different sizes of overlaps. The size of overlapping set varied between 0 to 100 percent of the size of partitions, which resulted in 0-50% of shared data patterns among classifiers. This ratio can be calculated using Eq. 2, as $s_p = \frac{S_p}{p+S_p}$, where p is the size of disjoint partitions and S_p is the size of overlapping set. We were, also, interested to study the influence of the sample size on MCS in the presence of shared and disjoint training data. We considered three sample sizes of small, medium and large for each of the training sets. The selection of sizes was based on the datasets dimensionality. We considered $p \leq m$ for *small* size, $p \geq m$ for *medium* size, and $p \gg m$ for *large* size where m is the data dimension.

We used Linear and Quadratic classifiers as base classifiers in our experiments. The use of stable classifier enabled us to obtain consistent results by eliminating drastic fluctuation in the system performance caused by classifiers themselves. The number of classifiers was varied from 6 to 9. Given the size of the datasets, creating partitions of more than 10 resulted in insufficient training data and poor performances for the classifiers.

Previous to training data partitioning, datasets were divided into training and testing subsets. Subsequently, the training set was partitioned to smaller subsets (Algorithm 1). The partitioning process and combination process were repeated between 20-30 times with different random seeds. The choice of repartitioning the training data was dictated by the need of having consistent results and eliminating the influence of random selection. We calculated the mean and

standard deviation of MCS performance to estimate the amount of changes in their performances.

5 Results and Discussion

We examined a set of plots and compared the performance of different partitioning methods. We observed a common pattern of change among three partitioning methods. Figure 2 illustrates error rate for 80-D Gaussian dataset using majority vote aggregation method (ensemble architecture). Adding larger overlaps in the FOP method always resulted in decline of generalization capability of the system. Unlike DP and DPR methods that resulted in either improvement or no change in the system. The difference between FOP method and other techniques is that the union of the partitions generated by FOP is not the same as the original training data. This means that by adding larger overlaps to the partitions, many of the patterns are eliminated and not used in the training of MCS. It made a little or no difference whether the partitions were created by sampling with or without replications. Results for DP and OP are highlighted for the rest of the paper.

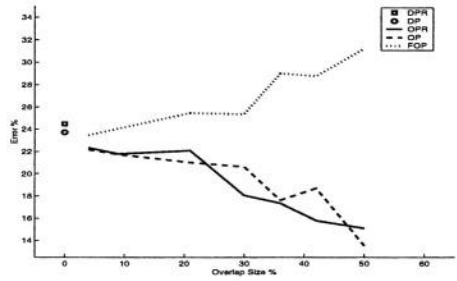


Fig. 2. Performance of Different Partitioning Techniques

Figures 3-14 compare the MCS error for different architectures and training sample sizes: small, medium and large. In addition, for each of the training sizes, these figures illustrate the changes in MCS error with respect to the size of overlap among partitions. The straight line, in these figures, represents performance of bagging on the *original* training set. By examining the results, the overall conclusion was that sharing training data was useful, especially for small and medium training sizes. In the case of 20 class dataset (Figure 3 and 4), combining classifiers on disjoint and small overlap sizes of large training set outperformed bagging. However, sharing patterns for small training set was not as effective as medium size. This is likely due to the fact that the size of partitions for small training size was too small for classifiers to be able to learn well. Another observation was that sharing training data for the high dimensional (complex) dataset such as 80-D Gaussian was more essential. The need for larger training sets can

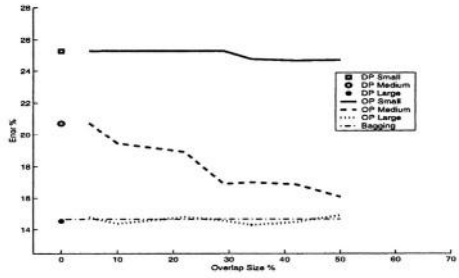


Fig. 3. 20 Class: Majority Vote

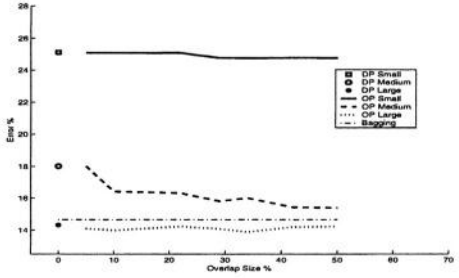


Fig. 4. 20 Class: Weighted Averaging

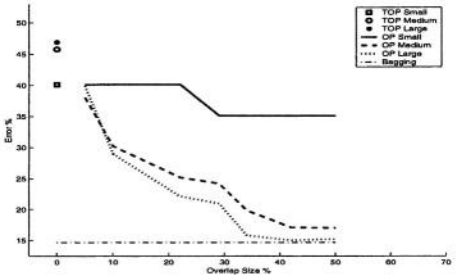


Fig. 5. 20 Class: Decoupled

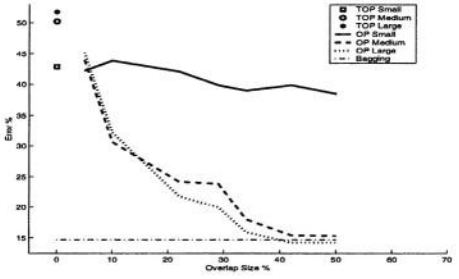


Fig. 6. 20-class: Top-down Competitive

be seen in Figures 9-14, in which bagging trained on the original training set outperformed all MCS that were trained on smaller sets.

Increasing the size of the sharing (overlaps) resulted in improvement of the performance for all architectures. This observation was examined for traditional combination rules (Figures 3, 4, 9, 10). For modular achitecture, classifiers constructed on TOP, disjoint partitions were biased toward their own classes (Figures 5, 6, 11, 12). Improvement in performance was likely due to the fact that presenting patterns from other classes might have helped the classifiers to gain a common view about the problem and, therefore, improve MCS performance. In the case of Stacked Generalization and HME architectures (Figures 7, 8, 13, 14), performances of the upper levels classifiers were highly dependent on the performance of classifiers at the lower levels. Classifiers trained on small datasets might have large variances, since parameters of classifiers were poorly estimated. Thus, input to the next level were not accurate, which resulted in incorrect estimation of the final decision.

Another observation was that, even though adding larger overlaps to disjoint subsets reduced diversity among the classifiers, most of the time resulted in better performance. Selecting disjoint subsets of patterns increased the chance of obtaining a group of independent classifiers. However, it might decline performance of the individual classifiers if too few patterns were used. Consequently, performance of combined less accurate classifiers would be low. These observations were consistent with Kuncheva *et al.* [8] findings in which they concluded

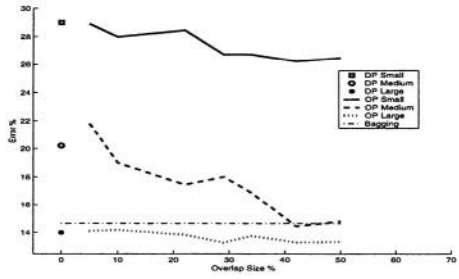


Fig. 7. 20 Class: Stacked Generalization

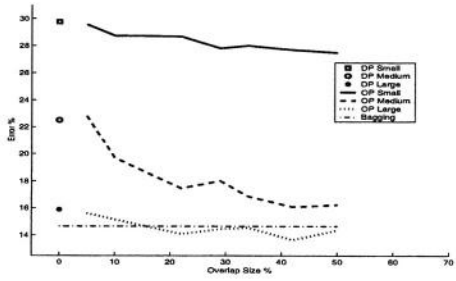


Fig. 8. 20 Class: HME

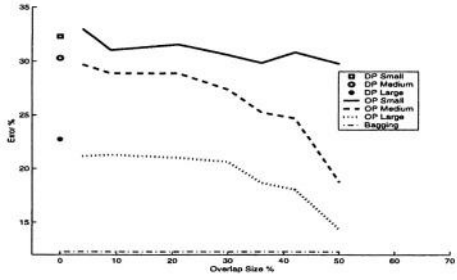


Fig. 9. 80-d Gaussian: Majority Vote

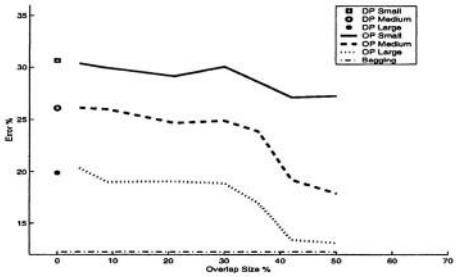


Fig. 10. 80-d Gaussian: Weighted Averaging

that there was no strong relationship between diversity and accuracy. As a result, the best construction strategy for the MCS is likely to be achieved if classifiers can meet the criteria of being diverse and be able to generalize well [14].

6 Conclusion

The performance of MCS is affected by many factors such as: the choice of base classifier, the training sample size, the way training data is partitioned, and the choice of architecture. Therefore, it is difficult to establish universal criteria to generalize the usefulness or drawbacks of sharing. In this paper, we empirically investigated correlation between sharing training data and MCS performance. We developed several partitioning techniques. The effect of sharing was examined considering these techniques with two datasets, several architectures, and training data sizes.

The overall observation from the results is that sharing is generally beneficial. Improvement over larger overlap subsets, for all the architectures, may be explained by the fact that applying disjoint partitions to classifiers, results in a set of biased classifiers towards their own training data partitions. Combining these biased classifiers may decline the performance of MCS. Sharing portions of training data is an alternative attempt to improve the performance. Shared

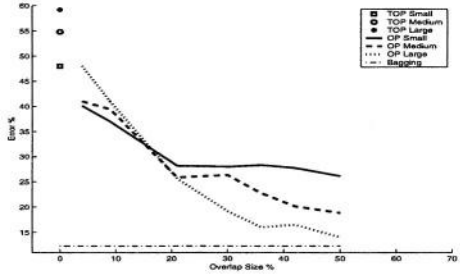


Fig. 11. 80-d Gaussian: Decoupled

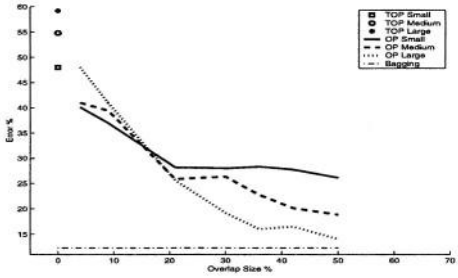


Fig. 12. 80-d Gaussian: Top-down Competitive

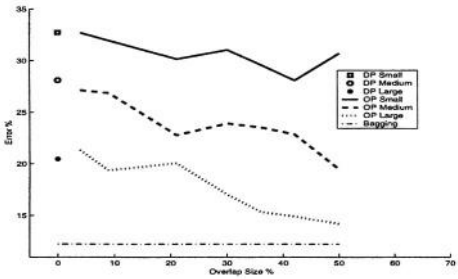


Fig. 13. 80-d Gaussian: Stacked Generalization

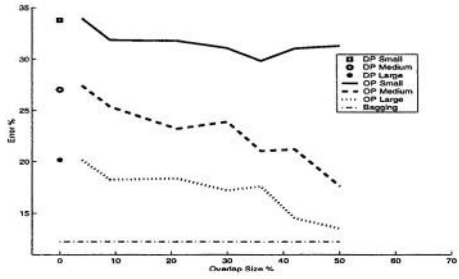


Fig. 14. 80-d Gaussian: HME

information replicated across the subsets of training data attempts to provide each individual classifier with a more accurate *view* of the problem in-hand.

This study suggests that if there is too little data or a complex problem in-hand, the gains achieved by replicating the patterns cannot compensate for the decrease in accuracy of individual classifiers. As a result, it is more advisable to use the whole data for training and try to obtain diversity through other methods. On the other hand, if the data is large enough, there is a sweet spot for the training data size for which diversity through data partitioning is most effective. Presented results even illustrated that, for less complex data with large enough training set, partitioning the data into disjoint sets may result in improvement in the accuracy of multiple classifiers.

References

1. Auda, G., Kamel, M.: Modular Neural Networks: A Survey. Int. Journal of Neural Systems, **9(2)** (1999) 129–151.
2. Auda, G., Kamel, M.: CMNN: Cooperative Modular Neural Networks. Neurocomputing, **20** (1999) 189-207.
3. Breiman, L., Bagging Predictors. Machine Learning, **24(2)** (1996) 123–140.

4. Chawla, N., Moore, T., Hall, L., Bowyer, K., Kegelmeyer, P. Springer, C.: Distributed Learning with Bagging-like Performance. Pattern Recognition Letters, 24 (2003) 455–471.
5. Dietterich, T.: Ensemble Methods in Machine Learning. First Int. Workshop on MCS, Cagliari Italy (2000) 1–15.
6. Duin, R.: The Combining Classifier: to Train or Not to Train? Proc. of 16th Int. Conf. on Pattern Recognition, Quebec City, CA, (2002) 765–770.
7. Kamel, M., Wanas, N.: Data Dependence in Combining Classifiers. Fourth Int. Workshop on MCS, Guilford UK (2003) 1–14.
8. Kuncheva, L., Skurichina, M., Duin, R.: An Experimental Study on Diversity for Bagging and Boosting with Linear Classifiers. Information Fusion, **3(4)**, (2002) 245–258.
9. Kuncheva, L., Bezdek, J., Duin, R.: Decision Templates for Multiple Classifier Fusion: An Experimental Comparison. Pattern Recognition, **34(2)** (2001) 299–314.
10. Jiang, W., Tanner, M.: Hierarchical Mixtures of Experts for Generalized Linear Models. Neural Computation **11** (1999) 1183–1198.
11. Lam, L.: Classifier Combinations: Implementations and Theoretical Issues. First Int. Workshop on MCS, Cagliari Italy (2000) 77–86.
12. Freund, Y., Schapire, R.: Experiments with a New Boosting Algorithm. Proc. of the 13th Int. Conf. on Machine Learning, Bari Italy (1996) 148–156.
13. Sharkey, A.: Types of Multinet system. Third Int. Workshop on MCS, Cagliari Italy (2002) 108–117.
14. Sharkey, A., Sharkey, N., Chandroth, G.: Diverse Neural Net Solutions to a Fault Diagnosis Problem. Neural Computing and Applications **4** (1996) 218–227.
15. Wang, W., Jones, P., Partridge, D.: Diversity Between Neural Networks and Decision Trees for Building Multiple Classifier Systems. First Int. Workshop on MCS, Cagliari Italy (2000) 240–249
16. Wolpert, D.: Stacked Generalization. Neural Networks, **5**, (1992) 241–259.

First Experiments on Ensembles of Radial Basis Functions*

Carlos Hernández-Espinosa, Mercedes Fernández-Redondo, and Joaquín Torres-Sospedra

Universidad Jaume I, Dept. de Ingeniería y Ciencia de los Computadores
Avda Vicente Sos Baynat s/n, 12071 Castellon, Spain
{redondo,espinosa}@icc.uji.es

Abstract. Building an ensemble of classifiers is an useful way to improve the performance with respect to a single classifier. In the case of neural networks the bibliography has centered on the use of Multilayer Feedforward. However, there are other interesting networks like Radial Basis Functions (RBF) that can be used as elements of the ensemble. Furthermore, as pointed out recently the network RBF can also be trained by gradient descent, so all the methods of constructing the ensemble designed for Multilayer Feedforward are also applicable to RBF. In this paper we present the results of using eleven methods to construct an ensemble of RBF networks. We have trained ensembles of a reduced number of networks (3 and 9) to keep the computational cost low. The results show that the best method is in general the *Simple Ensemble.*

1 Introduction

Probably the most important property of a neural network (NN) is the generalization capability, i.e., the ability to correctly respond to inputs which were not used in the training set.

One method to increase the generalization capability with respect to a single NN consist on training an ensemble of NNs, i.e., to train a set of NNs with different weight initialization or properties and combine the outputs of the different networks in a suitable manner to give a single output.

In the field of ensemble design, the two key factors to design an ensemble are how to train the individual networks to get uncorrelated errors and how to combine the different outputs of the networks to give a single output.

It seems clear from the bibliography that this procedure generally increases the generalization capability in the case of the NN Multilayer Feedforward [1,2].

However, in the field of NNs there are other interesting networks besides Multilayer Feedforward, and traditionally the use of ensembles of NNs has restricted to the use of Multilayer Feedforward as element of the ensemble.

Another interesting network which is quite used in applications is Radial Basis Functions (RBF). This network can also be trained in a fully supervised way by gradient descent as shown recently [3-4]. Furthermore the performance of this way of

* This research was supported by the project MAPACI TIC2002-02273 of CICYT in Spain.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 253–262, 2004.

training is even superior to the traditional unsupervised clustering for the centers and the supervised training for the weights of the outputs.

So with a fully supervised gradient descent training, this network can be also an element of an ensemble, and all methods of constructing the ensemble which are applicable to Multilayer Feedforward can now be also used with RBF networks.

In this paper we try different methods of constructing the RBF networks in the ensemble to obtain the first results on ensembles of RBF networks.

Among the methods of combining the outputs, the two most popular are *voting* and *output averaging* [5]. In this paper we will normally use *output averaging,* we have also performed experiments with *voting* and the results are completely similar in the case of RBF networks.

2 Theory

In this section, first we briefly review the basic concepts of RBF networks and gradient descent training and after that we review the different method of constructing the ensemble which are applied to the RBF networks. Full descriptions of both items can be found in the references.

2.1 RBF Networks with Gradient Descent Training

A RBF has two layer of networks (without considering the input units). The first layer is composed of neurons with a Gaussian transfer function and the second layer (the output units) has neurons with a linear transfer function. The output of a RBF network can be calculated with equation 1.

$$F_k(x) = \sum_{q=1}^{Q} w_q^k \cdot \exp\left(- \sum_{n=1}^{N} \left(C_{q,n}^k - X_n\right)^2 \Big/ \left(\sigma_q^k\right)^2 \right) \tag{1}$$

Where $C_{q,n}^{\ k}$ are the components of the center of the Gaussian functions, σ_q^k control the width of the Gaussian functions and $w_q^{\ k}$ are the weights among the Gaussian units and the output units.

In the case of RBF neural networks trained with gradient descent [3], a constant σ_q^k is used for all Gaussian units of the network, and this parameter is determined by trial and error, fixing a value before training, i.e., the width is not adaptively change during training.

The parameters which are changed during the training process are $C_{q,n}^{\ k}$ the centers of the Gaussian and $w_q^{\ k}$ the weights among the Gaussian units and the output units. In the original reference the training is performed off-line (i.e. in Batch mode) but we have concluded that the online version presented here has several advantages like a lower number of iterations to complete training. The equations for the adaptation of the weights is the following:

$$\Delta w_q^k = \eta \cdot \varepsilon_k \cdot \exp\left(- \sum_{n=1}^{N} \left(C_{q,n}^k - X_n\right)^2 \Big/ \left(\sigma\right)^2 \right) \tag{2}$$

Where η is the step size and ε_k is the difference between the target and the output of the network for the particular training pattern.

The equation 3 calculate the adaptation of the centers and the equation 4 $\varepsilon^h_{q,n}$.

$$\Delta C_q = \eta \cdot \varepsilon^h_{q,k} \cdot (X_k - C_q) \tag{3}$$

$$\varepsilon^h_{q,k} = \alpha_{q,k} \cdot \sum_{k=1}^{n_o} \varepsilon_k \cdot w^k_q \tag{4}$$

Where X_k is the kth training pattern and n_o is the number of output units and $\alpha_{q,k}$ is in the following equation.

$$\alpha_{q,k} = \frac{2}{\sigma} \cdot \exp\left(- \sum_{n=1}^{N} \left(C^k_{q,n} - X_n \right)^2 \Big/ (\sigma)^2 \right) \tag{5}$$

2.2 Ensemble Design Methods

Simple Ensemble: A simple ensemble can be constructed by training different networks with the same training set, but with different random initialization in the weights, center and weights in the case of RBF networks. In this ensemble technique, we expect that the networks will converge to different local minimum and the errors will be uncorrelated.

Bagging: This ensemble method is described in reference [5]. The ensemble method consists on generating different datasets drawn at random with replacement from the original training set. After that, we train the different networks in the ensemble with these different datasets (one network per dataset). We have used datasets which have a number of training points equal to twice the number of points of the original training set, as it is recommended in the reference [1].

Bagging with Noise (BagNoise): It was proposed in [2]. It is a modification of *Bagging,* we use in this case datasets of size 10·N (number of training points) generated in the same way of *Bagging,* where N is the number of training points of the initial training set. Also we introduce a random noise in every selected training point drawn from a normal distribution with a small variance.

Boosting: This ensemble method is reviewed in [5]. It is conceived for a ensemble of only three networks. It trains the three network of the ensemble with different training sets. The first network is trained with the whole training set, N input patterns. After this training, we pass all N patterns through the first network and we use a subset of them, such that the new training set has 50% of patterns incorrectly classified by the first network and 50% correctly classified. With this new training set we train the second network. After the second network is trained, the N original patterns are presented to both networks. If the two networks disagree in the classification, we add the

training pattern to the third training set. Otherwise we discard the pattern. With this third training set we train the third network.

In the original theoretical derivation of the algorithm, the evaluation of the test performance was as follows: present a test pattern to the three networks, if the first two networks agree, use this label, otherwise use the label assigned by the third network.

CVC: It is reviewed in [1]. In k-fold cross-validation, the training set is divided into k subsets. Then, k-1 subsets are used to train the network and results are tested on the subset that was left out. Similarly, by changing the subset that is left out of the training process, one can construct k classifiers, each of which is trained on a slightly different training set.

Adaboost: We have implemented the algorithm denominated *"Adaboost.M1"* in the reference [6]. In the algorithm the successive networks are trained with a training set selected at random from the original training set, but the probability of selecting a pattern changes depending on the correct classification of the pattern and on the performance of the last trained network. The algorithm is complex and the full description should be looked for in the reference. The method of combining the outputs of the networks is also particular to this algorithm.

Decorrelated (Deco): This ensemble method was proposed in [7]. It consists on introducing a penalty term added to the usual Backpropagation error function. The penalty term for network number *j* in the ensemble is in equation 6.

$$Penalty = \lambda \cdot d(i,j)(y - f_i)\cdot(y - f_j) \tag{6}$$

Where λ determines the strength of the penalty term and should be found by trial and error, y is the target of the training pattern and f_i and f_j are the outputs of networks number i and j in the ensemble. The term $d(i,j)$ is in equation 7.

$$d(i,j) = \begin{cases} 1, & if\ i = j-1 \\ 0, & otherwise \end{cases} \tag{7}$$

Decorrelated2 (Deco2): It was proposed also in reference [7]. It is basically the same method of *"Decorrelated"* but with a different term $d(i,j)$ in the penalty. In this case the expression of $d(i,j)$ is in equation 8.

$$d(i,j) = \begin{cases} 1, & if\ i = j-1 \quad and \quad i\ is\ even \\ 0, & otherwise \end{cases} \tag{8}$$

Evol: This ensemble method was proposed in [8]. In each iteration (presentation of a training pattern), it is calculated the output of the ensemble for the input pattern by voting. If the output is correctly classified we continue with the next iteration and pattern. Otherwise, the network with an erroneous output and lower MSE (Mean Squared Error) is trained in this pattern until the output of the network is correct. This procedure is repeated for several networks until the vote of the ensemble classifies correctly the pattern. For a full description of the method see the reference.

Cels: It was proposed in [9]. This method also uses a penalty term added to the usual Backpropagation error function to decorrelate the output of the networks in the ensemble. In this case the penalty term for network number i is in equation 9.

$$Penalty = \lambda \cdot (f_i - y) \cdot \sum_{j \neq i} (f_j - y) \tag{9}$$

Where y is the target of the input pattern and f_i and f_j the outputs of networks number i an j for this pattern.

The authors propose in the paper the winner-take-all procedure to combine the outputs of the individual networks, i. e., the highest output is the output of the ensemble. We have used this procedure and the usual output averaging.

Ola: This ensemble method was proposed in [10]. First, several datasets are generated by using bagging with a number of training patterns in each dataset equal to the original number of patterns of the training set. Every network is trained in one of this datasets and in *virtual data.* The *virtual data* for network i is generated by selecting randomly samples for the original training set and perturbing the sample with a random noise drawn from a normal distribution with small variance. The target for this new virtual sample is calculated by the output of the ensemble without network number i for this sample. For a full description of the procedure see the reference.

3 Experimental Results

We have applied the eleven ensemble methods to nine different classification problems. They are from the UCI repository of machine learning databases. Their names are Balance Scale (BALANCE), Cylinders Bands (BANDS), Liver Disorders (BUPA), Credit Approval (CREDIT), Glass Identification (GLASS), Heart Disease (HEART), the Monk's Problems (MONK' 1, MONK'2) and Voting Records (VOTE). The complete data and a full description can be found in the UCI repository (http://www.ics.uci.edu/~mlearn/MLRepository .html).

The general characteristic of these databases are resumed in Table 1. The second column with header "Ninput" is the number of inputs of the database. Column fifth "Noutput" is the number of output (classes) of the database. Column sixth "Ntrain" is the number of patterns included in the training set. Column number seven "Ncross" is the number of patterns in the cross-validation set and finally column number eight "Ntest" is the number of patterns to test the performance. Furthermore, in this table we have several characteristic of the RBF networks trained in the ensemble, column number three "Nclusters" is the number of Gaussian units in the RBF network and column number four contains the learning step of the gradient descent algorithm, these parameters were selected by trial and error and cross-validation to find the most appropriate values.

The first step to construct the ensembles was to determine the right parameters for each database, in the case of methods *Cels* (parameter lambda of the penalty), *Ola* (standard deviation of the noise), *Deco* and *Deco2* (parameter lambda of the penalty) and *BagNoise* (standard deviation of the noise). The value of the final parameters obtained by trial and error and cross-validation is in Table 2.

Table 1. General characteristics of the Databases and Networks.

Database	Ninput	Nclusters	Step η	Noutput	Ntrain	Ncross	Ntest
Balance	4	60	0.005	3	395	105	125
Band	39	40	0.01	2	177	45	55
Bupa	6	40	0.01	2	220	55	70
Credit	15	30	0.005	2	418	105	130
Glass	10	110	0.01	6	129	35	50
Hear	13	20	0.005	2	190	48	59
Monk1	6	30	0.005	2	282	70	80
Monk2	6	45	0.005	2	282	70	80
Vote	16	5	0.01	2	285	70	80

Table 2. Parameters of different ensemble methods.

	Cels		Ola		Deco	Deco2	Bag
	3	9	3	9			Noise
Balance	0.75	0.25	0.4	0.6	0.6	0.6	0.1
Band	0.1	0.5	0.3	0.3	0.6	0.6	0.1
Bupa	0.1	0.1	0.6	0.6	0.6	0.6	0.1
Credito	0.25	0.25	0.6	0.5	0.6	0.6	0.4
Glas	0.25	0.25	0.3	0.3	0.6	0.6	0.1
Heart	0.1	0.1	0.4	0.3	0.6	0.6	0.2
Mok1	0.1	0.1	0.2	0.6	0.6	0.6	0.1
Mok2	0.1	0.25	0.6	0.6	0.8	0.6	0.1
Vote	0.1	0.1	0.6	0.6	0.6	0.6	0.1

With these parameters we trained the ensembles of 3 and 9 networks, we selected a low number of networks to keep the computational cost low. We repeated this process of training an ensemble ten times for ten different partitions of data in training, cross-validation and test sets. In this way, we can obtain a mean performance of the ensemble for each database (the mean of the ten ensembles) and an error in the performance calculated by standard error theory. The results of the performance are in Table 3 for the case of ensembles of three networks and in Table 4 for the case of nine. We have included also in Table 3 the mean performance of a single network for comparison.

As commented before, we have performed experiments with *voting* and *output averaging* as combination methods, the results are similar and we have reproduced here the results for *output averaging,* except for the case of *Cels* which is clearly better the winner-take-all strategy and for the case of *Evol* where *voting* is better that *output averaging.*

By comparing the results of Tables 3 and 4 with the results of a single network we can see that the improvement of the ensemble is database and method dependent. Sometimes, the general performance of an ensemble (as in case of Balance) is worse than the single network, the reason may be the combination method *(output averaging)* which does not exploit the performance of the individual networks. Besides that, there is one method which clearly perform worse than the single network which is *Evol,* but we obtained the same result for ensembles of Multilayer Feedforward networks.

To see the results more clearly, we have also calculated the percentage of error reduction of the ensemble with respect to a single network. We have used equation 10 for this calculation.

Table 3. Results for the ensemble of three networks.

	BALANCE	BAND	BUPA	CREDIT	GLAS
Single Net.	90.2 ± 0.5	74.0 ± 1.1	70.1 ± 1.1	86.0 ± 0.8	93.0 ± 0.6
Adaboost	88.0 ± 1.2	73 ± 2	68.4 ± 0.5	84.7 ± 0.6	91.3 ± 1.3
Bagging	89.7 ± 0.8	73 ± 2	70.1 ± 1.6	87.4 ± 05.	93.8 ± 1.2
Bag_Noise	89.8 ± 0.8	73.1 ± 1.3	64 ± 2	87.1 ± 0.7	92.2 ± 0.9
Boosting	88.2 ± 0.8	70.7 ±1.8	70.6 ± 1.6	86.6 ± 0.7	92.2 ± 0.9
Cels	89.5 ± 0.8	75.3 ± 1.4	69.3 ± 1.4	86.9 ± 0.5	93.0 ± 1.0
CVC	90 ± 0.7	75.1 ± 1.4	69.9 ± 1.5	87.5 ± 0.5	92.4 ± 1.1
Decorrelated	89.8 ± 0.8	73.3 ± 1.7	71.9 ± 1.6	87.1 ± 0.5	93.2 ± 1.0
Decorrelated2	89.8 ± 0.8	73.8 ± 1.4	71.3 ± 1.4	87.2 ± 0.5	93.4 ± 1.0
Evol	89.5 ± 0.9	67.6 ± 1.3	63 ± 2	84.6 ± 1.1	88 ± 2
Ola	88.2 ± 0.9	73.1 ± 1.3	68.1 ± 1.5	86.1 ± 0.7	89.4 ± 1.8
Simple Ense	89.7 ± 0.7	73.8 ± 1.2	71.9 ± 1.1	87.1 ± 0.5	93.2 ± 1.0

Table 3 (continuation). Results for the ensemble of three networks.

	HEART	MOK1	MOK2	VOTE
Single Net.	82.0 ± 1.0	98.5 ± 0.5	91.3 ± 0.7	95.4 ± 0.5
Adaboost	82.0 ± 1.4	88.0 ± 1.7	84.9 ± 1.5	95.1 ± 0.8
Bagging	83.6 ± 1.8	99.1 ± 0.6	88.0 ± 1.4	95.8 ± 0.6
Bag_Noise	83.2 ± 1.5	94.3 ± 1.3	89.5 ± 1.1	95.6 ± 0.7
Boosting	82.4 ± 1.2	94.9 ± 1.4	88.4 ± 1.0	95.9 ± 0.6
Cels_	82.7 ± 1.7	94.3 ± 1.3	89.5 ± 1.3	94.9 ± 0.6
CVC	83.9 ± 1.6	95.5 ± 1.3	84.9 ± 1.7	95.9 ± 0.7
Decorrelated	84.1 ± 1.4	99.5 ± 0.4	89.8 ±1.3	96.0 ± 0.7
Decorrelated2	83.3 ± 1.4	99.4 ± 0.4	91.9 ± 1.2	96.1 ± 0.6
Evol	79 ± 2	74.8 ± 1.4	64.6 ± 1.6	92.3 ± 0.7
Ola	81.5 ± 1.2	75.1 ± 1.1	83.1 ± 1.1	96.1 ± 04.
Simple Ense	84.6 ± 1.5	99.6 ± 0.4	90.9 ± 1.1	96.4 ± 0.6

Table 4. Results for the ensemble of nine networks.

	BALANCE	BAND	BUPA	CREDIT	GLAS
Single Net.	90.2 ± 0.5	74.0 ± 1.1	70.1 ± 1.1	86.0 ± 0.8	93.0 ± 0.6
Adaboost	91.8 ± 0.8	71.5 ± 1.2	69.6 ± 1.1	84.8 ± 0.8	93.1 ± 1.3
Bagging	90 ± 0.8	74.3 ± 1.4	71.0 ± 1.5	87.5 ± 0.5	93.6 ± 1.2
Bag_Noise	90 ± 0.8	73.1 ± 1.1	64.1 ± 1.9	87.1 ± 0.5	91.4 ± 0.9
Cels	89.7 ± 0.8	74.0 ± 1.2	69.4 ± 1.9	87.1 ± 0.5	92.4 ± 1.1
CVC	89.8 ± 0.8	73.6 ± 1.3	70 ± 2	87.6 ± 0.5	93.0 ± 1.1
Decorrelated	89.8 ± 0.8	73.5 ± 1.8	71.4 ± 1.4	87.2 ± 0.6	93.0 ± 1.0
Decorrelated2	89.8 ± 0.8	73.8 ± 1.8	71.6 ± 1.2	87.2 ± 0.5	93.2 ± 1.0
Evol	88.1 ± 1.1	67.6 ± 1.3	63 ± 2	83.4 ± 1.3	82.6 ± 1.8
Ola	88.5 ± 0.7	74.5 ± 1.2	69.6 ± 1.4	81.8 ± 1.1	90.6 ± 1.2
Simple Ense	89.7 ± 0.7	73.3 ± 1.4	72.4 ± 1.2	87.2 ± 0.5	93.0 ± 1.0

Table 4 (continuation). Results for the ensemble of nine networks.

	HEART	MOK1	MOK2	VOTE
Single Net.	82.0 ± 1.0	98.5 ± 0.5	91.3 ± 0.7	95.4 ± 0.5
Adaboost	81.0 ± 1.5	91.1 ± 1.4	87.0 ± 1.2	95.9 ± 0.6
Bagging	84.9 ± 1.2	99.4 ± 0.4	89.3 ± 1.2	95.9 ± 0.6
Bag_Noise	83.6 ± 1.6	95.5 ± 1.4	90.1 ± 1.2	96.0 ± 0.7
Cels	82.5 ± 1.4	93.3 ± 0.7	87.5 ± 1.2	94.9 ± 0.6
CVC	84.1 ± 1.3	99.4 ± 0.5	90.5 ± 1.0	96.1 ± 0.7
Decorrelated	84.1 ± 1.3	99.4 ± 0.4	90.3 ± 1.1	96.1 ± 0.7
Decorrelated2	84.7 ± 1.4	99.6 ± 0.4	91.1 ± 1.2	96.1 ± 0.6
Evol	78 ± 2	74.8 ± 1.4	64.6 ± 1.6	92.3 ± 0.7
Ola	78 ± 2	71.1 ± 1.6	82.0 ± 1.4	96.1 ± 0.4
Simple Ense	83.9 ± 1.5	99.6 ± 0.4	91.4 ± 1.2	96.3 ± 0.6

$$PorError_{reduction} = 100 \cdot \frac{PorError_{single\ network} - PorError_{ensemble}}{PorError_{single\ network}} \quad (10)$$

In this last equation, $PorError_{single\ network}$ is the error percentage of a single network (for example, 100-90.2=9.8% in the case of database Balance, see Table 3) and $PorError_{ensemble}$ is the error percentage in the ensemble with a particular method (for example, 100-88.0=12.0% in the case of Adaboost and database Balance, see Table 3).

The value of the percentage of error reduction ranges from 0%, where there is no improvement by the use of a particular ensemble method with respect to a single network, to 100% where the error of the ensemble is 0%. There can also be negative values, which means that the performance of the ensemble is worse than the performance of the single network.

This new measurement is relative and can be used to compare more clearly the different methods. In Table 5 we have the results for each database and method in the case of the ensemble of nine networks.

Furthermore we have calculated the mean performance across all databases of the percentage of error reduction and is in the last column with header "Mean".

According to this mean measurement there are five methods which perform worse than the *Single Network,* they are *Adaboost, BagNoise, Cels, Evol* and *Ola.* The performance of *Evol* is clear, in all databases is worse than the single network. *Ola, Adaboost, BagNoise* and *Cels* are in general problem dependent (unstable), for example *Adaboost* gets an improvement in databases Balance and Vote, but in the rest the performance is poor.

Table 5. Percentage of error reduction for the ensemble of 9 networks.

	BALANCE	BAND	BUPA	CREDIT	GLAS
Adaboost	16.33	-9.62	-1.67	-8.57	1.43
Bagging	-2.04	1.15	3.01	10.71	8.57
Bag_Noise	-2.04	-3.46	-20.07	7.86	-22.86
Cels	-5.10	0	-2.34	7.86	-8.57
CVC	-4.08	-1.54	-0.33	11.43	0
Decorrelated	-4.08	-1.92	4.35	8.57	0
Decorrelated2	-4.08	-0.77	5.02	8.57	2.86
Evol	-21.42	-24.62	-23.75	-18.57	-148.57
Ola	-17.35	1.92	-1.67	-30	-34.29
Simple Ense	-5.10	-2.69	7.69	8.57	0

Table 5 (continuation). Percentage of error reduction for the ensemble of 9 networks.

	HEART	MOK1	MOK2	VOTE	MEAN
Adaboost	-5.56	-493.33	-49.43	10.87	-59-95
Bagging	16.11	60	-22.99	10.87	9.49
Bag_Noise	8.89	-200	-13.79	13.04	-25.82
Cels	2.78	-346.67	-43.68	-10.87	-45-18
CVC	11.67	60	-9.20	15.22	9.24
Decorrelated	11.67	60	-11.49	15.22	9.14
Decorrelated2	15	73.33	-2.30	15.22	12.53
Evol	-22.22	-1580	-306.90	-67.39	-245.94
Ola	-22.22	-1826.67	-106.90	15.22	-224.66
Simple Ense	10.56	73.33	1.15	19.57	12.56

The best and most regular method across all databases seems to be the *Simple Ensemble.*

Finally, to see the influence of the number of networks in the ensemble we present the results of the mean percentage of error reduction for ensembles of three and nine networks in Table 6. We can see from the results that in general there is an improvement in performance from the ensemble of three to the ensemble of nine networks. Perhaps, the unique notable exception is the performance of the *Simple Ensemble* which is roughly the same.

Table 6. Mean percentage of error reduction for ensembles of 3 and 9 networks.

	Ensemble of 3 networks	Ensemble of 9 networks
Adaboost	-93.96	-59-95
Bagging	3.57	9.49
Bag_Noise	-35.69	-25.82
Boosting	-33.20	---
Cels	-34.01	-45-18
CVC	-27.60	9.24
Decorrelated	9.34	9.14
Decorrelated2	11.14	12.53
Evol	-234.21	-245.94
Ola	-191.45	-224.66
Simple Ense	12.96	12.56

4 Conclusions

In this paper we have presented results of eleven different methods to construct an ensemble of RBF networks, using nine different databases. We trained ensembles of a reduced number of networks, in particular three and nine networks to keep the computational cost low. The results showed that in general the performance is method and problem dependent, sometimes the performance of the ensemble is even worse than the single network, the reason can be that the combination method (output averaging) is not appropriate in the case of RBF networks. The best and most regular performing method across all databases seems to be the *Simple Ensemble,* i.e., the rest of methods proposed to increase the performance of Multilayer Feedforward seems not to be so useful in RBF networks. Perhaps, another reason of this result in the combination method (output averaging) which may not be very appropriate as commented before, the future research will go in the direction of trying other combination methods with ensembles of RBF networks.

References

1. Tumer, K., Ghosh, J., "Error correlation and error reduction in ensemble classifiers", Connection Science, vol. 8, nos. 3 & 4, pp. 385-404, 1996.
2. Raviv, Y., Intrator, N., "Bootstrapping with Noise: An Effective Regularization Technique", Connection Science, vol. 8, no. 3 & 4, pp. 355-372,1996.
3. Karayiannis, N.B., "Reformulated Radial Basis Neural Networks Trained by Gradient Descent", IEEE Trans. On Neural Networks, Vol. 10, no. 3, pp. 657- 671, 1999.

4. Karayiannis, N.B., Randolph-Gips, M.M., "On the Construction and Training of Reformulated Radial Basis Function Neural Networks", IEEE Trans. On Neural Networks, Vol.14, no. 4, pp. 835-846, 2003.
5. Drucker, H., Cortes, C, Jackel, D., et alt., "Boosting and Other Ensemble Methods", Neural Computation, vol. 6, pp. 1289-1301, 1994.
6. Freund, .Y., Schapire, R., "Experiments with a New Boosting Algorithm", Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148-156, 1996.
7. Rosen, B., "Ensemble Learning Using Decorrelated Neural Networks", Connection Science, vol. 8, no. 3 & 4, pp. 373-383, 1996.
8. Auda, G., Kamel, M., "EVOL: Ensembles Voting On-Line", Proc. of the World Congress on Computational Intelligence, pp. 1356-1360,1998.
9. Liu, Y., Yao, X., "A Cooperative Ensemble Learning System", Proc. of the World Congress on Computational Intelligence, pp. 2202-2207, 1998.
10. Jang, M., Cho, S., "Ensemble Learning Using Observational Learning Theory", Proceedings of the International Joint Conference, on Neural Networks, vol. 2, pp. 1281-1286, 1999

Random Aggregated and Bagged Ensembles of SVMs: An Empirical Bias–Variance Analysis

Giorgio Valentini

DSI - Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, Via Comelico 39, Milano, Italy
valentini@dsi.unimi.it

Abstract. Bagging can be interpreted as an approximation of random aggregating, an ideal ensemble method by which base learners are trained using data sets randomly drawn according to an unknown probability distribution. An approximate realization of random aggregating can be obtained through subsampled bagging, when large training sets are available. In this paper we perform an experimental bias–variance analysis of bagged and random aggregated ensembles of Support Vector Machines, in order to quantitatively evaluate their theoretical variance reduction properties. Experimental results with small samples show that random aggregating, implemented through subsampled bagging, reduces the variance component of the error by about 90%, while bagging, as expected, achieves a lower reduction. Bias–variance analysis explains also why ensemble methods based on subsampling techniques can be successfully applied to large data mining problems.

1 Introduction

Random aggregating is a process by which base learners, trained on samples drawn accordingly to an unknown probability distribution from the entire universe population, are aggregated through majority voting (classification) or averaging between them (regression).

Random aggregating is only a theoretical ensemble method. When large data sets are available, *subsampled bagging* can simulate random aggregating, using an uniform probability distribution and resampling techniques, assuming that the large available data set and the uniform probability distribution are good approximations respectively of the universe population and of the unknown probability distribution.

Breiman showed that in regression problems, random aggregation of predictors always improves the performance of single predictors, while in classification problems this is not always the case, if poor base predictors are used [1]. The improvement depends on the stability of the base learner: random aggregating and bagging are effective with unstable learning algorithms, that is when small changes in the training set can result in large changes in the predictions of the base learners [3].

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 263–272, 2004.

Breiman showed also that bootstrap aggregating *(bagging)* techniques improve the accuracy of a single predictor reducing mainly the variance component of the error [1,2].

Recently bias–variance decomposition of the error has been applied as a tool to study the behaviour of learning algorithms and to develop new ensemble methods well-suited to the bias–variance characteristics of the base learners [4]. In this paper we extend to bagged and random aggregated ensembles of Support Vector Machines (SVMs) the bias–variance analysis performed on single SVMs [5].

The main purpose of this contribution consists in quantitatively evaluating the variance reduction properties of both random aggregating and bagging, through an extended experimental bias–variance decomposition of the error. In this way we can quantitatively verify the theoretical results obtained by Breiman about the variance reduction properties of random aggregating, and we can also understand the extents to which Breiman's results obtained for random aggregating can be extended to bagging.

The experimental results suggest that subsampled bagged ensembles of SVMs could be successfully applied to very large scale data mining problems, as they significantly reduce the variance component of the error with respect to a single Support Vector Machine. In order to verify this hypothesis, we performed some preliminary experiments on a large synthetic data set, comparing the accuracy and the computational time of single SVMs trained on the entire large training set with ensembles of SVMs trained on small subsamples of the available data.

2 Random Aggregating, Subsampled and Classical Bagging

There are close relationships between random aggregating, subsampled and classical bagging: bagging can be interpreted as an approximation of random aggregating. Subsampled bagging, in turn, can be interpreted as an approximate implementation of random aggregating, where the universe population is replaced by a large training set $\mathcal{D}$, and subsamples are randomly drawn from $\mathcal{D}$ according to an uniform probability distribution.

2.1 Random Aggregating

Let D be a set of m points drawn identically and independently from U according to P, where U is a population of labeled training data points $(\boldsymbol{x}_j, t_j)$, and $P(\boldsymbol{x}, t)$ is the joint distribution of the data points in U, with $\boldsymbol{x} \in \mathbb{R}^d$.

Let $\mathcal{L}$ be a learning algorithm, and define $f_D = \mathcal{L}(D)$ as the predictor produced by $\mathcal{L}$ applied to a training set D. The model produces a prediction $f_D(\boldsymbol{x}) = y$. Suppose that a sequence of learning sets $\{D_k\}$ is given, each i.i.d. from the same underlying distribution P. Breiman proposed to aggregate the f_D trained with different samples drawn from U to get a better predictor $f_A(\boldsymbol{x}, P)$ [1]. For regression problems $t_j \in \mathbb{R}$ and $f_A(\boldsymbol{x}, P) = E_D[f_D(\boldsymbol{x})]$, where

$E_D[\cdot]$ indicates the expected value with respect to the distribution of D, while in classification problems $t_j \in S \subset \mathbb{N}$, and $f_A(\boldsymbol{x}, P) = \arg\max_j |\{k|f_{D_k}(\boldsymbol{x}) = j\}|$.

As the training sets D are randomly drawn from U, we name the procedure to build f_A *random aggregating*. In order to simplify the notation, we denote $f_A(\boldsymbol{x}, P)$ as $f_A(\boldsymbol{x})$. Considering regression problems, if T and $\boldsymbol{X}$ are random variables having joint distribution P, the expected squared loss EL for the single predictor $f_D(\boldsymbol{X})$ is:

$$EL = E_D[E_{T,\boldsymbol{X}}[(T - f_D(\boldsymbol{X}))^2]] \tag{1}$$

while the expected squared loss EL_A for the aggregated predictor is:

$$EL_A = E_{T,\boldsymbol{X}}[(T - f_A(\boldsymbol{X}))^2] \tag{2}$$

Breiman showed [1] that $EL \geq EL_A$. This disequality depends on the instability of the predictions, that is on how unequal the two sides of eq. 3 are:

$$E_D[f_D(\boldsymbol{X})]^2 \leq E_D[f_D^2(\boldsymbol{X})] \tag{3}$$

There is a strict relationship between the instability and the variance of the base predictor. Indeed the variance $V(\boldsymbol{X})$ of the base predictor is:

$$\begin{aligned} V(\boldsymbol{X}) &= E_D[(f_D(\boldsymbol{X}) - E_D[f_D(\boldsymbol{X})])^2] \\ &= E_D[f_D^2(\boldsymbol{X})] - E_D[f_D(\boldsymbol{X})]^2 \end{aligned} \tag{4}$$

Comparing eq. 3 and 4 we see that higher the instability of the base classifiers, higher their variance is. In other words the reduction of the error in random aggregating is due to the reduction of the variance component (eq. 4) of the error, and hence if $V(\boldsymbol{X})$ is quite large we may expect a considerable reduction of the error.

Breiman showed also that in classification problems, as in regression, aggregating relatively good predictors can lead to better performances, as long as the base predictor is unstable, whereas, unlike regression, aggregating poor predictors can lower performances.

2.2 Bagging

Bagging is an approximation of random aggregating, for at least two reasons. First, bootstrap samples are not real data samples: they are drawn from a data set D, that is in turn a sample from the population U. On the contrary f_A uses samples drawn directly from U. Second, bootstrap samples are drawn from D through an uniform probability distribution, which is only an approximation of the unknown true distribution P.

Moreover each base learner, on the average, uses only 63.2% of the available data with bagging, and so we may expect for each base learner a larger bias, as the effective size of the learning set is reduced. This can also affect the bias of the bagged ensemble that critically depends on the bias of the component base learners: we could expect sometimes an increment of the bias of the bagged ensemble with respect to the unaggregated predictor trained on the entire available training set.

2.3 Subsampled Bagging

While bagging had been successfully applied to different classification and regression problems [2,6,7], random aggregating is almost ideal, because in most cases the true distribution P is unknown and often only small data sets are available.

In order to obtain an approximation of random aggregating, we can approximate the universe U with a large data set $\mathcal{D}$ from which subsampled data, that is samples $D \subset \mathcal{D}$ much smaller than $\mathcal{D}$, are randomly drawn with replacement.

In real problems, if we have very large learning sets, or on-line available learning data, we could use *subsampled bagging* in order to overcome the space complexity problem arising from learning too large data sets, or to allow on-line learning [8]. For instance most of the implementations of the SVM learning algorithm have a $\mathcal{O}(n^2)$ space complexity, where n is the number of examples. If n is relatively large (e.g. $n = 10^6$) we need room for 10^{12} elements, a too costly memory requirement for most current computers, even if SVMs implemented through decomposition methods can achieve a certain reduction of space and time complexity [9]. As an alternative, we could use relatively small data sets randomly drawn from the large available data set, using random subsampling and aggregation techniques [8,10].

3 Bias–Variance Decomposition of the Error

Bias–variance decomposition of the error has been recently applied as a tool to analyze learning algorithms and ensemble methods [5,4]. The main purpose of this analysis consists in discovering relationships between properties and parameters of learning algorithms and ensemble methods with respect to their bias–variance characteristics, in order to gain insights into their learning behaviour and to develop ensemble methods well-tuned to the properties of a specific base learner [11].

Here we present only a brief overview of the main concepts of Domingos' bias–variance decomposition of the error theory [12] necessary to understand the bias–variance analysis of random aggregating and bagging. For more details about the application of Domingos theory to the bias–variance analysis of ensemble methods see [11].

Expected loss depends on the randomness of the training set and the target. Let $\mathcal{L}$ be a learning algorithm, and define $f_D = \mathcal{L}(D)$ as the classifier produced by $\mathcal{L}$ applied to a training set D, where D is a set of m points $(\boldsymbol{x}_j, t_j)$, $t_j \in \mathcal{C}$, $\boldsymbol{x}_j \in \mathbb{R}^d$, $d \in \mathbb{N}$, drawn identically and independently from the "universe" population U according to $P(\boldsymbol{x}, t)$, the joint distribution of the data points in U. The model produces a prediction $f_D(\boldsymbol{x}) = y$. Let $L(t, y)$ be the 0/1 loss function, that is $L(t, y) = 0$ if $y = t$, and $L(t, y) = 1$ otherwise. The expected loss EL of a learning algorithm $\mathcal{L}$ at point $\boldsymbol{x}$ can be written by considering both the randomness due to the choice of the training set D and the randomness in t due to the choice of a particular test point $(\boldsymbol{x}, t)$: $EL(\mathcal{L}, \boldsymbol{x}) = E_D[E_t[L(t, f_D(\boldsymbol{x}))]]$,

where $E_D[\cdot]$ and $E_t[\cdot]$ indicate the expected value with respect to the distribution of D, and to the distribution of t.

Purpose of bias–variance analysis. It consists in decomposing the expected loss into terms that separate the bias and the variance. To derive this decomposition, we need to define the *optimal prediction* and the *main prediction*: bias and variance can be defined in terms of these quantities. The *optimal prediction* y_* for point $\boldsymbol{x}$ minimizes $E_t[L(t,y)] : y_*(\boldsymbol{x}) = \arg\min_y E_t[L(t,y)]$. The *main prediction* y_m at point $\boldsymbol{x}$ is defined as $y_m = \arg\min_{y'} E_D[L(f_D(\boldsymbol{x}), y')]$. For 0/1 loss, the main prediction is the class predicted most often by the learning algorithm $\mathcal{L}$ when applied to training sets D.

Bias and variance. The *bias* $B(\boldsymbol{x})$ is the loss of the main prediction relative to the optimal prediction: $B(\boldsymbol{x}) = L(y_*, y_m)$. For 0/1 loss, the bias is always 0 or 1. We will say that $\mathcal{L}$ is *biased at point* $\boldsymbol{x}$, if $B(\boldsymbol{x}) = 1$. The *variance* $V(\boldsymbol{x})$ is the average loss of the predictions relative to the main prediction: $V(\boldsymbol{x}) = E_D[L(y_m, f_D(\boldsymbol{x}))]$ It captures the extent to which the various predictions $f_D(\boldsymbol{x})$ vary depending on D.

Unbiased and biased variance. Domingos distinguishes between two opposite effects of variance on the error with classification problems: in the unbiased case variance increases the error, while in the biased case variance decreases the error. As a consequence we can define an *unbiased variance,* $V_u(\boldsymbol{x})$, that is the variance when $B(\boldsymbol{x}) = 0$ and a *biased variance,* $V_b(\boldsymbol{x})$, that is the variance when $B(\boldsymbol{x}) = 1$. Finally we can also define the *net variance* $V_n(\boldsymbol{x})$, to take into account the combined effect of the unbiased and biased variance: $V_n(\boldsymbol{x}) = V_u(\boldsymbol{x}) - V_b(\boldsymbol{x})$

Noise-free decomposition of the 0/1 *loss.* The decomposition for a single point $\boldsymbol{x}$ can be generalized to the entire population by defining $E_{\boldsymbol{x}}[\cdot]$ to be the expectation with respect to $P(\boldsymbol{x})$. Then we can define the *average bias* $E_{\boldsymbol{x}}[B(\boldsymbol{x})]$, the *average unbiased variance* $E_{\boldsymbol{x}}[V_u(\boldsymbol{x})]$, and the *average biased variance* $E_{\boldsymbol{x}}[V_b(\boldsymbol{x})]$. In the noise-free case, the expected loss over the entire population is:

$$E_{\boldsymbol{x}}[EL(\mathcal{L}, \boldsymbol{x})] = E_{\boldsymbol{x}}[B(\boldsymbol{x})] + E_{\boldsymbol{x}}[V_u(\boldsymbol{x})] - E_{\boldsymbol{x}}[V_b(\boldsymbol{x})].$$

4 Experimental Bias–Variance Analysis in Bagged and Random Aggregated Ensembles of SVMs

The main purpose of the experiments consists in understanding the effect of bagging and random aggregating on the bias and variance components of the error. We used gaussian, polynomial and dot-product Support Vector Machines (SVMs) as base learners.

4.1 Experimental Set–Up

For bagging we draw with replacement from a learning set $\mathcal{D}$ $n = 100$ samples D_i of size $s = 100$, according to an uniform probability distribution. From each D_i, $1 \leq i \leq n$, we generate by bootstrap $m = 60$ replicates D_{ij}, collecting them in n different sets $\bar{D}_i = \{D_{ij}\}_{j=1}^{m}$.

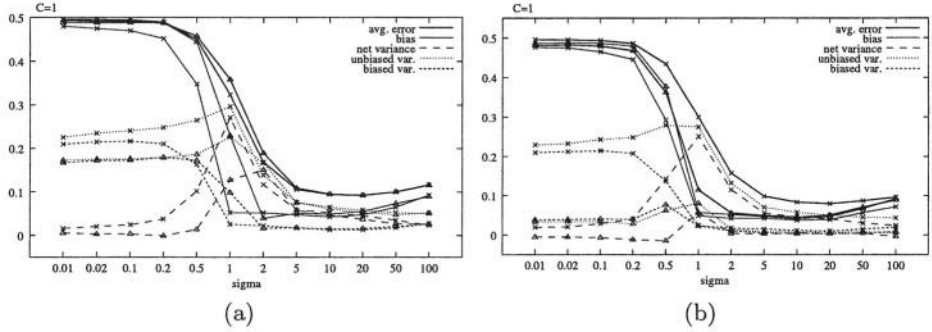


Fig.1. Comparison of bias-variance decomposition between single gaussian SVMs (lines labeled with crosses) and gaussian SVM ensembles (lines labeled with triangles), while varying σ and for a fixed values of the regularization parameter ($C = 1$) with the Letter-Two data set: (a) Bagged ensemble (b) Random aggregated ensemble.

We approximated random aggregating through subsampled bagging. In particular we draw with replacement from $\mathcal{D}$ $n = 100$ sets of samples $\bar{D}_i$, according to an uniform probability distribution. Each set of samples $\bar{D}_i = \{D_{ij}\}_{j=1}^{m}$ is composed by $m = 60$ samples D_{ij} drawn with replacement from $\mathcal{D}$, using an uniform probability distribution; each D_{ij} is composed by 100 examples. Note that in this case each sample D_{ij} is directly drawn from $\mathcal{D}$ and not from the samples $D_i \subset \mathcal{D}$. In order to evaluate the bias–variance decomposition of the error for single SVMs, we used 100 samples D_i of size 100.

We computed the error and its decomposition in bias, net–variance, unbiased and biased variance with respect to a separated large testing set $\mathcal{T}$, comparing the bias–variance decomposition of the error in random aggregated, bagged and single SVMs. To perform this computationally intensive analysis (more than 10 millions of single SVMs were trained and tested) we used a cluster of workstations, and we developed new classes and specific C++ applications, using the *NEURObjects* software library [13] and the *SVMlight* applications [9].

4.2 Results

In particular we analyzed the relationships of the components of the error with the kernels and kernel parameters, using data sets from UCI [14] (Waveform, Grey-Landsat, Letter-Two, Letter-Two with added noise, Spam, Musk) and the P2 synthetic data set[1]. We achieved a characterization of the bias–variance decomposition of the error in bagged and random aggregated ensembles that resembles the one obtained for single SVMs [5] (Fig. 1. For more details see [11]). The results show that bagging reduces the error with respect to single SVMs, but not so largely as random aggregating. Fig. 2 summarizes the main outcomes of our experiments: bagged ensembles of SVMs achieved a reduction of the relative average error from 0 to 20 % with 35% decrement of the net-variance, while

[1] The application `gensimple` for the automatic generation of the P2 data set is available at `ftp://ftp.disi.unige.it/person/ValentiniG/BV/gensimple`.

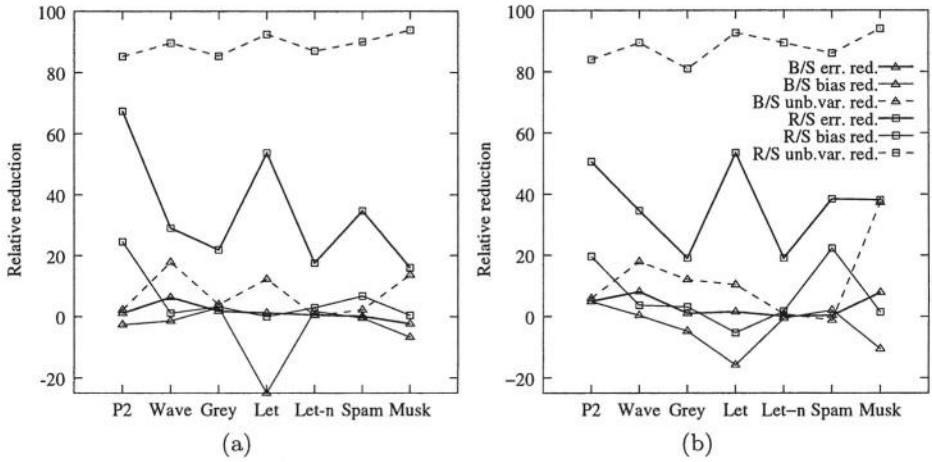


Fig. 2. Comparison of the relative error, bias and unbiased variance reduction between bagged and single SVMs (lines labeled with triangles), and between random aggregated and single SVMs (lines labeled with squares). B/S stands for Bagged versus Single SVMs, and R/S for random aggregated versus Single SVMs. Results refers to 7 different data sets. (a) Gaussian kernels (b) Polynomial kernels.

random aggregated ensembles obtained about a 90 % net-variance decrement, with a relative reduction of the error from 20 to 70 %. Note that a negative sign in the relative bias reduction means that bagging with some data sets may increment the bias with respect to single SVMs (Fig. 2).

These results are not surprising: bagging is a variance reduction method, but we cannot expect as a large decrement of the variance as in random aggregating. Indeed with random aggregating the base learners use more variable training sets drawn from U. In this way random aggregating exploits information from the entire population U, while bagging can exploit only the information from a single data set D drawn from U, through bootstrap replicates of the data drawn from D. Note that these results do not show that random aggregating is "better" than bagging, as they are referred to differently sized data sets, but rather they confirm the theoretical variance reduction properties of random aggregating and bagging. Moreover they show that with well-tuned and accurate base learners a very large variance reduction is obtained only with random aggregating (Fig. 2). Anyway note that in our experiments we used small samples: we could expect that with larger samples the differences between random aggregating and bagging will be smaller.

5 Subsampled Bagged Ensembles of SVMs for Large Scale Classification Problems

The bias–variance analysis of random aggregated ensembles shows that the variance component of the error is strongly reduced, while the bias remains unchanged or it is lowered (Fig. 2). Breiman proposed random subsampling tech-

Table 1. Comparison of the results between single and subsampled bagged SVMs (see text).

	N.samples	*Error*	*Time*	$\Delta Error$	$Speed - up$
S-SVM	100000	0.0023	5474.68	——	——
SB-SVM	10000	0.0032	4089.82	0.39	1.3
SB-SVM	5000	0.0043	855.85	0.87	6.4
SB-SVM	2000	0.0076	124.93	2.30	43.8
SB-SVM	1000	0.0127	38.49	4.52	142.2
SB-SVM	200	0.0358	2.92	14.57	1874.9
SB-SVM	100	0.0539	1.69	22.43	3239.4

niques for classification in large databases, using decision trees as base learners [8], and these techniques have been also successfully applied in distributed environments [10].

These facts suggest to apply subsampled bagging to large scale classification problems, using SVMs as base learners, considering also that the SVM algorithm, as well as other learning algorithms, does not scale too well when very large samples are available [9].

To get some insights into this hypothesis, we performed a preliminary experiment with the synthetic data set *P2,* using a quite large learning set of 10^5 examples, and comparing the results of single and subsampled bagged SVMs on a separated large testing set. Tab. 1 summarizes the results of the experiments: S-SVM stands for single gaussian SVMs trained on the entire available learning set; SB-SVM stands for Subsampled Bagged SVMs trained on subsamples of the available training set, whose cardinality are shown in the column *N.samples.* The column *Error* shows the error on the separated test set (composed by 100000 examples); the column *Time* shows the the training time in seconds, using an AMD Athlon 2000+ processor (1.8 GHz) with 512 Mb RAM. $\Delta Error$ is the relative error increment using an ensemble instead of a single SVM, and the last column represents the speed-up achieved.

The results show that with subsampled bagging we can obtain a consistent speed-up, but with a slight decrement in the accuracy. For instance, we need two minutes for training ensembles with 1/50 of the available data against one hour and a half for a single SVM trained on the entire data set (Tab. 1), with an increment of the error from 0.23 % to 0.76 % (a small absolute difference,but a high relative error increment). Note that if accuracy is not the main goal, we achieve most of the error reduction with about 30 base learners (Fig. 3), and using a parallel implementation we may expect a further speed-up linear in the number of the base learners.

These results confirm similar ones obtained in [15]. Moreover the bias–variance analysis explains the behaviour of subsampled bagging when small learning sets are used: the increment of the bias due to the reduction of the size of the samples is partially counterbalanced by the reduction of the net–variance (data not shown), and on the other hand a substantial speed-up is obtained.

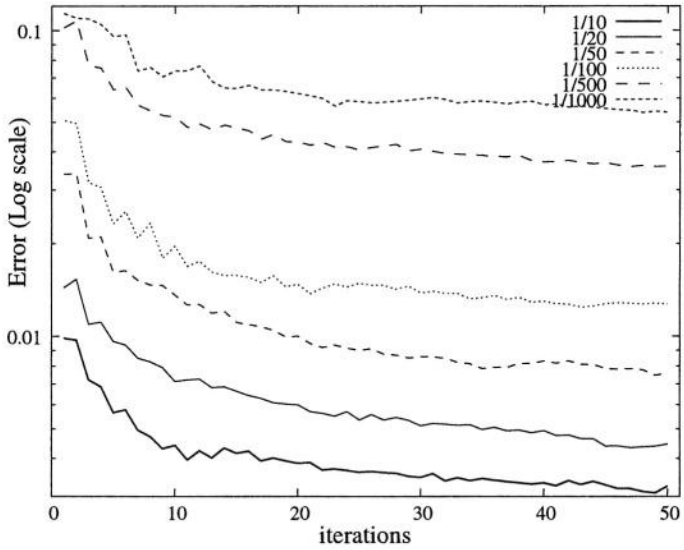


Fig.3. Error (Log scale) as a function of the number of base learners (gaussian SVM) employed. The different curves refer to ensembles with base learners trained with different fractions of the learning set.

6 Conclusions

The extensive experimental analysis of bias–variance decomposition of the error in random aggregated and bagged ensembles of SVMs shows a consistent reduction of the net-variance with respect to single SVMs. This behaviour is due primarily to the unbiased variance reduction, while the bias remains substantially unchanged.

In our experiments random aggregating always reduces the variance close to 0, while classical bagging reduces the variance only of about 1/3, confirming the theoretical results of Breiman, and providing a quantitative estimate of the variance reduction both in bagged and random aggregated ensembles of SVMs.

Preliminary results show also that subsampled bagging, as an approximation of random aggregating, can be in practice applied to classification problems when single SVMs cannot comfortably manage very large data sets, if accuracy is not the main concern.

Acknowledgments

This work has been funded by the italian COFIN project *Linguaggi formali ed automi: metodi, modelli ed applicazioni.*

References

1. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
2. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. Machine Learning **40** (2000) 139–158
3. Bousquet, O., Elisseeff, A.: Stability and Generalization. Journal of Machine Learning Research **2** (2002) 499–526
4. Valentini, G., Dietterich, T.G.: Bias–variance analysis of Support Vector Machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research (accepted for publication)
5. Valentini, G., Dietterich, T.: Bias–variance analysis and ensembles of SVM. In: MCS2002, Cagliari, Italy. Vol. 2364 of Lecture Notes in Computer Science., Springer-Verlag (2002) 222–231
6. Andersen, T., Rimer, M., Martinez, T.R.: Optimal artificial neural network architecture selection for voting. In: Proc. of the IEEE International Joint Conference on Neural Networks IJCNN'01, IEEE (2001) 790–795
7. Kim, H., Pang, S., Je, H., Kim, D., Bang, S.: Pattern Classification Using Support Vector Machine Ensemble. In: Proc. of ICPR'02. Vol. 2., IEEE (2002) 20160–20163
8. Breiman, L.: Pasting Small Votes for Classification in Large Databases and On-Line. Machine Learning **36** (1999) 85–103
9. Joachims, T.: Making large scale SVM learning practical. In Scholkopf B., Burges C., S.A., ed.: Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 169–184
10. Chawla, N., Hall, L., Bowyer, K., Moore, T., Kegelmeyer, W.: Distributed pasting of small votes. In: MCS2002, Cagliari, Italy. Vol. 2364 of Lecture Notes in Computer Science., Springer-Verlag (2002) 52–61
11. Valentini, G.: Ensemble methods based on bias–variance analysis. PhD thesis, DISI, Università di Genova, Italy (2003), ftp://ftp.disi.unige.it/person/ValentiniG/Tesi/finalversion/vale-th-2003-04.pdf.
12. Domingos, P.: A Unified Bias-Variance Decomposition for Zero-One and Squared Loss. In: Proc. of the Seventeenth National Conference on Artificial Intelligence, Austin, TX, AAAI Press (2000) 564–569
13. Valentini, G., Masulli, F.: NEURObjects: an object-oriented library for neural network development. Neurocomputing **48** (2002) 623–646
14. Merz, C., Murphy, P.: UCI repository of machine learning databases (1998) www.ics.uci.edu/mlearn/MLRepository.html.
15. Evgeniou, T., Perez-Breva, L., Pontil, M., Poggio, T.: Bounds on the Generalization Performance of Kernel Machine Ensembles. In Langley, P., ed.: Proc. of the Seventeenth International Conference on Machine Learning (ICML 2000), Morgan Kaufmann (2000) 271–278

Building Diverse Classifier Outputs to Evaluate the Behavior of Combination Methods: The Case of Two Classifiers

Héla Zouari[1,2], Laurent Heutte[1], Yves Lecourtier[1], and Adel Alimi[2]

[1] Laboratoire Perception, Systèmes, Information (PSI), Université de Rouen
Place Emile Blondel, 76821 Mont-Saint-Aignan, CEDEX, France
{Hela.Khoufi,Laurent.Heutte,Yves.Lecourtier}@univ-rouen.fr
http://psiserver.insa-rouen.fr/psi/fr/accueil.html
[2]Groupe de Recherche sur les Machines Intelligentes (REGIM), Université de Sfax
Ecole National des ingénieurs, BP W, Sfax, 3038, Tunisie
Adel.Alimi@ieee.org

Abstract. In this paper, we report an experimental comparison between two widely used combination methods, i.e. sum and product rules, in order to determine the relationship between their performance and classifier diversity. We focus on the behaviour of the considered combination rules for ensembles of classifiers with different performance and level of correlation. To this end, a simulation method is proposed to generate with fixed accuracies and diversity a set of two classifiers providing measurement outputs. A measure of distance is used to estimate the correlation among the pairs of classifiers. Our experimental results tend to show that with diverse classifiers providing no more than three solutions, the product rule outperforms the sum rule, whereas when classifiers provide more solutions, the sum rule becomes more interesting.

1 Introduction

Combining classifiers is now a common approach for improving the performance of recognition systems. Diversity among classifiers is considered as a desired characteristic to achieve this improvement. Though still an ill-defined concept, the diversity has been studied recently in many multiple classifier problems such as analysing the theoretical relationships between diversity and classification error [12], using diversity for identifying the minimal subset of classifiers that achieves the best prediction accuracy [1, 5] or building ensembles of diverse classifiers [8]. For this last direction, i.e. construction of diverse classifiers, several implicit and explicit methods have been investigated. Duin [3] lists the main implicit approaches to build diverse classifiers. The principal one is to use different data representations adapted to the classifiers [7]. The diversity can also be implicitly encouraged either by varying the classifier topology [11], the learning parameters [6] or by training each classifier on different parts of the data which is done for example by Bagging [2]. On the contrary, the aim of explicit methods is to design a set of classifiers by asserting the diversity measure in

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 273–282,2004.

the process of building ensembles [8, 9]: the advantage of the incorporation of diversity measure is to control a priori the diversity between the classifier outputs in order to facilitate the analysis of the combination behavior. Although interesting, the existing works are rather limited since the generated classifier ensembles can only be used to study the performance of abstract-level combination methods and are thus not applicable to evaluate the measurement type combination methods.

For measurement type combination methods, the use of independent classifiers is considered to be essential to achieve better performance. Therefore, one can found in the literature numerous studies dealing with the evaluation of measurement type combination methods with independent classifiers. Among these studies, one can refered the works presented in [7] and [11], For instance, the authors in [7] present an experimental comparison between various combination schemes such as the product rule, sum rule, min rule, max rule, median rule and majority voting. They show that under the assumption of independent classifiers, the sum rule outperforms other classifier combination schemes and is more resilient to estimation errors. Under the same assumption, the work presented in [11] compares the mean rule and the product rule and confirms that in the case of a two-class problem, the combination rules perform the same, whereas in the case of problems involving multiple classes and only with poor posterior probability estimates, the mean rule is to be used. The analysis of the performance of the measurement type combination methods according to diversity has on the contrary received less attention [4, 10, 12]. In [4] for example, the authors report a theoretical and experimental comparison between the simple average and the weighted average. They show that weighted averaging significantly improves the performance of simple averaging only for ensembles of classifiers with highly imbalanced performance and correlations. In [10], the author demonstrates on simulated data that product can perform better than sum with independent classifiers. With correlated classifiers, sum outperforms product whatever the performance of classifiers. As we can see from the studies of sum and product rules reported in the literature, the experimental results about the performance of the two rules are still conflicting. For instance, in [7], it was found that sum outperforms product under the assumption of classifier independence, while, in [10], product can perform better than sum. One can also note that to the best of our knowledge, there are no reported results for these two rules (i.e. product and sum) concerning the relationship between their performance and classifier diversity.

In this paper, we report an experimental comparison between sum and product rules according to classifier dependency. The aim of our experiments is to investigate the behaviour of the above fusion rules for classifier ensembles with different level of correlation. To this end, we propose a simulation method for building pairs of correlated classifiers. The idea behind this method is to use the simulator developped in [13] for generating classifier outputs according to fixed accuracies and diversity. A distance measure is used to estimate this diversity between the classifier confidences. The paper is thus organized as follows. Section 2 presents the diversity measure used to generate the classifier outputs. Section 3 describes the principle of the proposed method. The algorithm for generating two classifiers according to a predefined distance and fixed accuracies is presented in section 4. First experimental results are

reported in section 5 to show the efficiency of the proposed method. Conclusion is drawn in section 6.

2 Measuring the Diversity

In a recent work [13], we have proposed a classifier simulator for the evaluation of measurement type combination methods. Though this simulator was able to generate randomly measurement type outputs (i.e. a confidence value associated to each solution proposed by the classifier), it should be noted however that this classifier simulator was only able to create independent classifiers. We have therefore investigated the generation of correlated classifiers. Building such ensembles of correlated classifiers needs to use an appropriate measure to control the diversity between the classifier confidences. Distance, correlation or mutual information are possible measures that can be used to estimate the diversity between confidences [1]. Although used in many classification problems, there is no analytic proof that a particular measure is preferable to another.

In this work, we use the distance to estimate the diversity between two classifiers. Precisely, consider two classifiers, A and B, providing S outputs for a N-class classification problem. Let s_i^A and s_i^B (i=1 to S) be their outputs, each of which is composed of up to K solutions with $K \geq 1$ and $K \leq N$. An example of output lists is given in Figure 1 (the generated confidences are normalized).

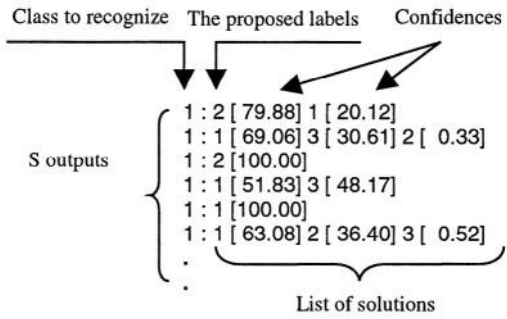


Fig. 1. Description of the classifier outputs.

Each output s_i^A (respectively s_i^B) can be represented as a confidence vector $(c_{i1}^A, ..., c_{iN}^A)$ where c_{ij}^A is the confidence associated to class j. As we assume that the confidences are normalized, we have therefore $\sum_{j=1}^{N} c_{ij}^A = 100$ with $c_{ij}^A \in [0,100]$.

The *diversity* D_{AB} between the two classifiers A and B providing this type of outputs, is defined as:

$$D_{AB} = \frac{1}{S}\sum_{i=1}^{S} d(s_i^A, s_i^B) \quad (1)$$

where $d(s_i^A, s_i^B)$ is the Hamming distance between the outputs s_i^A and s_i^B given by:

$$d(s_i^A, s_i^B) = \frac{1}{200}\sum_{j=1}^{N} \left| c_{ij}^A - c_{ij}^B \right| \quad (2)$$

200 is the maximal distance between two confidence vectors so that the distance $d(s_i^A, s_i^B)$ is normalized and varies between 0 and 1. $d(s_i^A, s_i^B) = 0$ means that the outputs s_i^A and s_i^B contain the same solutions with the same confidences. On the contrary, $d(s_i^A, s_i^B) = 1$ indicates that the two outputs are totally different (there is no intersection of class labels as shown in the following example for a 5-class problem):

Classifier A	Classifier B
1: 1 [75.34] 2 [24.66]	1: 3 [60.00] 4 [40.00]
1: 1 [100.00]	1: 2 [50.00] 5 [50.00]
1: 2 [100.00]	1: 4 [100.00]

We slightly favoured the distance measure for controlling the diversity for the following reasons: D_{AB} is a simple measure that does not depend on the classifier accuracy and its range is limited between 0 and 1 representing the extremes of positive and negative dependency. We thus assume that for building classifier ensembles, controlling the diversity by means of a distance is somewhat equivalent to control the correlation between the classifiers. This idea is illustrated in Figure 2 showing the relationship between correlation and distance measured for a 5-class problem between 50 pairs of classifiers having the same accuracy p=0.6.

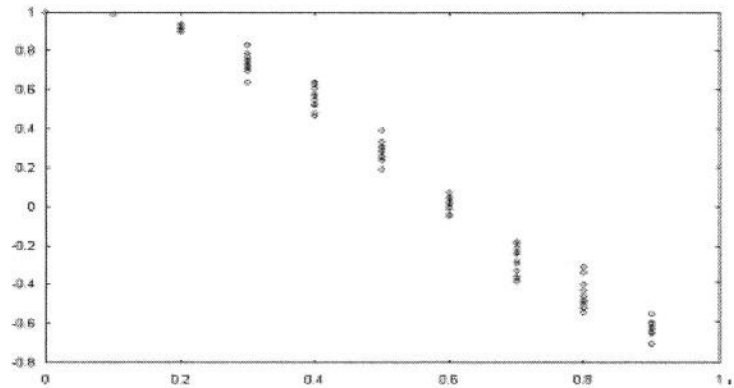


Fig. 2. Relationship between correlation and distance with 50 pairs of classifiers (where x-axis stands for the distance and y-axis for the correlation).

3 Principle of the Method

As noted previously, the output generation of the first classifier of the team is realized by the classifier simulator presented in [13]. In this section, we are interested in building a second classifier B from the outputs of the first classifier A according to a desired distance δ.

To illustrate the principle of the method, let us consider, without loss of generality, the generation of B outputs from A outputs for a 3-class problem. In this case, any output of classifier A, say $s_i^A = (c_{i1}^A, c_{i2}^A, c_{i3}^A)$, belongs to the plane P defined by $\sum_{j=1}^{3} c_{ij}^A = 100$ with $c_{ij}^A \in [0,100]$ as shown in Figure 3 (the plane P is the grey triangle).

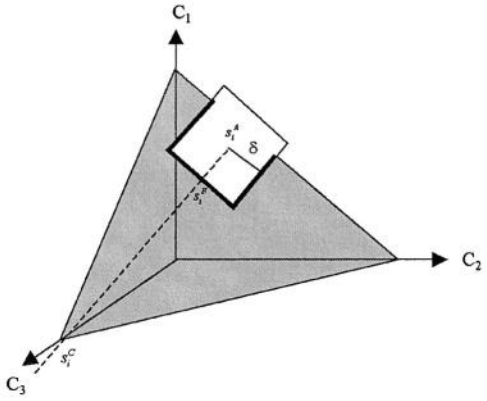


Fig. 3. Determination of s_i^B from s_i^A through a barycentric method.

Now, generating an output s_i^B of classifier B at a distance δ from an output s_i^A of classifier A comes down to determine the intersection between the plane P and the set of points s_i^B which respect (see figure 3):

$$\sum_{j=1}^{3} \left| c_{ij}^A - c_{ij}^B \right| = \delta \text{ with } \sum_{j=1}^{3} c_{ij}^A = 100, \sum_{j=1}^{3} c_{ij}^B = 100 \; ; \; c_{ij}^A, c_{ij}^B \in [0,100] \tag{3}$$

There are of course an infinity of solutions. However, one solution to this problem can be simply obtained through a barycentric calculus of s_i^B. We need for that another point, say s_i^C, belonging to the plane P, which is quite far from s_i^A (see figure 3) so that:

$$\sum_{j=1}^{3} \left| c_{ij}^A - c_{ij}^B \right| = \delta \quad \text{and} \quad \sum_{j=1}^{3} \left| c_{ij}^A - c_{ij}^C \right| = \delta_{\max} \tag{4}$$

It comes therefore from constraints (4) that:

$$\frac{\sum_{j=1}^{3}\left|c_{ij}^{A}-c_{ij}^{B}\right|}{\delta}=\frac{\sum_{j=1}^{3}\left|c_{ij}^{A}-c_{ij}^{C}\right|}{\delta_{\max}} \tag{5}$$

Through algebraic manipulations of equation (5), it comes finally that each confidence of s_i^B can be obtained as:

$$c_{ij}^{B}=\frac{(\delta_{\max}-\delta)c_{ij}^{A}+\delta c_{ij}^{C}}{\delta_{\max}} \tag{6}$$

4 The Generation Algorithm

The aim of the procedure described in this section is to generate automatically, for a N-class problem, S outputs of two classifiers (namely A and B) according to a fixed distance D_{AB} and accuracies p_A and p_B. In this procedure, we first generate the outputs of classifier A according to desired accuracy p_A and we use these outputs to generate the outputs of classifier B through the method described in section 3. The generation of classifier A outputs is beyond the scope of this paper and has been discussed earlier in a separate paper [13]. The idea proposed here is first to generate the outputs of classifier B according to fixed distance D_{AB} and next modify these outputs for respecting the desired accuracy p_B. This consists in generating temporary classifier outputs which are different from those of classifier A (i.e. having a distance of $\delta_{max}=1$) and using them in the formula (6) to determine the confidences of the outputs of B. This process tends however to generate more than K solutions to fit the desired distance. Therefore, in order to respect a desired accuracy in the Top K solutions of classifier B, outputs of more than K solutions are troncated to K solutions and confidences are re-distributed among the remaining classes.

The final step of the generation procedure is to respect the desired accuracy p_B i.e. obtain p_B*S true class labels in the outputs of classifier B. Let us recall that the accuracy p_B is the ratio of the number of true classes that appear in the top K solutions of the output lists among the total number S of outputs. Respecting only this accuracy is easy. For example, if we should increase the number of true class labels, it is sufficient to search the outputs that do not contain the true class and modify one of their class labels and replace it by the true class. However, when we should respect at the same time the accuracy and the desired distance the process is not straightforward. The distance measure used in this work is based on the confidence values associated to the class labels of the two classifier outputs s_i^A and s_i^B for (i=1 to S). Therefore, modifying the class labels needs to take into account the confidence values of the two classifiers.

To illustrate this procedure, suppose that we have generated two outputs for classifier A and that we want to determine the outputs of classifier B according to a desired distance $D_{AB}=0.6$ for a 3-class problem with K=2 and p_B=100%. The temporary outputs (classifier C) are first generated and next used through equation(6) to derive outputs of classifier B, for example:

```
Classifier A    Classifier C            Classifier B
1: 1[100.00]    1: 2[90.00] 3[10.00]    1: 2[54.00] 1[40.00] 3[ 6.00]
1: 2[100.00]    1: 3[100.00]            1: 3[60.00] 2[40.00]
```

As noted above, the barycentric method tends to generate more than K solutions for classifier B (as this is the case for the first output). To respect K, the last solution of the first output of B (i.e. 3 [10.00]) has therefore to be eliminated. One solution to do that is to add the confidence of this solution to the first one (i.e. 2 [54.00]) in order to respect the predefined distance. We do not add the confidence to the second solution because the class label "1" exists in the output of classifier A. After this elimination, we obtain:

```
Classifier A                Classifier B
1: 1[100.00]                1: 2[60.00] 1[40.00]
1: 2[100.00]                1: 3[60.00] 2[40.00]
```

Now, to respect the accuracy p_B, we should add one true class to the second output. In this case, it is possible to choose only the first solution because the second one has a class label that exists in the output of A. This leads finally to the following outputs which respect both the desired distance and the fixed accuracy of classifier B:

```
Classifier A                Classifier B
1: 1[100.00]                1: 2[60.00] 1[40.00]
1: 2[100.00]                1: 1[60.00] 2[40.00]
```

The algorithm for the generation of classifier B outputs from classifier A is presented in Table 1.

Table 1. Generation of output lists of classifier B from classifier A.

Inputs: S (number of outputs), s_i^A (outputs of classifier A),
p_B (the desired accuracy of classifier B),
D_{AB} (the fixed distance between A and B)
Outputs: s_i^B : outputs of classifier B (i=1 to S)
Begin
  For each output i=1 to S **Do**
    Generate an output s_i^C different from s_i^A
    Determine the output s_i^B of classifier B using (6)
  Compute the accuracy p of classifier B
  If (p < p_B) **Then**
  Select among classifier B outputs |p-p_B|*S outputs s_i^B which do not contain a true class label T_i
  Else
  Select among classifier B outputs |p-p_B|*S outputs s_i^B which contain a true class label T_i
  For each selected output s_i^B **Do**
   If $T_i \notin s_i^A$ **Then** select a class label that $\notin s_i^A$ and replace it by T_i
   Else select a class label that $\in s_i^A$ and replace it by T_i
End

5 Experimental Results

In this section, we investigate the performance of sum and product rules according to diversity in a two classifier case for a 10-class problem. For this experiment, we simulate 50 pairs of classifiers for each of different values of diversity D. The classifiers provide 1000 outputs according to the same accuracies $p_A = p_B = 50\%$, 70% or 90% with K=3 (Top3 accuracy). The results are evaluated according to the maximal accuracy obtained by the combination over 50 runs. In the first experiment, we show that whatever the value of p, the performance of the two rules increase as the diversity between the two classifiers increases (see figure 4).

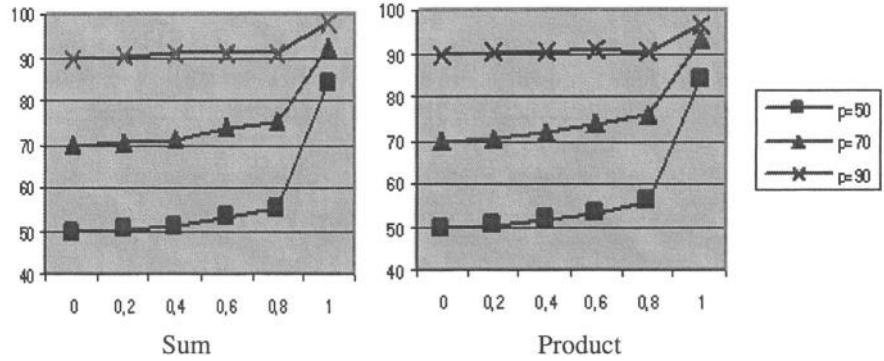


Fig. 4. The performance of sum and product rules vs. diversity for different values of p (where the X-axis stands for the distance between the classifiers and Y-axis stands for the recognition rates of the combination rules).

In the second experiment, we examine the effect on Sum and Product rule performance of the number K of solutions for different degrees of classifier diversity. The experimental conditions are the same as in the first experiment (10-class problem, two classifiers, 50 pairs of classifiers for each of different values of diversity D) but we compare here the two rules when the length of the solution list increases (K=3 or 5) with $p_A = p_B = 90\%$. The accuracy of the combination methods versus the diversity measure is depicted in figure 5. The results show that in the situation of dependent classifiers there are no difference between the two combination rules whatever the number of solutions provided (three or five). But, as the diversity between the classifiers increases, the difference between Sum and Product appears. When the classifiers are different (D>0.6) and provide no more than three solutions, the product rule achieves a significant improvement over the sum rule. However, when the classifiers provide more solutions (five), the sum rule becomes more interesting. This would mean that the product rule is more sensitive to the number of solutions and less efficient especially for diverse classifiers [7]. These preliminary results are obviously to be confirmed by a more intensive generation of pairs of classifiers (currently 50 runs is obviously not sufficient) but they show that our method can be a useful tool to

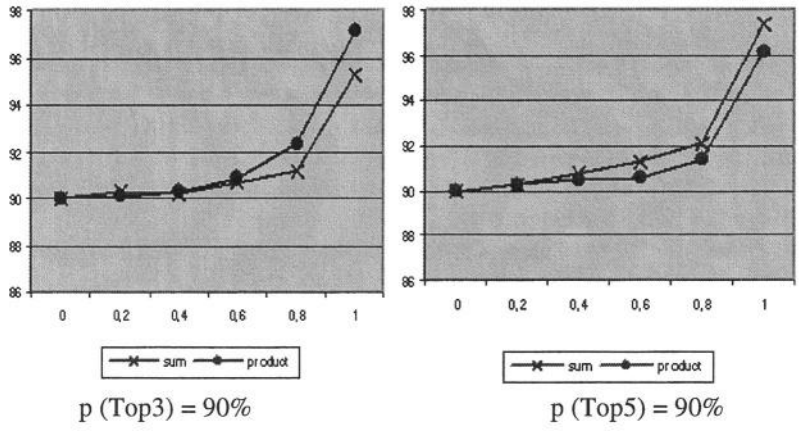


Fig. 5. Comparison of sum and product rules vs diversity for different values of K (where the X-axis stands for the distance between the classifiers and Y-axis stands for the recognition rates of the combination rules).

better understand the effect of classifier diversity on the behavior of measurement-type combination rules.

6 Conclusions

In this paper, we have proposed a new simulation method for the generation of dependent classifier outputs. An algorithm for building two classifiers with specified accuracies and a distance measure between them is presented. Given the outputs of the first classifier, the proposed method relies on a two-step procedure which first generates the outputs of the second classifier according to the specified distance and next modify these outputs in order to respect the predefined accuracy (recognition rate). By a simple example, we have shown that the proposed method can help to clarify (in the case of two classifiers) the conditions under which the measurement combination methods can be used according to diversity. As a future work, we plan to extend the proposed method to generate more than two classifiers and control all the between-pair distances.

References

1. M. Aksela, Comparison of Classifier Selection Methods for Improving Committee Performance, *Lecture Notes in computer Science,* 4th International Workshop, Multiple Classifier Systems, Guildford, UK, (2003), 84-93,
2. L. Breiman, Bagging predictors, *Machine Learning*, 24:2, (1996), 123-140.
3. R.P. Duin, The Combining Classifier: To Train Or Not To Train?, in: R. Kasturi, D. Laurendeau, C. Suen (eds.), ICPR16, *Proceedings 16th International Conference on Pattern Recognition,* Quebec City, Canada, Vol. II, IEEE Computer Society Press, Los Alamitos, (2002), 765-770.

4. G. Fumera, F. Roli, Performance analysis and comparison of linear combiners for classifier fusion, *Inter. Workshop on Statistical Pattern Recognition,* SSPR/SPR, (2002), 424-432.
5. G. Giacinto, F. Roli, Design of Effective Neural Network Ensembles for Image Classification Processes, *Image Vision and Computing Journal,* 19:9/10, (2001), 699-707.
6. L. Hansen, P. Salamon, Neural Network Ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12, (1990), 993-1001.
7. J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On Combining Classifiers, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20:3, (1998).
8. L.I. Kuncheva, R.K. Kountchev, Generating Classifier Outputs of Fixed Accuracy and Diversity, *Pattern Recognition Letters,* 23. (2002), 593-600.
9. V.D. Lecce, G. Dimauro, A. Guerrierro, S. Impedovo, G. Pirlo, A. Salzo, Classifier Combination: The Role of a-priori Knowledge, *in Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition,* Amsterdam, The Netherlands, (2000), 143-152.
10. J.R. Parker, Rank And Response Combination From Confusion Matrix Data. Information Fusion, Vol. 2. (2001) 113-120.
11. D.M.J. Tax, M.V. Breukelen, R.P.W. Duin, J. Kittler, Combining Multiple Classifiers by Averaging or by Multiplying ?, *Pattern Recognition,* 33. (2000) 1475-1485.
12. K. Tumer, J. Gosh, Linear and order statistics combiners for pattern classification, In A. Sharky (Eds.), *Combining Artificial Neural Nets,* London,: Springer-Verlag, (1999), 127-161.
13. H. Zouari, L. Heutte, Y. Lecourtier, A. Alimi, Simulating Classifier Outputs for Evaluating Parallel Combination Method, *Lecture Notes in computer Science, 4th International Workshop, Multiple Classifier Systems,* Guildford, UK, (2003), 296-305.

An Empirical Comparison of Hierarchical vs. Two-Level Approaches to Multiclass Problems

Suju Rajan and Joydeep Ghosh

Laboratory of Artificial Neural Systems
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin TX-78712, USA
{rsuju,ghosh}@lans.ece.utexas.edu

Abstract. The Error Correcting Output Codes (ECOC) framework provides a powerful and popular method for solving multiclass problems using a multitude of binary classifiers. We had recently introduced [10] the Binary Hierarchical Classifier (BHC) architecture that addresses multiclass classification problems using a set of binary classifiers organized in the form of a hierarchy. Unlike ECOCs, the BHC groups classes according to their natural affinities in order to make each binary problem easier. However, it cannot exploit the powerful error correcting properties of an ECOC ensemble, which can provide good results even when the individual classifiers are weak. In this paper, we provide an empirical comparison of these two approaches on a variety of datasets, using well-tuned SVMs as the base classifiers. The results show that while there is no clear advantage to either technique in terms of classification accuracy, BHCs typically achieve this performance using fewer classifiers, and have the added advantage of automatically generating a hierarchy of classes. Such hierarchies often provide a valuable tool for extracting domain knowledge, and achieve better results when coarser granularity of the output space is acceptable.

1 Introduction

Classification techniques such as the k-nearest neighbors and multi-layered perceptron can directly deal with multiclass problems. However, in difficult pattern recognition problems involving a large number of classes, it has often been observed that obtaining a classifier that discriminates between two (groups of) classes is much easier than one that simultaneously distinguishes among all classes. This observation has prompted substantial research on using a collection of binary classifiers to address multiclass problems. Further interest in this area has been fuelled by the popularity of classifiers such as SVMs [1] whose native formulation is for binary classification problems. While several extensions to multiclass SVMs have been proposed, a careful study in [2] showed that none of these approaches are superior to using a set of binary SVMs in an "All Pairs" framework.

The One-Vs-All method [3], the Round Robin Method [4] or the All Pairs method [5] (also known as pairwise method), and the Error Correcting approaches [6] [7] [8] are some of the techniques proposed for solving multiclass problem by decomposing the output space. All of these methods can be unified under a common framework wherein the output space is represented by a binary code matrix, which depends on

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 283–292, 2004.

the technique being used [7] [9]. The above techniques can also be considered as two-level approaches in which the second level provides some relatively simple mechanism to output the final class label based on the decisions obtained from a set of binary "base" classifiers at the first level. Another common characteristic of these methods is that they do not take into consideration the affinities among the individual classes. As a result, some of the groupings might result in complex decision boundaries. In particular, this suggests the need to use powerful base classifiers in the ECOC framework (decision stumps will not do!). The performance of ECOC methods also hinges heavily on the choice of the code matrix, and though multiple solutions have been proposed, none of them are guaranteed to produce the optimal matrix for a problem at hand.

The BHC [10] is a multiclassifier system that was primarily developed to deal with multiclass hyperspectral data. Like the multiclassifier systems mentioned above, the BHC decomposes a C class problem into (C-1) binary meta-class problems. However, the grouping of the individual classes into meta-classes is determined by the class distributions. A recent work [11], that compared the performance of BHC against that of an ECOC system, while using "base" Bayesian classifiers in both systems, showed the superior performance of the BHC method on real-world hyperspectral data. Since the BHC not only yields valuable domain knowledge, but also resolves the problem of having to come up with an optimal code matrix, we decided to evaluate the performance of the BHC against the ECOC methods on a wide range of standard datasets. SVM, being a popular and powerful binary classifier was used as the "base" classifier in both the systems. Our experiments show that the performance of the BHC is comparable to that of ECOC classifiers while remaining robust for small training sets. We also show that besides using far lesser number of classifiers in most cases, the binary trees produced by the BHC are consistent with those a human expert might have constructed when given just the class labels.

2 Background

2.1 Binary Hierarchical Classifier

The Binary Hierarchical Classifier (BHC) [10] involves recursively decomposing a multiclass (C-classes) problem into (C-1) two meta-class problems, resulting in (C-1) classifiers arranged as a binary tree. The given set of classes is first partitioned into two disjoint meta-classes and each meta-class thus obtained is partitioned recursively until it contains only one of the original classes. The number of leaf nodes in the tree is thus equal to the number of classes in the output space. The partitioning of a parent set of classes into two meta-classes is not arbitrary, but is obtained through a deterministic annealing process, which encourages similar classes to remain in the same partition [12]. Thus, as a direct consequence of the BHC algorithm, classes that are similar to each other in the input feature space are lumped into the same meta-class higher up in the tree. Interested readers are referred to [10] for details of the algorithm. Also, note that the BHC is an example of a coarse-to-fine strategy, which has seen several application-specific successes and for which solid theoretical underpinnings are beginning to emerge [13].

The BHC algorithm offers a lot of flexibility in designing the classifier system. For instance, one can replace the Bayesian classifier at the internal nodes of the generated BHC tree with stronger binary classifiers such as SVM [1]. Moreover, different feature selection methods can be used at each node that is specific to the domain of the input data. Note that the maximum number of classifiers required in a BHC system is (C-1), unlike other methods such as All-Pairs in which the number of classifiers is quadratic in the cardinality of the output space. Further since the BHC essentially clusters the classes, it yields additional information about the inherent relationships among the different classes. Such information can be leveraged in creating classifiers with varying levels of granularity, extracting simple rules and identifying those features that help distinguish between groups of classes.

2.2 ECOC Classifiers

The ECOC method of combining binary classifiers is based on the framework used in [14] for the NETalk system. In this method, the training data of each class is associated with a unique binary string of length n. The different bit positions in the binary string represent a particular independent feature, the presence or absence of which helps differentiate one class from the other. During training, n binary functions are learned, one for each bit position. New data samples are then classified by each of the n classifiers, whose combined output gives rise to a n-bit binary string. The bit string thus obtained is then compared with the representative bit strings of each class, and the class with the closest distance measure to the bit string of the data sample is assumed to have generated it.

This idea was extended in [6] where instead of generating bit strings that in some sense represent the input space, a matrix of code vectors having good error correcting properties was used. Thus, if d is the Hamming distance between any two code vectors in the matrix, then the error correcting properties of the code can correct up to $\lfloor(d\text{-}1)/2\rfloor$ errors. Hence, if we train a set of classifiers, one for each bit position then we can obtain the right class label even when there are $\lfloor(d\text{-}1)/2\rfloor$ errors. These error-correcting codes will be successful only when the errors made by the individual classifiers are independent [15]. The ECOC matrices used in [6] were predefined and did not take into account any other factor inherent in the data other than the number of classes.

An error–correcting matrix is considered to be a good one if, the codewords are well separated in Hamming distance and if the columns of the code matrix are uncorrelated. The latter condition translates into having a large Hamming distance between pairs of columns, and also between each column and the complements of the rest. Binary classifiers do not differentiate between learning a particular bit string or its complement. Hence, even though complementary columns have the maximum Hamming distance it is important to not include either identical or complementary columns in the code matrix. Dietterich et al. [6] provide a simple set of rules to form the code matrices depending on the cardinality of the output space k.

- Exhaustive codes of length 2^{k-1}-1 when $k<=7$.
- Column selection from exhaustive codes when $8<=k<=11$.
- For $k>11$ a method based on randomized hill climbing, or on using BCH [16] codes to form the code matrices.

The above methods however do not guarantee the best possible code matrix for a given problem and involve some heuristics in choosing a good matrix among the available choices. Finding the optimal code matrix is still an open research question though many alternatives have been suggested. Allwein et al. [7] use the confidence of predictions and perform decoding based on a loss function instead of the Hamming distance. The performance of different choices of code matrices was also evaluated. Some of the code matrices used in [7] are,

- A complete code that unifies One-Vs.-All, Error Correcting codes and All-pairs classification.
- A dense random code with $\lfloor 10\log_2(k) \rfloor$ columns for a k-class problem. The code matrix was selected by generating 10,000 matrices in which each column has a uniform distribution of $\{+1$ and $-1\}$ and then choosing that matrix that had the largest inter-row Hamming distance.
- A sparse random code with $\lfloor 15\log_2(k) \rfloor$ columns in which the elements are 0 with probability 0.5 and 1 or –1 with probability 0.25 each. The code matrix with the largest Hamming distance was again chosen as the best one.

Crammer et al. [8] attempt to find a code matrix that is problem dependent by trying to learn a matrix with real-valued entries that has a good performance on the training data. More sophisticated methods such as recursive ECOC [17] have also been proposed.

Though there are numerous methods of combining the different binary classifiers in the ECOC framework, no one method has till date claimed superior performance over the rest in terms of the error rates on the test data. In fact, a recent evaluation [9] of the different multiclassifier methods shows comparable performance of the different classifiers and hence, suggests the use of the One-Vs.-All method (with very well tuned and powerful base classifiers) as being the most intuitive and simplest method of implementing multiclassifier systems.

3 Experiments

Table 1 summarizes the datasets used in our experimental evaluation. All the datasets are publicly available at the UCI Machine Learning Repository [18]. These datasets were chosen as they all have numeric attributes and have been widely used to evaluate ECOC systems. None of the datasets have any missing values. The training data was normalized to have a zero mean and unit variance and the test set was normalized with the mean and variance of the training data. We tested the performance of BHC with base Bayesian Classifiers, BHC with base SVM classifiers and ECOC with base SVM classifiers.

For datasets with independent test sets we used the original test/train split and for datasets which do not contain an independent test set, we performed five-fold cross validation. For the BHC with "base" Bayesian Classifiers, the in-built feature extraction method based on the fisher discriminant was applied at each of the internal nodes. This BHC method did not require the setting of any parameter and hence, the training set was used as is.

Table 1. Datasets

No.	Name	# Train	# Test	# Classes	# Attributes
1.	Glass	214	5C.V	6	9
2.	Yeast	1484	5 C.V	11	9
3.	Satimage	4435	2000	6	36
4.	Segment	2310	5C.V	7	16
5.	Page-Blocks	5473	5C.V	5	10
6.	Pendigits	7494	3498	10	16
7.	Optdigits	3823	1797	10	64
8.	Letter	16000	4000	26	16
9.	Vowel	528	462	11	12

The BHC algorithm was then modified to accommodate the SVM classifiers, which were implemented in Matlab using the toolbox provided by [19]. Each training set was split into a 90% training set and a 10% validation set, which was then used to tune the parameters of the SVM. RBF kernels were used since they consistently yielded better performance than polynomial kernels. To tune the kernel width, it was first set at 1 and then was increased or decreased until there was no improvement on the validation set. The kernel width was then set at the best possible value within the observed range. For both the BHC-SVM and the ECOC-SVM methods, parameter tuning was done individually for each validation set. BHC-SVM also used a simple Forward Feature Selection algorithm at each node, as the nodes towards the leaves tend to have far less training data than the nodes near the root. Feature selection was performed only when the ratio of the number of training samples to the input dimensionality was less than 5 [11].

For the ECOC-SVM method, the kernel parameter was tuned as in the previous case. The code matrices were generated by:

- The Exhaustive method in [6] when the number of classes k<=7.
- The Dense random code method of [7] when 7<k<=11.
- The BCH code matrix (obtained from [19]) when k>11.

Table 2 shows the absolute error rates of the three different classifier schemes. The number in the parentheses under an accuracy rate is the standard deviation of the percentage accuracy over the five cross-validation runs. An asterisk indicates that the results are statistically significantly different from that of the BHC-SVM at the 90% confidence level.

From the above table, it can be seen that the use of SVM as the internal classifier improves the performance of the BHC significantly. The BHC-SVM also has percentage accuracies that are comparable if not better than that of ECOC-SVM in most of the cases. Moreover, the BHC uses fewer classifiers in almost all cases except that of the Letter dataset for which a BCH-like code matrix of length 15 was used. It has been remarked in [6] that the BCH codes have some practical drawbacks and human intervention is necessary to generate good shortened BCH codes. All other methods of binary classifier generation except for the One-Vs-All method use far more classifiers than the BHC while showing similar performance on the UCI datasets [4] [6] [7] [9].

As mentioned earlier, the BHC performs a clustering of the output classes based on their separability and hence the algorithm tends to generate hierarchies of classes that appeal intuitively. A few sample trees for the datasets are included in the Appendix.

Table 2. Percentage Accuracy across the different datasets

No.	Name	BHC	BHC-SVM	ECOC-SVM	# classifiers BHC / ECOC
1.	Glass	59.2 (7.79)	66.59 (6.2)	63.7 (6.14)	6 / 31
2.	Yeast	43.56* (3.25)	56.91 (3.28)	53.81 (3.7)	10 / 34
3.	Satimage	83.85*	91.45	91.4	5 / 31
4.	Segment	43.9* (10.1)	89.94 (1.34)	89.17 (0.90)	6 / 63
5.	Page-Blocks	82.64* (5.66)	94.5 (2.05)	94.8 (1.9)	4 / 15
6.	Pendigits	89.30*	97.9	98.45*	9 / 33
7.	Optdigits	93.9*	96.6	96.99	9 / 33
8.	Letter	73.7* (1.13)	96.23 (0.60)	96.83 (0.32)	25 / 15
9.	Vowel	47.4*	61.03	51.29	10 / 34

The numbers at the internal nodes of the trees represent the meta-classes corresponding to these nodes. It can be seen that the BHC does indeed generate meaningful class hierarchies. For instance in the Satimage tree, the Vegetation/Foliage classes form one meta-class while the different soil classes form another. In some cases, it might be sufficient to just know whether an incoming data sample is one of vegetation or soil. A single classifier at the root node is capable of performing this task whereas in, say, the One-Vs.-All method one would have to evaluate six different binary classifiers to obtain the same result. Feature selection techniques can also be used to generate simple classifiers that can distinguish between the meta-classes and one can identify those features that help distinguish classes at a coarser level of granularity. Such knowledge extraction is not easily possible in the other multiclassifier systems.

A strength of ECOC based classifiers is that it can compensate for weak base classifiers. Hence, one should also examine results based on lesser amounts of training data to see if ECOCs have superior properties in such situations. Figures 1 and 2. show typical BHC-SVM and ECOC-SVM training curves obtained for two of the datasets. The fact that the BHC curves are still comparable to the ECOC-SVM based ones even at very low percentages of training data is noteworthy.

4 Conclusions

The empirical studies presented in this paper show that the Binary Hierarchical Classifier is a very viable approach to multiclass problems. As compared to ECOC, the BHC offers the added advantages of not requiring an optimal code matrix, while using far fewer classifiers and automatically generating class hierarchies. While independence of the (meta)-classes is desired for success of the ECOC schemes, the BHC exploits the very dependence to generate a robust and scalable classifier with several beneficial properties.

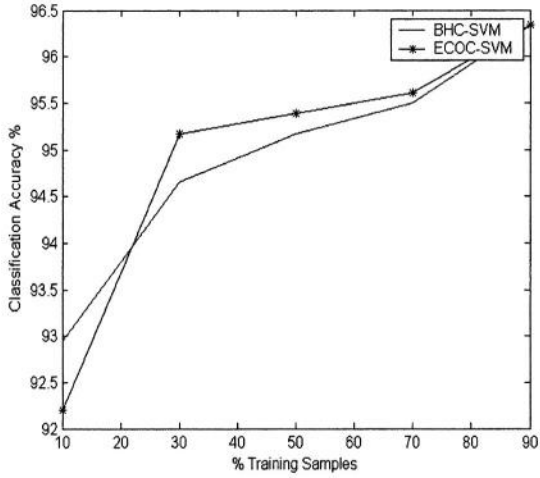


Fig. 1. Training curves for the Page-Blocks Dataset

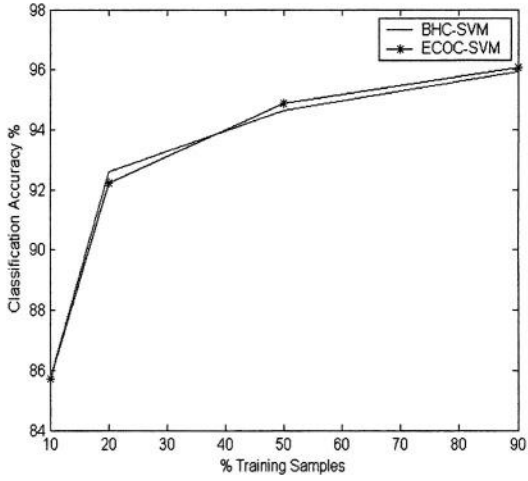


Fig. 2. Training curves for the Letters Dataset

Acknowledgements

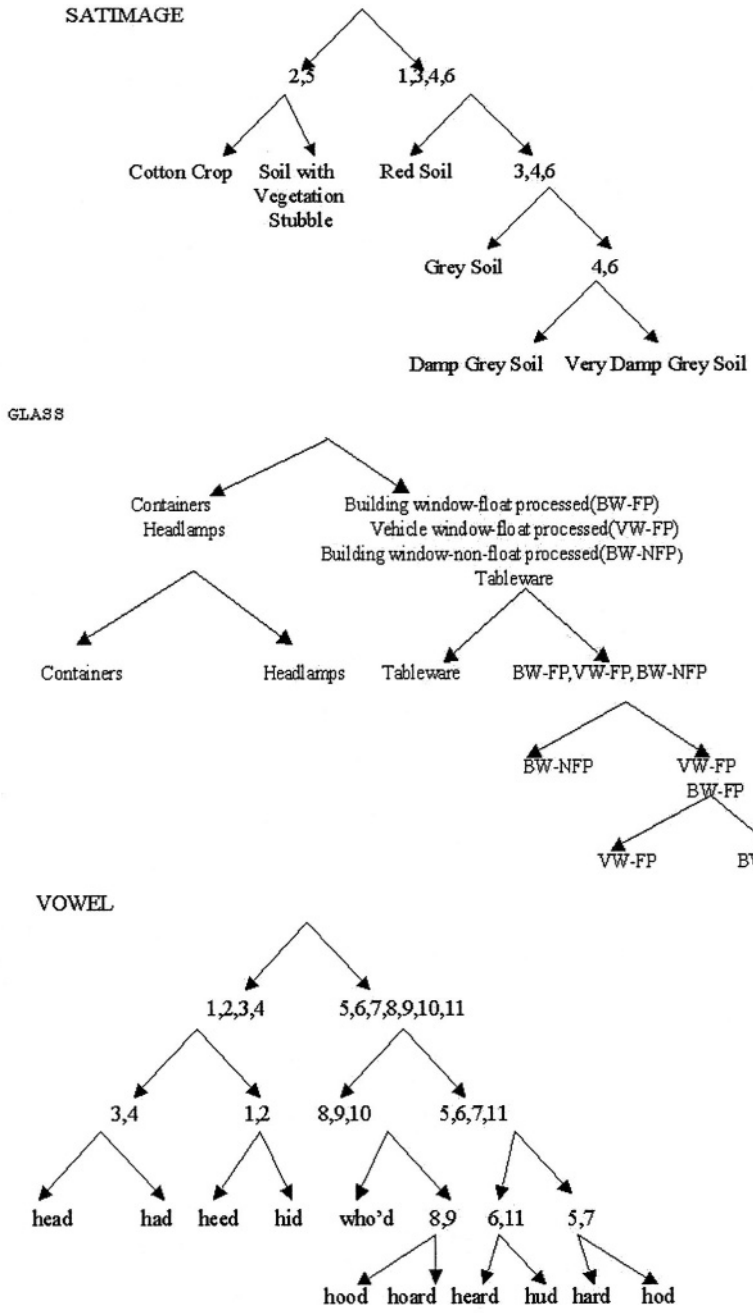
This research was supported by NSF (Grant IIS-0312471). We thank Prof. Melba Crawford and other members of CSR for their collaboration on developing the BHC framework.

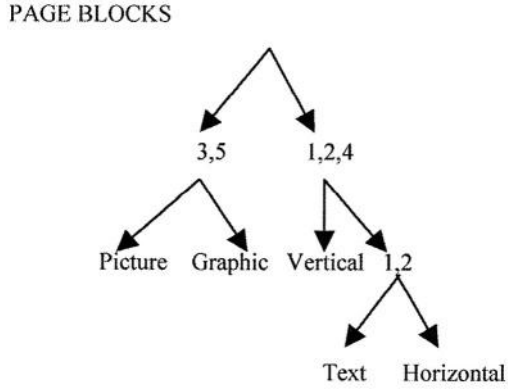
References

1. Vapnik, V.: *The Nature of Statistical Learning Theory.* Springer-Verlag, NY, USA, 1995.
2. Hsu, C., Lin, C.: A Comparison of Methods for Multiclass Support Vector Machines, *IEEE Transactions on Neural Networks,* 13 (2002), 415-425.
3. Nilsson, N. J.: *Learning machines,* McGraw-Hill, New York, 1965.
4. Fürnkranz, J.: Round Robin Classification, *Journal ofMachine Learning Research,* 2(Mar), (2002), 721-747.
5. Hastie, T., Tibshirani, R.: Classification by Pairwise Coupling, *Advances in Neural Information Processing Systems,* Vol.10, The MIT Press, (1998).
6. Dietterich, T. G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes, *Journal of Artificial Intelligence Research,* Vol.2, (1995), 263-286.
7. Allwein, E. L., Schapire, R. E., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Proc. 17th International Conf. on Machine Learning,* Morgan Kaufmann, San Francisco, CA, (2000), 9-16.
8. Crammer, K., Singer, Y.: On the Learnability and Design of Output Codes for Multiclass Problems, *Computational Learning Theory,* (2000), 35-46.
9. Rifkin, R., Klautau, A.: In Defense of One-Vs-All Classification, *Journal of Machine Learning Research,* 5(Jan), (2004), 101-141.
10. Kumar, S., Ghosh, J., Crawford, M. M.: Hierarchical Fusion of Multiple Classifiers for Hyperspectral Data Analysis, *Pattern Analysis and Applications,* spl. Issue on Fusion of Multiple Classifiers Vol. 5, No. 2, (2002), 210-220.
11. Morgan, T. J., Henneguelle, A., Ham, J., Ghosh, J., Crawford, M.M.: Adaptive Feature Spaces for Land Cover Classification with Limited Ground Truth Data, *International. Journal of Pattern Recognition and Artificial Intelligence,* J. Kittler and F. Roli (Eds) (2004) (to appear).
12. Kumar, S., Ghosh, J.: GAMLS: A Generalized framework for Associative Modular Learning Systems, *Application and Science of Computational Intelligence II,* SPIE Vol. 3722, (1999), 24-35.
13. Kittler, J, Ahmadyfard, A, and Windridge, D: Serial Multiple Classifier Systems Exploiting a Coarse to Fine Output Coding, *LNCS, Vol 2709, (Proc. MCS 2003 Workshop),* T. Windeatt and F. Roli (Eds), Springer, (2003), 96-104.
14. Sejnowski, T. J., Rosenberg, C. R.: Parallel Networks that learn to pronounce English text, *Complex Systems,* 1, (1987), 145-168.
15. Kong, E. B., Dietterich, T. G.: Error-Correcting Output Coding Corrects Bias and Variance, *International Conference on Machine Learning,* (1995), 313-321
16. Bose, R. C., Ray-Chauduri, D. K.: On a Class of Error Correcting Binary Group Codes, *Information and Control,* (3), (1960), 68-79.
17. Tapia, E, Gonzalez, J. C., Garcia-Villalba, J: Good Error Correcting Output Codes for Adaptive Multiclass Learning, *LNCS, Vol 2709, (Proc. MCS 2003 Workshop),* T. Windeatt and F. Roli (Eds), Springer, (2003), 156-165.
18. Blake, C. L, Merz, C. J.: UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, (1998).
19. http://www.cis.tugraz.at/igi/aschwaig/software.html

Appendix

A few of the sample trees obtained using the BHC:




PAGE BLOCKS
3,5
1,2,4
Picture
Graphic
Vertical
1,2
Text
Horizontal

Experiments on Ensembles with Missing and Noisy Data

Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J. Mooney

Department of Computer Sciences
University of Texas at Austin, Austin TX 78712, USA

Abstract. One of the potential advantages of multiple classifier systems is an increased robustness to noise and other imperfections in data. Previous experiments on classification noise have shown that bagging is fairly robust but that boosting is quite sensitive. DECORATE is a recently introduced ensemble method that constructs diverse committees using artificial data. It has been shown to generally outperform both boosting and bagging when training data is limited. This paper compares the sensitivity of bagging, boosting, and DECORATE to three types of imperfect data: missing features, classification noise, and feature noise. For missing data, DECORATE is the most robust. For classification noise, bagging and DECORATE are both robust, with bagging being slightly better than DECORATE, while boosting is quite sensitive. For feature noise, all of the ensemble methods increase the resilience of the base classifier.

1 Introduction

In addition to their many other advantages, multiple-classifier systems hold the promise of developing learning methods that are robust in the presence of imperfections in the data; in terms of missing features, and noise in both the class labels and the features. Noisy training data tends to increase the variance in the results produced by a given classifier; however, by learning a committee of hypotheses and combining their decisions, this variance can be reduced. In particular, variance-reducing methods such as *Bagging* [2] have been shown to be robust in the presence of fairly high levels of noise, and can even *benefit* from low levels of noise [3].

Bagging is a fairly simple ensemble method which is generally outperformed by more sophisticated techniques such as ADABOOST [4,13]. However, ADABOOST has a tendency to overfit when there is significant noise in the training data, preventing it from learning an effective ensemble [3]. Therefore, there is a need for a general *ensemble meta-learner*[1] that is at least as accurate as ADABOOST when there is little or no noise, but is more robust to higher levels of random error in the training data.

DECORATE [9,10] is a recently introduced ensemble meta-learner that directly constructs diverse committees by employing specially-constructed artificial

[1] An ensemble meta-learner, like Bagging and ADABOOST, takes an arbitary *base learner* and uses it to build a more effective committee of hypotheses [17].

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 293–302, 2004.

training examples. Extensive experiments have demonstrated that DECORATE constructs more accurate diverse ensembles than ADABOOST and Bagging when training data is limited, and does at least as well as ADABOOST when the training set is relatively large. By using artificial training data to construct diverse committees and prevent over-fitting, DECORATE has been shown to be a very effective ensemble meta-learner on a wide variety of data sets.

This paper explores the resilience of DECORATE to the various forms of imperfections in data. In our experiments, the training data is corrupted with missing features, and random errors in the values of both the category and the features. Results on a variety of UCI data demonstrate that, in general, DECORATE continues to improve on the accuracy of the base learner, despite the presence of each of the three forms of imperfections. Furthermore, DECORATE is clearly more robust to missing features than the other ensemble methods.

2 The DECORATE Algorithm

This section summarizes the DECORATE algorithm; for further details see [9,10]. The approach is motivated by the fact that combining the outputs of multiple classifiers is only useful if they disagree on some inputs [6]. We refer to the amount of disagreement as the *diversity* of the ensemble, which we measure as the probability that a random ensemble member's prediction on a random example will disagree with the prediction of the complete ensemble,

DECORATE was designed to use additional artificially-generated training data in order to generate highly diverse ensembles. An ensemble is generated iteratively, learning one new classifier at each iteration and adding it to the current ensemble. The ensemble is initialized with the classifier trained on the given data. The classifiers in each successive iteration are trained on the original data and also on some artificial data. In each iteration, a specified number of artificial training examples are generated based on a simple model of the data distribution. The category labels for these artificially generated training examples are chosen so as to differ maximally from the current ensemble's predictions. We refer to this artificial training set as the *diversity data.* We train a new classifier on the union of the original training data and the diversity data. If adding this new classifier to the current ensemble increases the ensemble training error, then this classifier is rejected, else it is added to the current ensemble. This process it repeated until the desired committee size is reached or a maximum number of iterations is exceeded.

The artificial data is constructed by randomly generating examples using an approximation of the training data distribution. For numeric attributes, a Gaussian distribution is determined by estimating the mean and standard deviation of the training set. For nominal attributes, the probability of occurrence of each distinct value is determined using Laplace estimates from the training data. Examples are then generated by randomly picking values for each feature based on these distributions, assuming attribute independence. In each iteration, the artificially generated examples are labeled based on the current ensemble. Given an

example, we compute the class membership probabilities predicted by the current ensemble, replacing zero probabilities with a small ϵ for smoothing. Labels are then sampled from this distribution, such that the probability of selecting a label is inversely proportional to the current ensemble's predictions.

3 Experimental Evaluation

3.1 Methodology

Three sets of experiments were conducted in order to compare the performance of ADABOOST, Bagging, DECORATE, and the base classifier, J48 [2], under varying amounts of three types of imperfections in the data:

1. **Missing features:** To introduce $N\%$ missing features to a data set of D instances, each of which has F features (excluding the class label), we select randomly with replacement $\frac{N \cdot D \cdot F}{100}$ instances and for each of them delete the value of a randomly chosen feature. Missing features were introduced to both the training and testing sets.
2. **Classification noise:** To introduce $N\%$ classification noise to a data set of D instances, we randomly select $\frac{N \cdot D}{100}$ instances with replacement and change their class labels to one of the *other* values chosen randomly with equal probability. Classification noise was introduced only to the training set and not to the test set.
3. **Feature noise:** To introduce $N\%$ feature noise to a data set of D instances, each of which has F features (excluding the class label), we randomly select with replacement $\frac{N \cdot D \cdot F}{100}$ instances and for each of them we change the value of a randomly selected feature. For nominal features, the new value is chosen randomly with equal probability from the set of *all* possible values. For numeric features, the new value is generated from a Normal distribution defined by the mean and the standard deviation of the given feature, which are estimated from the data set. Feature noise was introduced to both the training and testing sets.

In each set of experiments, ADABOOST, Bagging, DECORATE, and J48 were compared on 11 UCI data sets using the Weka implementations of these methods [17]. Table 1 presents some statistics about the data sets. The target ensemble size of the first three methods was set to 15. In the case of DECORATE, this size is only an upper bound on the size of the ensemble, and the algorithm may terminate with a smaller ensemble if the number of iterations exceeds the maximum limit. As in [9], this maximum limit was set to 50 iterations, and the number of artificially generated examples was equal to the training set size.

To ascertain that no ensemble method was being disadvantaged by the small ensemble size, we ran additional experiments on some datasets with the ensemble size set to 100. The trends of the results are similar to those with ensembles of

[2] J48 is a Java implementation of C4.5 [12] introduced in [17].

size 15. Details of these experiments are omitted here, but can be found in the extended version of this paper [11].

For each set of experiments, the performance of each of the learners was evaluated at increasing noise levels from 0% to 40% at 5% intervals using 10 complete 10-fold cross validations. In each 10-fold cross validation the data is partitioned into 10 subsets of equal size and the results are averaged over 10 runs. In each run, a distinct subset is used for testing, while the remaining instances are provided as training data.

To compare two learning algorithms across all domains we employ the statistics used in [16], namely the significant win/draw/loss record and the geometric mean error ratio. The win/draw/loss record presents three values, the number of data sets for which algorithm A obtained better, equal, or worse performance than algorithm B with respect to classification accuracy. A win or loss is only counted if the difference in accuracy is determined to be significant at the 0.05 level by a paired t-test.

The geometric mean (GM) error ratio is defined as $\sqrt[n]{\prod_{i=1}^{n} \frac{E_A}{E_B}}$, where E_A and E_B are the mean errors of algorithm A and B on the same domain. If the geometric mean error ratio is less than one it implies that algorithm A performs better than B, and vice versa. We compute error ratios to capture the degree to which algorithms out-perform each other in win or loss outcomes.

Table 1. Summary of Data Sets

Name	Cases	Classes	Attributes	
			Numeric	Nominal
autos	205	6	15	10
balance-scale	625	3	4	–
breast-w	699	2	9	–
colic	368	2	10	12
credit-a	690	2	6	9
glass	214	6	9	–
heart-c	303	2	8	5
hepatitis	155	2	6	13
iris	150	3	4	–
labor	57	2	8	8
lymph	148	4	–	18

3.2 Results

In this section, we only present the statistics summarized over all 11 datasets, For detailed tables of results, see the extended version of this paper [11].

Missing Features: The results of running the algorithms when missing features are introduced, are presented in Tables 2–4. Each table compares the accuracy of DECORATE versus another algorithm for increasing percentages of missing features.

Table 2. Missing Features: DECORATE vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	8/3/0	10/1/0	10/1/0	10/1/0	11/0/0	11/0/0	11/0/0	11/0/0	11/0/0
GM Error Ratio	0.8286	0.7882	0.7877	0.7815	0.7921	0.8039	0.8004	0.8095	0.8047

Table 3. Missing Features: DECORATE vs Bagging

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	2/7/2	2/8/1	4/7/0	3/7/1	5/5/1	4/7/0	4/7/0	5/5/1	8/3/0
GM Error Ratio	0.9520	0.9298	0.9201	0.9177	0.9041	0.9083	0.9085	0.9150	0.8882

Table 4. Missing Features: DECORATE vs ADABOOST

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	4/4/3	5/4/2	6/4/1	4/6/1	4/7/0	6/5/0	8/3/0	6/5/0	8/3/0
GM Error Ratio	0.9534	0.9382	0.9197	0.9024	0.9109	0.8982	0.8827	0.8968	0.8876

These results demonstrate that DECORATE is fairly robust to missing features, consistently beating the base learner, J48, at all noise levels (Table 2). In fact, when the amount of missing features is 20% or higher, DECORATE produces statistically significant wins over J48 on all datasets. The amount of error reduction produced by using DECORATE is also considerable, as is shown by the mean error ratios.

For this kind of imperfection in the data, in general, all of the ensemble methods produce some increase in accuracy over the base learner. However, the improvements brought about by using DECORATE are higher than those caused by both Bagging and ADABOOST. The amount of error reduction achieved by DECORATE also increases with greater amounts of missing features; as is clearly demonstrated by the GM error ratios.

Figure 1 (a) shows the results on a dataset that clearly demonstrates DECORATE's superior performance at all levels of missing features. In Figure 1(b), we see a dataset on which ADABOOST has the best performance when there are no missing features; but with increasing amounts of missing features, both Bagging and DECORATE outperform it.

The superior performance of DECORATE could be attributed to the fact that it adds artificial examples to the training set. These artificial examples do not contain any missing features, and are generated based on the distributions of features estimated over the visible (non-missing) values. A thorough analysis of how using artificial examples can increase robustness to missing features is an important subject for future research.

Classification Noise: The comparison of each ensemble method with the base learner, in the presence of classification noise are summarized in Tables 5-7. The tables provide summary statistics, as described above, for each of the noise levels considered.

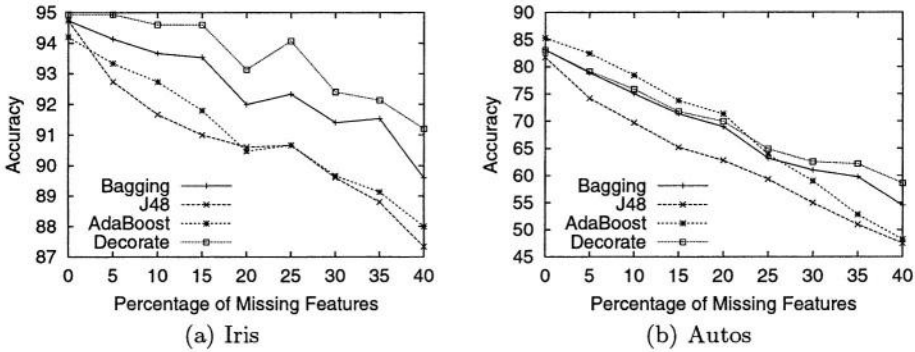


Fig. 1. Missing Features

The win/draw/loss records indicate that, both Bagging and DECORATE consistently outperform the base learner on most of the datasets at almost all noise levels; demonstrating that both are quite robust to classification noise. In the range of 10-35% of classification noise, Bagging performs a little better than DECORATE, as is seen from the error ratios. This is because, occasionally, the addition of noise helps Bagging, as was also observed in [3].

Unlike, Bagging and DECORATE, ADABOOST is very sensitive to noise in classifications. Though ADABOOST significantly outperforms J48 on 7 of the 11 datasets in the absence of noise, its performance degrades rapidly at noise levels as low as 10%. With 35-40% noise, ADABOOST performs significantly worse that the base learner on 7 of the datasets. Our results on the performance of ADABOOST agree with previously published studies [3,1,7]. As pointed out in these studies, ADABOOST degrades in performance because it tends to place a lot of weight on the noisy examples.

Figure 2(a) shows a dataset on which DECORATE has a clear advantage over other methods, at all levels of noise. Figure 2(b) presents a dataset on which Bagging outperforms the other methods at most noise levels. This figure also clearly demonstrates how rapidly the accuracy of ADABOOST can drop below that of the base learner. These results confirm that, in domains with noise in classifications, it is beneficial to use DECORATE or Bagging, but detrimental to apply ADABOOST.

Feature Noise: The results of running the algorithms with noise in the features are presented in Tables 8–10. Each table compares the accuracy of each ensemble method versus J48 for increasing amounts of feature noise.

In most cases, all ensemble methods improve on the accuracy of the base learner, at all levels of feature noise. Bagging performs a little better than the other methods, in terms of significant wins according to the win/draw/loss record. In general, all systems degrade in performance with added feature noise. The drop in accuracy of the ensemble methods seems to mirror that of the base learner, as can be seen in Figure 3. The performance of the ensemble methods seems to be tied to how well the base learner deals with feature noise.

Table 5. Class Noise: DECORATE vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	8/3/0	7/4/0	8/3/0	8/1/2	8/1/2	6/3/2	6/3/2	7/2/2	8/1/2
GM Error Ratio	0.8286	0.8398	0.8633	0.8734	0.8809	0.8960	0.9121	0.9229	0.8995

Table 6. Class Noise: Bagging vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	7/4/0	9/2/0	9/2/0	9/2/0	8/3/0	7/4/0	8/2/1	7/3/1	7/3/1
GM Error Ratio	0.8704	0.8687	0.8526	0.8508	0.8443	0.8719	0.8867	0.8972	0.8995

Table 7. Class Noise: ADABOOST vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	7/3/1	6/1/4	2/4/5	1/5/5	1/4/6	2/2/7	1/4/6	1/3/7	1/3/7
GM Error Ratio	0.8691	0.9930	1.0984	1.1604	1.2322	1.2242	1.2431	1.2120	1.1989

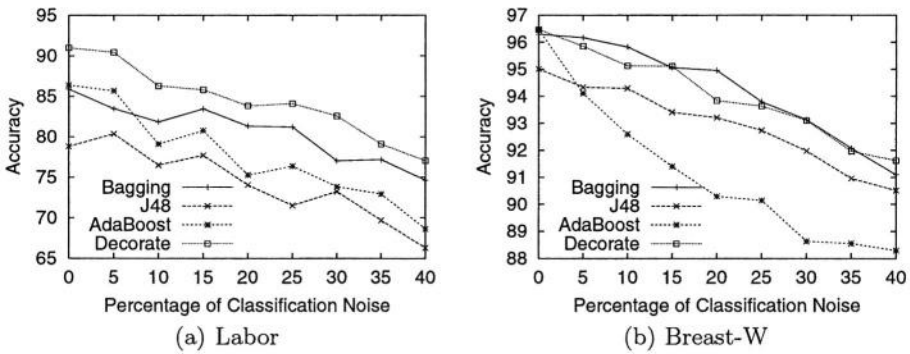


Fig. 2. Classification Noise

4 Related Work

Several previous studies have focused on exploring the performance of various ensemble methods in the presence of noise. A thorough comparison of Bagging, ADABOOST, and Randomization (a method for building a committee of decision trees, which randomly determine the split at each internal tree node) is presented in [3]. This study concludes that while ADABOOST outperforms Bagging and Randomization in settings where there is no noise, it performs significantly worse when classification noise is introduced.

Other studies have reached similar conclusions about AdaBoost [1,7], and several variations of AdaBoost have been developed to address this issue. For example, Kalai and Servedio [5] present a new boosting algorithm and prove that it can attain arbitrary accuracy when classification noise is present. Another algorithm, Smooth Boosting, that is proven to tolerate a combination of

Table 8. Feature Noise: DECORATE vs J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	8/3/0	7/4/0	7/4/0	8/3/0	7/3/1	7/4/0	6/5/0	7/4/0	6/5/0
GM Error Ratio	0.8286	0.8335	0.8434	0.8329	0.8593	0.8554	0.8690	0.8723	0.8782

Table 9. Feature Noise: Bagging vs. J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	7/4/0	10/1/0	10/1/0	10/1/0	10/1/0	8/3/0	10/1/0	10/1/0	10/1/0
GM Error Ratio	0.8704	0.8586	0.8450	0.8496	0.8473	0.8627	0.8634	0.8614	0.8661

Table 10. Feature Noise: ADABOOST vs. J48

Noise Level %	0	5	10	15	20	25	30	35	40
Sig. W/D/L	7/3/1	7/2/2	8/2/1	7/3/1	8/1/2	8/1/2	8/2/1	7/4/0	8/2/1
GM Error Ratio	0.8691	0.8449	0.8575	0.8455	0.8463	0.8564	0.8830	0.8900	0.8750

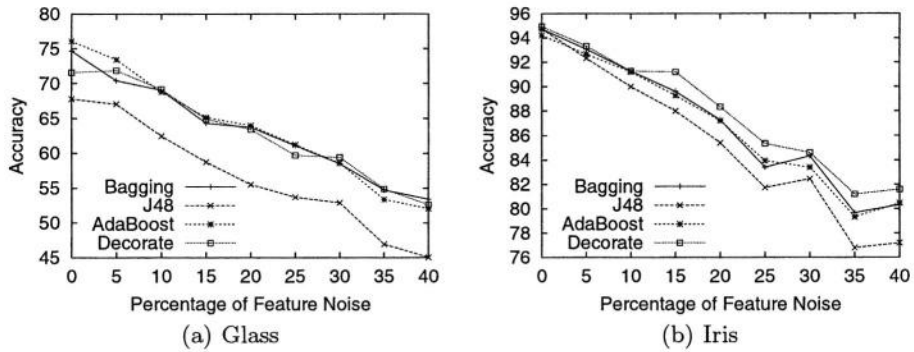


Fig. 3. Feature Noise

classification and feature noise is presented in [14]. McDonald et al. [8] compare ADABOOST to two other boosting algorithms – LogitBoost and BrownBoost – and conclude that BrownBoost is quite robust to noise. In an earlier study an extension to BrownBoost for multi-class problems was presented and shown empirically to outperform ADABOOST on noisy data [7]. However, BrownBoost's drawback is that it requires a time-out parameter to be set, which can be done only if the user can estimate the level of noise.

5 Future Work

Noise in training data chiefly contributes to an increase in the error due to variance of the base learner; and hence, variance-reduction techniques would be ideal to combat such noise. Bagging is a very effective variance reduction

method; whereas ADABOOST is primarily a bias reduction technique, though empirically it has shows to produce some reduction in variance as well [16]. In general, DECORATE also produces significantly more accurate classifiers than the base learner. We are currently investigating whether this improvement in accuracy is mainly due to a reduction in bias or variance. This should lend some more insight into DECORATE'S resilience to imperfections in data.

In our study, all the ensemble methods were used to generate committees of size 15. It may be beneficial to generate larger ensembles, so that the difference in performance between the systems is more pronounced.

An interesting avenue for future work would be to compare the performance of DECORATE and Bagging with the new boosting algorithms mentioned in Section 4. Another interesting subject for future experimentation is testing how the ensemble methods discussed in this study compare to noise elimination techniques such as the ones presented in [15].

6 Conclusion

This paper evaluates the performance of three ensemble methods, Bagging, ADABOOST and DECORATE, in the presence of different kinds of imperfections in the data. Experiments using J48 as the base learner, show that in the case of missing features DECORATE significantly outperforms the other approaches. In the case of classification noise, both DECORATE and Bagging are effective at decreasing the error of the base learner; whereas ADABOOST degrades rapidly in performance, often performing worse than J48. In general, Bagging performs the best at combating high amounts of classification noise. In the presence of noise in the features, all ensemble methods produce consistent improvements over the base learner.

Acknowledgments

Raymond J. Mooney and Prem Melville were supported by DARPA grants F30602-01-2-0571 and HR0011-04-1-007. Lilyana Mihalkova was supported by the MCD Fellowship from the University of Texas at Austin.

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36, 1999.
2. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2): 123–140, 1996.
3. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2): 139–157, 2000.
4. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In Lorenza Saitta, editor, *Proc. of 13th Intl. Conf. on Machine Learning (ICML-96).* Morgan Kaufmann, July 1996.

5. Adam Kalai and Rocco A. Servedio. Boosting in the presence of noise. In *Thirty-Fifth Annual ACM Symposium on Theory of Computing*, 2003.
6. A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In *Advances in Neural Information Processing Systems 7*, 1995.
7. Ross A. McDonald, Idris A. Eckley, and David J. Hand. A multi-class extension to the brownboost algorithm. Technical Report TR-03-14, Imperial College, London, 2003.
8. Ross A. McDonald, David J. Hand, and Idris A. Eckley. An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *Fourth International Workshop on Multiple Classifier Systems*, pages 35–44. Springer, 2003.
9. Prem Melville and Ray Mooney. Constructing diverse classifier ensembles using artificial training examples. In *Proc. of 18th Intl. Joint Conf. on Artificial Intelligence*, pages 505–510, Acapulco, Mexico, August 2003.
10. Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion: Special Issue on Diversity in Multiclassifier Systems*, 2004.
11. Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J. Mooney. Experiments on ensembles with missing and noisy data. Technical Report UT-AI-TR-04-310, University of Texas at Austin, 2004.
12. J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo,CA, 1993.
13. J. Ross Quinlan. Bagging, boosting, and C4.5. In *Proc. of 13th Natl. Conf. on Artificial Intelligence (AAAI-96)*, pages 725–730, Portland, OR, August 1996.
14. Rocco A. Servedio. Smooth boosting and learning with malicious noise. *The Journal of Machine Learning Research*, 4:633–648, 2003.
15. Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In *Fourth International Workshop on Multiple Classifier Systems*, pages 317–325. Springer, 2003.
16. G. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning,* 40, 2000.
17. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, San Francisco, 1999.

Induced Decision Fusion in Automated Sign Language Interpretation: Using ICA to Isolate the Underlying Components of Sign

David Windridge and Richard Bowden

Centre for Vision, Speech and Signal Processing
Dept. of Electronic & Electrical Engineering, University of Surrey
Guildford, GU2 5XH Surrey, United Kingdom
d.windridge@eim.surrey.ac.uk
Tel: +44 1483 876043

Abstract. We utilise the techniques of independent component analysis and principle component analysis to derive an independent set of gestural primitives for visual sign-language, employing existing sign linguistics as a reference point in the feature reduction.
In this way it is possible both to reduce (by several orders of magnitude) the requisite quantity of HMM computation involved in word classification, as well as to significantly improve performance through having transformed the initial classification problem into one of decision fusion. Moreover, the independent and optimally-compact representation of the gestural primitives ensures a maximum of classifier diversity prior to combination.

1 Introduction

The problem of sign language recognition has engendered considerable interest within the pattern-recognition/machine-learning community over recent years [eg 1-4], predominantly as a consequence of its unique interweaving of syntactic and image-processing concerns. Our interest in the problem falls both within these terms, as well as within the broader context of cognitive visual systems; the attempt, in essence, to 'reverse engineer' the human visual system. To this end, we are particularly interested in the nature of the feature-set most appropriate to gesture recognition *in so far as it reflects the signer's intention*, as much as we are concerned with optimising the classification performance of the recognition system. It is evident, however, the two goals are in no way mutually exclusive: indeed by encoding within the features' structure the way in which sign-language gestures are visually transcribed, we might expect to achieve a significant improvement in both the dimensional requirements of our pattern-space as well in the diversity and generalisability of the classifiers contained therein.

This design philosophy is then in contrast to the 'image-processing-led' approaches characterised by eg [1] and [3], that employ very large sets of well understood feature primitives (for instance the quantised position vectors and hand-shape descriptors of Starner and Pentland [1]). Such methods can, of course, be

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 303–313, 2004.

extremely effective, and indeed are broadly representative of the way in which pattern recognition is generally carried out; feature selection and classification being the ways in which the total body of problem information contained within the features are made usefully generalisable. However, much of the computational effort, danger of overclassification and loss of interpretability involved in this process might be avoided if there exists exploitable prior knowledge of gestural intention that can be straightforwardly encompassed by the structure of the features.

Fortunately, we have access to just such information in the field of gesture recognition by virtue of the existence of sign language dictionaries, which have evolved over a significant number of years as a means both of visually teaching new signs to the sign language user, as well as referencing unfamiliar signs from sequences of gestural primitives (see for instance [9]). Thus the diagrammatic dictionary descriptions represent, in a sense, both the ground truth of gestural intention behind signed word sequences, as well as the gesture descriptor's perceptual medians, these 'visemes' having been progressively developed for ease of recognition and compactness of description. Such descriptors are thus inherently independent of those issues, such as signing size and speed, that are irrelevant to gestural meaning, but which tend, nevertheless, to be intrinsically represented by feature-sets deriving from classical motion and image processing techniques.

It is then the individual and combined characteristics of these fundamental or 'intentional' components of gesture which we shall seek to capture in the following paper, the first aspect of which will be to provide a temporally-ordered series of binary descriptors denoting the presence or absence of the respective gestural viseme components for each word class. Once these are obtained, we will set out to reduce the redundancy inherent to the *visual* sign descriptions by establishing the underlying independent components of description, in effect the minimal description of gestural signification in information-theoretic terms. We do this via a combination of independent component analysis (ICA) and principle component analysis (PCA), the use of which is made uniquely possible by the partial dependency reduction already inherent in the dictionary descriptors.

It is hence *after* the independent temporal channels are established for each of the classes that the most important practical consequence of our technique makes itself apparent: if we are to go on to represent the temporal class sequences by Hidden Markov Models (HMMs), the independence of the feature channels implies that we can consider them on an *individual* basis. That is, rather than requiring that a single HMM encompass all possible transitions between observational states deriving from combinations of binary feature states (totalling 2^n for n channels), we need only consider n two-state HMMs per class, which, being independent, have their class likelihood statistics combined via multiplication. (Which is to say, the n HMMs require a decision fusion rule equivalent to the Product rule: however, we adopt the Sum Rule, and log-likelihood statistics for reason of maximal error robustness; see for instance [12]. In an error-free scenario these approaches are, of course, equivalent). Clearly, this transformation of a classification problem into a decision-fusion problem represents a very sig-

nificant reduction in computational complexity: of the order of $7 \times 10^{10} \rightarrow 72$ computational cycles for typical practical scenarios[1].

Another immediately beneficial consequence of the application of ICA is that, in terms of the consequent orthogonalisation of the individual HMM outputs, the classifier diversity significantly improves: an attempt to quantify this effect will thus constitute the initial experimental component of this paper. A third benefit derives from applying PCA in *conjunction* with ICA: it becomes possible to reduce the problem dimensionality through a prior determination of the actual number of underlying independent factors relevant to classification. In the outlined experimental scenario utilising this option will act to reduce the feature-space dimensionality from 21 to 18, a 17 percent improvement in the immediate processing requirements.

In essence, then, the utilisation of ICA and PCA enables us to make maximally efficient use of the feature 'bandwidth' for the purposes of classification, the resultant features, it is anticipated, corresponding to the basic components of gestural signification. Being independent, we are thus required to re-evaluate the problem as one of fusion of maximally diverse classifier outputs.

In terms of organisation, the paper will hence commence with a description of the dictionary-derived feature-set, along with a precis of the underlying motivation for this type of viseme-based approach. Following this is a brief treatment of ICA analysis, comparing and contrasting it with its more ubiquitous relation, PCA. Thereafter we will outline a simple experiment to quantify the relative utility of a composite ICA/PCA treatment in terms of the effect on class separation in the HMM output space, as well as on the overall classification performance in the context of Sum-Rule decision fusion.

2 Viseme-Based Feature Descriptors: Utilising a Pre-existing Pictorial Grammar

2.1 Nature of Features

We are, in the current investigation, concerned specifically with British Sign Language (BSL), which is classed by linguists as a topic-comment language (for instance, the English query 'What is your name?' would be most nearly rendered in BSL as the syntactic sequence 'Name: you: what'). To this specifically gesture-syntactic language-form there is added a parallel finger-spelling component for exact English transliteration. Our long term goal being to provide a machine-translation that captures both of these aspects of the language, we

[1] Obviously, an *effective* treatment of the composite binary feature need not be as complex as indicated: non-ergodicity and vector-quantisation can be exploited to significantly decrease the computational work-load. However, it would necessarily still be several orders of magnitude greater than that implied by the independent approach. Furthermore, since much of the class variability derives from minor asynchronicities between the feature channels, a fully independent treatment means that significantly fewer 'overlap' transition states need be quantified.

require that any proposed gestural feature-set must lend itself to treatment in terms of grammarial context as well as letter-by-letter or word-by-word classification (it is, for instance, the case in BSL that facial expression can act as a modifier of hand gesture, or that differing positions of signing relative to the body can indicate third-person narrative). Thus it is necessary, if we are to retain this extensibility, to isolate *independent* 'gestural meaning' components as far as possible, an objective that will ultimately motivate our use of ICA/PCA to extract the absolutely basic gesture features for classification.

Our approach, then, attempts to describe events via a set of 2-D lexical features, a key rational being that if the features *already* naturally generalise about events within the image, then very much less training data is required to provide accurate classification. In a manner somewhat akin to Volger *et al*'s [11] we thus attempt to describe the constituents of sign at a component level, via a decomposition of the various gesture sequences into visemes (the constituent visual components of sign). The actual constitution of these visemes is derived from linguistical studies of sign and the consequent notation systems they have evolved to catalogue sign vocabularies. Volger *et al* thus sought to recognise the 22 component phonemes of American sign-language dictionary notation by utilising a distinct HMM for each phoneme in conjunction with a large set of training vectors. Our approach, on the other hand, attempts to coarsely describe sign events via a similar 'Ha-Tab-Sig' notation [9], but differs critically in that no prior training is required to determine feature membership, our rule-based body-motion descriptors being sufficiently robust to take these as given. Thus, in a sense, Volger *et al*'s classification *output* is our feature-set *input.* The particular mechanism by which this body-feature description is enacted consists in the dynamic fitting of a 2D contour representing the head and shoulders. Specifically, the 18 connected points constituting the contour have their local edge strengths computed along normals in order to track the head and shoulder movement. From the position, scale and orientation of the contour we then estimate approximate location and sizes of the key body parts (for instance, the shoulders, hips, chest, stomach, forehead, chin etc.).

We thus have a method of describing the position and motion of a signer's gestures which is largely independent of individual body morphology and camera set-up. Crucially, it also provides a description that naturally generalises, consequently reducing the training requirements. Linguistic evidence further indicates that sign recognition is primarily performed upon the dominant hand (which conveys the majority of information), and consequently we currently discard the non dominant hand in order concatenate the HA, TAB, SIG features together to produce a 33 dimensional vector describing the viseme component of a single frame of video.

2.2 Intelligibility Considerations

While this dictionary-based feature-set has evolved towards some criterion of optimality through essentially heuristic reasoning acting over a number of years, it does not, as we have indicated, necessarily represent the most compact or

independent set of features (being limited chiefly by physical co-articulation restraints); hence our motivation for proposing ICA of the gestural time sequence, independence critically allowing us to utilise *separate* HMMs for distinct feature channels. It does, however, represent a readily comprehensible feature set. Of some concern to us, therefore, is the question of what implication ICA has for the intelligibility of features so transformed: we should not, in particular, wish to gain computational efficiency at the expense of comprehensibility if the eventuality might be avoided.

In assessing just how far intelligibility is conserved, it is of some advantage to have commenced with a readily-intelligible feature prior set to ICA transformation: it would not be straightforward to derive a comprehensible gesture syntax from (say) a set of vector-quantised Voronoi cell transitions. In having obtained an initial feature set that broadly corresponds to the underlying visual components of gestural intention (such that, for instance, we could learn to emulate without having to refer to an actual signer), we have in fact nearly optimally isolated feature components in intelligibility terms. However, because this isolation has not been carried out on the basis of mutual independence, there exists an inevitable superfluity of description in the viseme-based feature-set, manifesting itself as a dependency among the gestural components of the type that ICA immediately removes: critically for our intelligibility concerns, however, it would do so explicitly *in terms* of the intelligible components. Thus, to give a purely illustrative example, if a gesture-dictionary feature encoding were to contain a time sequence of relative (x, y) positions for both right and left hands, and if it were to transpire that, throughout the entirety of the dictionary, there existed an exact mirror-symmetry between the left and right hand gestures for each word, then an ICA+PCA reduction of the original feature space coordinatisation:

$$(\mathrm{Lefthand}_{x\_\mathrm{position}}, \mathrm{Lefthand}_{y\_\mathrm{position}}, \mathrm{Righthand}_{x\_\mathrm{position}}, \mathrm{Righthand}_{y\_\mathrm{position}}),$$

would produce a set of three fully independent coordinates:

$$(\mathrm{Hand's\_Midpoint}_{x\_\mathrm{position}}, \mathrm{Hand's\_Midpoint}_{y\_\mathrm{position}}, \mathrm{Inter\_hand\_distance}).$$

Of course, it is by no means guaranteed in general that the ICA+PCA reduction would be so transparently comprehensible, but very often an intuitive understanding of the transformed components can be forged by virtue of the intelligibility inherent in the original component descriptors. Thus our newly discretized classifier set broadly corresponds to comprehensible aspects of sign, and consequently, decision-fusion of their outputs is correlated with a visualisable distinction between alternative gestural patterns.

3 ICA verses PCA for Gestural Analysis

3.1 Description of ICA and PCA Methodologies

Principle Component Analysis (PCA) has long been a tool in the pattern recognition researcher's toolkit as a means of reducing pattern-space dimensionality

while retaining class separability. It does so by providing a linear transformation of pattern vectors that maximises the retained variance with respect to the subspace dimensionality, thereby preserving class discriminant information and prioritising (via the eigenvector eigenvalues) the translated feature axes.

As such, PCA methods will, in general, find truly independent (as opposed to decorrelated) feature axes for only a small subset of pattern-vector distributions. ICA, in contrast, specifically seeks independent subspace orientations in the data without retaining variance information (components essentially being normalised). Thus ICA can be considered a factor analysis for multivariate data where the mixing system is initially unknown. As a tool for data *interpretation*, ICA is thus much more useful than PCA, and on occasion, capable of greatly improving the overall classification performance.

We shall give a very brief overview of the former process as follows (with more detailed and general treatments being available in, for instance, [7] and [8]): Let $\boldsymbol{t}$ be a two-dimensional binary vector describing the state of the 21 individual feature channels over the complete temporal range, t. We wish to describe this vector in terms of the fully independent feature-set vector $\boldsymbol{t'}$, of presently unknown channel size. The two vector quantities are related by a mixing matrix, M, thus: $\boldsymbol{t'} = M\boldsymbol{t}$. The determination of the matrix M is thus the objective of our calculation, the first stage of which is the decorrelating (or 'whitening', 'sphering') of the original input space. That is, we shall require a linear transformation $\boldsymbol{t}_w = W\boldsymbol{t'}$ such that the expectation $E(\boldsymbol{t}_w \boldsymbol{t}_w{}^T)$ is equal to I (I being the identity matrix). A simple solution to this constraint exists via the expansion:

$$I = E(\boldsymbol{t}_w \boldsymbol{t}_w{}^T) = E(W\boldsymbol{t'}[W\boldsymbol{t'}]^T) = E(W\boldsymbol{t'}\boldsymbol{t'}{}^T W^T) = E(W[\boldsymbol{t'}\boldsymbol{t'}{}^T]W^T) \quad (1)$$

Setting $S = E(\boldsymbol{t'}\boldsymbol{t'}{}^T)$, we observe that $W = S^{\frac{1}{2}}$ fulfils the terms of the constraint, the last term in the equation becoming $S^{\frac{1}{2}} S S^{\frac{1}{2}} (= I)$. Having found a suitable W, it only remains to perform a rotation of the whitened pattern-space axes such that the non-Gaussianity of the probability distribution of the individual variates of the transformed space is maximised (linear mixtures of variates being invariably more Gaussian than their components via the central limit theorem). This is usually achieved via an appropriate statistical, information theoretic or morphological measure of non-Gaussianity, and any one of a number of algorithms for finding global maxima/minima.

3.2 Implication of ICA and PCA for Classification

The practical distinction between the two component analysis methods can, in the context of classification, be rather subtle, and in general the unmodified ICA approach is only found to be useful when distinguishing classes with a high degree of stochastic independence (for instance, the star/cosmic ray astronomical distinctions of [10]), or else where classes are composed of distinct sets of independent features (for instance, the texture classifications of [5], where the array of potential class sub-textures has a visual aspect somewhat akin to two-dimensional Fourier components). As a consequence of the normalising properties of ICA detailed earlier, subtle feature independences can be very much

amplified: whether or not this is useful for classification, though, tends to be situation dependent.

Thus, a heuristic rule for advocating the implementation of ICA over PCA for the purposes of classification, might read 'employ PCA where class discriminant information is contained within small numbers of correlated features, but employ ICA when class discriminant information is contained within independent features composed of large numbers of correlated components, and the isolation of those components is crucial to class interpretability'. Note it is also possible to perform PCA *prior to* ICA where we have reason to suppose the existence of fewer significant independent components than the pre-specified dimensionality of our feature-space by imposing an eigenvalue ordering on the ICA components.

It is precisely the latter situation that we expect to present itself in relation to our dictionary-derived feature vectors: for instance (and here we very much simplify), it might transpire that left and right hand positions are highly correlated with arm and shoulder positions, while being completely independent of each other; or finger positions might be independent of each other, but correlated with hand positions, and so on. Thus the gestural channels through which information is conveyed (hands, fingers, etc), are physically bound to various other body articulations in a way that must of necessity be depicted in a sign dictionary, but which may in fact be superfluous to gesturing *intent.* By performing PCA as an initial thresholding mechanism and *subsequent* ICA, it is thus possible extract the minimal subset of gestural indicators.

When the *temporality* of these gestural sequences is additionally considered, the independence of the components has a crucial bearing on how intelligibly the modifications of meaning imposed by grammatical *context* are encompassed: fully independent components will contain only *single* context modifiers in individual channels (a fact exploited through our combination of HMM channel likelihoods in such a way that additional context information can be added as the technique develops).

4 Experimental Implementation and Findings

In deriving the ICA transformation matrix for our initial feature data-set, we have thus firstly to concatenate all 232 word instances in the training data base (which would ideally consist of large number of actual word sequences, such that multiple word instances are represented at their correct relative probability of occurrence). Thus we obtain, for our limited case, a 21×3640 data matrix describing the 3640 temporal states of the 21 feature 'channels' (there being an immediate redundancy of 9 features for the selected training vectors): see fig. 1 for an indication of the format of the feature-vectors. ICA itself is carried out via the negentropy minimisation algorithm developed at the Helsinki University of Technology [6], which also performs a PCA-based assessment of feature redundancy. In this way, the resulting transformation matrix converts our original 21 feature-channels to a series of 18 independent feature components. We shall seek to quantify the improvement in feature diversity implied by this transformation

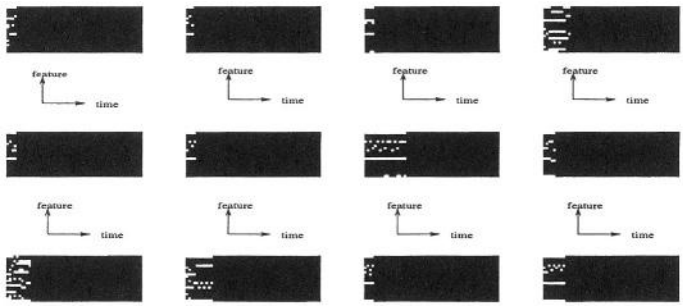


Fig. 1. Untransformed gestural feature sequence for first 12 words in the database

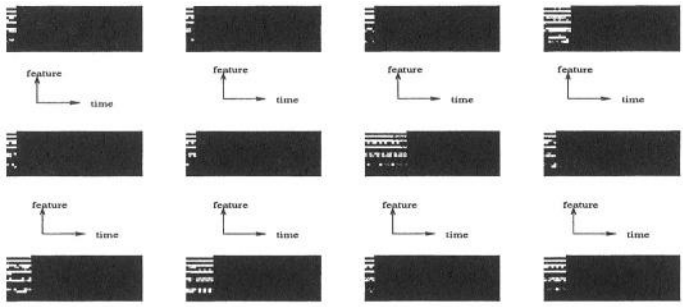


Fig. 2. Transformed gestural feature sequence for first 12 words in the database with grey scale values in the range –2 : 2 (note increase in information density over fig. 1)

of temporal feature information in terms of the average Mahalanobis distance between classifier outputs in the transformed and untransformed classes, respectively. The actual attribution of class probabilities is carried-out via HMM modelling in the following manner: The post-ICA feature-set (figure 2) now contains 18 independent temporal feature sequences, with a *continuous* range of values in the interval –2 : 2 (recall the the features were formerly quantised as the binary digits 1 and 0, representing presence and absence respectively). It is consequently necessary, in producing a meaningful comparison between the classification abilities of the ICA-transformed features and those that have not undergone the process, that we represent the former in a similarly quantised fashion (that is, with an equivalent number of hidden and observable HMM states). The number of observables in the untransformed feature-set being two thus requires that we map the continuous interval (–2 : 2) to the discrete digits 0 and 1: we choose to do this in the most symmetric fashion by allocating negative values to 0 and positive values to 1 (which would, of course, imply a loss of information, such that the untransformed pattern vectors could not now be recovered from the ICA features: in general this is not a conceptually serious omission, being roughly the equivalent of retaining phase information at the

expense of amplitude information in a Fourier transformation, which has only marginal consequences for signal recovery unlike its converse).

We have now to select the number of hidden states per HMM required to describe the range of behaviours implicit in the transformed and untransformed features. Empirical evidence suggests that four are sufficient. Thus for the 18 temporal channels of the transformed data and the 21 channels of the untransformed data, we train a series of HMMs of 4 hidden and 2 observational states each, one for each of the classes in order to provide a comparable test environment for the transformed and untransformed data.

To allocate a class probability to a particular observed gesture sequence, we have therefore to utilise the multiple log-likelihood outputs of the trained class HMMs; respectively 18 or 21 separate HMMs per class. Uniquely for the ICA-transformed features it is possible to sum these channel likelihoods together to obtain a class *probability.* With respect to the untransformed features, however, we may still sum these likelihoods together to obtain an overall *measure* of class likelihood, but the lack of independence implies that this is no longer strictly a probability. (The only truly stochastic way to treat the untransformed channels, would be, as indicated earlier, to concatenate the channels together to represent a single observable state, requiring a single HMM of 2^{21} observable states (assuming ergodicity) rather than the 21 HMMs of just 2 observable states employed by our comparison statistic; clearly a computational impossibility). It is thus only the prior application of ICA that *requires* the use of decision fusion.

Having established a comparable performance measure for both the transformed and untransformed classes, we can, prior to computation of these quantities, obtain a preliminary quantification of the relative value of the two approaches by considering the ensemble average inter-class Mahalanobis distance between every pair of class centroids in the total probability space deriving from the various HMM likelihoods.

In our limited training set, for which we have a total of 3640 pattern vectors representing 115 distinct classes, the Mahalanobis distance is thus computed from an average of 31.7 pattern vectors ensemble-averaged over ${}^{115}C_2 = 6555$ possible class pairs: the mean ensemble distances so derived are 4.7424 for the transformed and 0.8583 for the untransformed features. Thus we have, aside from the other benefits achieved, obtained a 6-fold improvement in the class differentiability as a consequence of the ICA/PCA transformation. The result of Sum-Rule decision fusion for the transformed and untransformed data reflects this discrepancy at a more modest level, where we obtain classification rates of 63.2% and 59.0%, respectively. Clearly a more complex decision fusion scheme such as a weighted summation or clustering approach could potentially improve upon this, and remains for future work.

One caveat that we should issue in relation to this preliminary performance assessment is that the word lengths, being highly variable and subject to 'nesting' difficulties, require that a range of windows of varying temporal length be simultaneously considered in order to return a probability statistic for all of the classes. Thus, we are still required to perform classical Viterbi sorting in order

to perform real-time sign-language processing. In future publications we shall consequently aim to provide a more detailed performance study of the implications of PCA and ICA methods for gesture-recognition with this consideration taken explicitly into account. We do not, however, expect that such contextual issues will significantly modify our findings with regard to class-separability: the existing argument for ICA in terms of the reduction in HMM computational requirement is, of course, completely unaffected.

5 Conclusions

We have instigated a programme to isolate the fundamental components of gestural intent by referring to existing sign dictionaries for the construction of an appropriate feature-set. To eliminate redundancies in this dictionary-based description, we have further employed the techniques of ICA and PCA to derive a minimal set of independent gestural components. In doing so, we have necessitated the implementation of a decision fusion framework, and consequently been able to reduce the HMM computational requirement for word-recognition by several orders of magnitude, as well to significantly enhance classifier diversity. Additionally, we have allowed for further extensibility of the feature set, and permitted grammatical context to be straightforwardly included within the system design, an endeavour that will, we anticipate, reach fruition in a future context-sensitive implementation of the viseme-based gesture recognition system.

Acknowledgment

This work is funded under the EC Cognitive Vision Systems framework.

References

1. T. Starner, A. Pentland, Visual recognition of ASL using hidden Markov models, Proc. Int. Workshop Autom. Face Gesture Recogn., Zürich 1995, p189-194.
2. Christopher Lee, Yangsheng Xu. "Online, Interactive Learning of Gestures for Human/Robot Interfaces." 1996 IEEE International Conference on Robotics and Automation, Minneapolis, MN. vol. 4, pp 2982-2987.
3. Bowden R, Sarhadi M, A non-linear Model of Shape and Motion for Tracking Finger Spelt ASL, Image and Vision Computing, vol 20/9-10, pp597, Aug 2002, Elsevier.
4. S. Gibet, A. Braffort, T. Lebourque, F. Forest, R. Gherbi, C. Collet and P. Bourdot, "Gesture in Human-Machine Communication", Proc. of Gesture Workshop '96, Springer-Verlag London Limited 1997, ISBN 3-540-76094-6.
5. R. Manduchi, J. Portilla, "Independent Component Analysis of Textures", Proc. of the IEEE Intern. Conference on Computer Vision, Kerkyra, Greece, Sept. 1999.
6. Erkki Oja, Aapo Hyvarinen, Juha Karhunen, "Independent Component Analysis", John Wiley & Sons Inc, 2001, ISBN: 047140540X
7. A. Hyvärinen, E. Oja, "Independent Component Analysis: Algorithms and Applications", Neural Networks, 13(4-5):411-430, 2000.

8. P. Comon, "Independent component analysis, A new concept?," IEEE Signal Processing Mag., vol. 36, no. 3, 1994.
9. C. Smith, "A Guide to B.S.L.", Souvenir Press Ltd, 1990; ISBN 0285650831.
10. Funaro M., "Analysis of Astrophysical Image Data by PCA/ICA', European Meeting on I.C.A., Salerno University, 2002, Vietri sul Mare (SA), Italy.
11. Vogler, C., Metaxas, D.: Towards Scalability in ASL Recognition: Breaking Down Signs into Phonemes. Gesture Workshop'99, March 1999, Gif-sur-Yvette, France.
12. J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, "On combining classifiers", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, 1998, 226-239.

Ensembles of Classifiers Derived from Multiple Prototypes and Their Application to Handwriting Recognition

Simon Günter and Horst Bunke

Department of Computer Science, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
sguenter,bunke@iam.unibe.ch

Abstract. There are many examples of classification problems in the literature where multiple classifier systems increase the performance over single classifiers. Normally one of the following two approaches is used to create a multiple classifier system: 1. Several classifiers are developed completely independent of each other and combined in a last step. 2. Several classifiers are created out of one prototype classifier by using so called classifier ensemble methods. In this paper a novel algorithm which combines both approaches is introduced. This new algorithm is experimentally evaluated in the context of hidden Markov model (HMM) based handwritten word recognizers and compared to previously introduced methods which also combine both approaches.

Keywords: Handwriting Recognition; Hidden Markov Model (HMM); Multiple Classifier System; Ensemble Method.

1 Introduction

The field of off-line handwriting recognition has been a topic of intensive research for many years. First only the recognition of isolated handwritten characters was investigated [25], but later whole words [24] were addressed. Most of the systems reported in the literature until today consider constrained recognition problems based on vocabularies from specific domains, e.g. the recognition of handwritten check amounts [13] or postal addresses [14]. Free handwriting recognition, without domain specific constraints and large vocabularies, was addressed only recently in a few papers [15, 21]. The recognition rate of such systems is still low, and there is a need to improve it.

The combination of multiple classifiers was shown to be suitable for improving the recognition performance in difficult classification problems [18, 28]. Also in handwriting recognition, classifier combination has been applied. Examples are given in [2,19, 29]. Recently new ensemble creation methods have been proposed in the field of machine learning, which generate an ensemble of classifiers from a single classifier [4]. Given a single classifier, the base classifier, a set of classifiers can be generated by changing the training set [3], the input features [11], the input data by injecting randomness [6], or the parameters and the architecture

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 314–323, 2004.

of the classifier [22]. Another possibility is to change the classification task from a multi-class to many two-class problems [5]. Examples of widely used methods that change the training set are Bagging [3] and AdaBoost [7]. Random subspace method [11] is a well-known approach based on changing the input features. A summary of ensemble creation methods is provided in [4]. In the present paper we focus on Bagging, random subspace method and a version of AdaBoost for multi-class problems, AdaBoost.M1 [7].

One common feature of the ensemble creation methods discussed above is the fact that they all start from a single classifier to derive an ensemble. In [8] a more general approach was proposed where we initially consider a set of classifier prototypes and separately apply an ensemble method to each of the prototypes. The final ensemble is then constructed by fusing all ensembles. The contribution of the present paper is twofold. First, the ensemble methods introduced in [8] are applied in conjunction with significantly improved HMM-based recognizers. Second, a new algorithm for ensemble generation is introduced and evaluated. This algorithm takes explicit advantage of the diversity of prototype classifiers.

The rest of this paper is organized as follows. In Section 2, the method for classifier generation, which starts from a set of prototypes, rather than a single base classifier, is described. In Section 3 the new algorithm which takes explicit advantage of the diversity of the prototype classifiers is described. The prototype classifiers for handwriting recognition used in the experiments are presented in Section 4. Then, in Section 5, results of experiments are reported. Finally, some conclusions are drawn in Section 6.

2 Creation of Ensembles from Sets of Prototypical Classifiers

The method used in this paper for creating ensembles from sets of prototypical classifiers was introduced in [8] and is shortly described in this section. The underlying idea is very simple. Rather than starting with a single classifier, as it is done, for example, in Bagging, AdaBoost and the random subspace method, we initially consider a set of classifiers (called prototypes in the following) and use an ensemble method to generate an ensemble out of each individual prototype. Then we merge all classifiers of these ensembles to get a single ensemble. For further details see [8].

An issue that needs to be addressed when implementing the method sketched before is the generation of the initial prototype classifiers $C_1, \ldots, C_n$. Sometimes different classifiers for the same task may already exist. In the experiment of this paper we use HMM classifiers with different architecture and different input features as prototypes.

3 Multi-probabilistic Boosting

The new algorithm is based on the simple probabilistic boosting (SPB) method introduced in [9]. The SPB algorithm works with a distribution d of weights

over the training set, similarly to AdaBoost [7]. Here we interpret the weight $d(x)$ of an element x of the original training set T as the probability of being selected when sampling elements for the training set of the next classifier of the ensemble. The main idea of the SPB algorithm is to set the weight $d(x)$ of a training element x proportional to the probability $e(x)$ that the ensemble of classifiers will misclassify elements similar to element x.

Similarly to AdaBoost, the weights of "hard" elements, i.e. elements which are likely to be misclassified, are set to a high value. It was decided to make the weights linearly dependent on the misclassification probability, because this approach is simple and the optimal function $o : e(x) \rightarrow d(x)$ is unknown anyway. The question remains how to calculate $e(x)$ from the results of the classifiers $C_1, \ldots, C_m$ that were already created. Four different functions were considered in [9]. In this paper only one function, $E(x)$, will be used[1], as it produced the best results in previous experiments. The error function $E(x)$ is defined in the following equation:

$$E(x) = \begin{cases} 0 & \text{if the voting result of the ensemble for } x \text{ is correct} \\ 1 - \frac{k(x)}{m} & \text{otherwise} \end{cases} \tag{1}$$

In this equation $k(x)$ is the number of correct classifiers for pattern x and m is the total number of classifiers in the ensemble.

The novel multi probabilistic boosting (MPB)[2] algorithm to be introduced in this paper, is an extension of SPB for the case of several prototypes. In contrast to the SPB algorithm, and also to AdaBoost, we have in MPB a distribution d of weights over the training set for each prototype. The algorithm produces one classifier for each prototype per step where the distribution corresponding to the actual prototype is used. The weight of the distribution of the $i-$th prototype is set according to the following equations

$$(\sum_i d_i(x)) \propto e(x) \tag{2}$$

$$d_i(x) = (\sum_i d_i(x)) \cdot \frac{(1 - e_i(x))}{\sum_j (1 - e_j(x))} \tag{3}$$

In the equation, $e(x)$ is the probability that the ensemble consisting of all classifiers produced in the previous steps misclassifies x, and $e_i(x)$ is the probability of a wrong result for pattern x being produced by the ensemble consisting only of the classifiers produced from the $i - th$ prototype. If $\sum_j (1 - e_j(x)) = 0$, i.e. the error probability is 1 for all prototypes, then all $d_i(x)$ are set to the same value (i.e. $\forall_{i,j}(d_i(x) = d_j(x))$). In the experiments of this paper always the error function $E(x)$ described above was used.

In this algorithm the prototypes which are likely to correctly recognize element x receive higher weights than prototypes whose classifiers often misclassified the pattern. The reason behind this kind of weight assignment is that we

[1] The function $E(x)$ was denoted by $e_2(x)$ in [9].

[2] The first word 'multi" refers to the fact that MPB works with multiple prototypes.

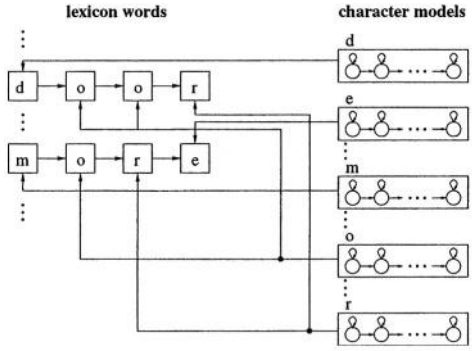


Fig. 1. Concatenation of character models yields the word models

want each prototype to "focus" on those elements for which its chances of successful recognition are high. We also notice that the weight is proportional to the error probability of the ensemble consisting of all classifiers produced in the previous steps. This means that the weight is especially high for patterns which are likely to be misclassified by the whole ensemble and for which the actual prototype is likely to classify them correctly.

4 Handwritten Text Recognizer

In Section 5 two sets of experiments are described. In the first set of experiments two HMM-based handwritten word recognizers are used. These two recognizers, which will be called C_1 and C_2 in the following, are similar to the one described in [21]. We assume that each handwritten word input to the recognizers has been normalized with respect to slant, skew, baseline location and height (for details of the normalization procedures see [21]). A sliding window of one pixel width is moved from left to right over the word and nine geometric features are extracted at each position of the window. Thus an input word is converted into a sequence of feature vectors in a 9-dimensional feature space. After the extraction of a feature vector the window is shifted by one pixel, i.e. the number of extracted feature vectors is the same as the width of the word in pixels. The geometric features used in the system Include the fraction of black pixels in the window, the center of gravity, and the second order moment. These features characterize the window from the global point of view. The other features give additional information. They represent the position of the upper- and lowermost pixel, the contour direction at the position of the upper- and lowermost pixel[3], the number of black-to-white transitions in the window, and the fraction of black pixels between the upper- and lowermost black pixel. In [21] a more detailed description of the feature extraction procedures can be found.

[3] To compute the contour direction, the windows to the left and to the right of the actual window are used.

For each uppercase and lowercase character, an HMM is build. For all HMMs the linear topology is used, i.e. there are only two transitions per state, one to itself and one to the next state. To model entire words, the character models are concatenated with each other. Thus a recognition network is obtained (see Fig. 1). This network exactly represents the set of words included in the underlying dictionary. Note that the network doesn't include any contextual knowledge on the character level, i.e., the model of a character is independent of its left and right neighbor. There is exactly one model for each word from the underlying dictionary. This approach makes it possible to share training data across different words. That is, each word in the training set containing character x contributes to the training of the model of x. Thus the words in the training set are more intensively utilized than in the case where an individual model is build for each word as a whole, and characters are not shared across different models. One important advantage of using HMMs on the word level is that the segmentation of the words into characters is done automatically by the Viterbi recognition algorithm [23].

The feature distributions in each state of an HMM are modeled by single Gaussians and four iterations of the Baum-Welch algorithm [23] are used for the training of the classifiers. The two classifiers, C_1 and C_2, differ in the way the number of states is determined for each individual character. For C_1, the Quantile method [31] and for C_2 the Bakis method [31] was used.

In the second set of experiments described in Section 5 two more sophisticated classifiers were used. The first classifier, C_3, is an optimized version of classifier C_1. For classifier C_3 the distribution of the features in each state of an HMM is modeled by a Gaussian mixture instead of a single Gaussian. The training method of the classifier, which involves the determination of the number of Gaussians in each state and the number of training iterations, was optimized on a validation set, using a strategy described in [10]. The other classifier, C_4, is a modified version of the classifier presented in [26, 27]. Classifier C_4 uses a sliding window for feature extraction were the window width is 16. Also for this classifier the window is shifted only by one pixel after the extraction of a feature vector. The window is partitioned into 16 cells arranged in a 4×4 grid. The average grey value of the pixels of each cell is used as a feature. A Karhunen-Loeve transformation [16] is then applied to the feature vectors and only the first 14 components of the transformed feature vectors are used. Classifier C_4 also uses a training method optimized by a strategy presented in [10], and models the distribution of the features by Gaussian mixtures. In addition the number of states in each HMM is optimized by the Quantile method introduced in [31].

The implementation of all systems is based on the Hidden Markov Model Toolkit (HTK), which was originally developed for speech recognition [30]. This software tool employs the Baum-Welch algorithm for training and the Viterbi algorithm for recognition [23]. The output of each HMM classifier is the word with the highest rank among all word models together with its score value.

5 Experiments

For isolated character and digit recognition, a number of commonly used databases exist. However, for the task considered in this paper, there exists only one suitable database to the knowledge of the authors, holding a sufficiently large number of words produced by different writers [20]. Consequently this database was used in the experiments.

Two sets of experiments were done. In the first set classifiers C_1 and C_2 were used and the number of classifiers per ensemble was fixed to 10. In the second set of experiments classifiers C_3 and C_4 were used. For this set of experiments the optimal number of classifiers was determined in a separate experiment.

To combine the individual classifiers of the ensembles, the following combination schemes were applied:

1. Voting (*voting*): Only the top choice of each classifier is considered. The word class that is most often on the first rank is the output of the combined classifier. Ties are broken by means of the maximum rule, which is only applied to the competing word classes. The maximum rule decides for the word class with the highest score among all word classes and all classifiers.
2. Weighted voting (*perf. v.*): Here we consider again the top class of each classifier. In contrast with regular voting, a weight is assigned to each classifier. The weight is equal to the classifier's performance (i.e. recognition rate) on the training set. The output of the combined classifier is the word class that receives the largest sum of weights.
3. GA weighted voting (*ga v.*): This combination scheme is similar to weighted voting, but the optimal weights are calculated by a genetic algorithm based on the results of the classifiers achieved on the training set.

In the first set of experiments a data set of 10,927 words with a vocabulary of size 2,296 was used. That is, a classification problem with 2,296 different classes was considered. The total number of writers who contributed to this set is 81. Prototype classifiers, C_1 and C_2, as described in Section 4 were used.

A training set containing 9,861 words and a test set containing 1,066 words were chosen in such a way that none of the writers of the test set were represented in the training set. So the experiments are writer independent. The recognition rate in this experiment was 70.92 % for prototype C_1, and 70.71 % for prototype C_2. The results of the first set of experiments are shown in Table 1. The ensemble method is indicated in the column *algorithm.* The entries in column *C* denote the classifiers used in the experiment. If there is only one classifier then the normal ensemble method was applied. If the entry contains both classifiers then the algorithm described in Section 2 was used. In this case five classifiers were generated from each prototype C_1 and C_2. The number of features for the random subspace method was set to six.

First we focus on rows 1-9 in Table 1, i.e. all methods but MPB, are considered. All ensemble methods using both prototypes outperform the corresponding algorithms using only one prototype for any of the considered combination rules

Table 1. Results of the ensemble methods. The recognition rate of the prototype classifiers C_1 and C_2 is 70.92 % and 70.71 %, respectively.

algorithm	C	voting	perf. v.	ga. v.
Bagging	C_1	71.11 %	71.2 %	70.82 %
Bagging	C_2	70.83 %	71.01 %	70.92 %
AdaBoost	C_1	72.23 %	72.33 %	72.23 %
AdaBoost	C_2	71.39 %	71.76 %	71.67 %
random subspace	C_1	71.29 %	71.01 %	71.29 %
random subspace	C_2	70.26 %	70.45 %	70.08 %
Bagging	C_1, C_2	71.29 %	71.67 %	71.67 %
AdaBoost	C_1, C_2	72.8 %	72.51 %	72.7 %
random subspace	C_1, C_2	71.86 %	71.95 %	71.39 %
MPB	C_1, C_2	73.36 %	73.08 %	73.17 %

(see rows 1-9). This means that in 6 out 6 cases the algorithm described in Section 2 produced results that are superior to the corresponding classic ensemble methods. When applying the sign test [12] the finding that the algorithm described in Section 2 is better is statistically significant using a significance level of 2%. This shows that the algorithm described in Section 2 takes advantage of the diversity of the prototypes.

AdaBoost in conjunction with C_1, C_2 (abbreviated as AdaBoost(C_1, C_2) in the following) produced the best result out of rows 1 to 9. Considering also the last row, we notice that the novel algorithm, MPB, was better than AdaBoost(C_1, C_2) for any of the combination rules. MPB achieved in average a recognition rate that is 0.5 % higher than AdaBoost(C_1, C_2). This shows that the MPB algorithm has the potential of improved performance over the algorithm described in Section 2.

In the second set of experiments a training set of 18,920 words and a test set of 3,264 words were used. The vocabulary of the experiment contains 3,997 words, i.e. a classification problem with 3,997 different classes is considered. The set of writers of the training set and the set of writers of the test set are disjoint. So the experiments are again writer independent. The total number of writers who contributed to this set is 153. Prototype classifiers C_3 and C_4, as described in Section 4, are used. The recognition rates of these classifiers are 80.36 % and 71.57 %, respectively.

As classifier C_4 uses transformed (and reduced) feature vectors, an application of the random subspace method is not suitable for this classifier. Yet Bagging and AdaBoost are applicable.

For this set of experiments the optimal number of classifiers was determined in a separate experiment. The optimal number of classifiers was found to be 21 for Bagging and 14 for AdaBoost. For the algorithm described in Section 2 we produced ensembles of similar size as for the corresponding classical method. In addition the same number of classifiers as AdaBoost, 14, was used for MPB. There are two reason for doing this. First, MPB and AdaBoost are quite similar

Table 2. Results of the second set of experiments. The recognition rate of prototype classifiers C_3 and C_4 is 80.36 % and 71.57 %, respectively.

algorithm	C	voting	perf. v.	ga. v.
Bagging	C_3	81.1 %	80.91 %	81.13 %
Bagging	C_4	73.07 %	73.04 %	72.95 %
AdaBoost	C_3	82.02 %	81.89 %	81.92 %
AdaBoost	C_4	74.3 %	73.9 %	73.77 %
random subspace	C_3	80.73 %	80.58 %	80.61 %
Bagging	C_3, C_4	83 %	83.64 %	83.21 %
AdaBoost	C_3, C_4	83.76 %	83.79 %	84.07 %
MPB	C_3, C_4	83.79 %	84.16 %	84.16 %

because they are both boosting algorithms. Second, by using the same number of classifiers, a more objective comparison is possible.

The results of the second set of experiments are shown in Table 2 where the same notation as in Tables 1 is used. Again, we notice that all ensemble methods using both prototypes outperform the corresponding algorithms using only one prototype for any combination rule. To compare the ensemble methods more thoroughly, the results of the methods using several prototypes were compared to the results of the algorithms using prototype C_3 with respect to statistical significance. It was observed that the superiority of the ensemble methods using both prototypes was statistically significant for all combination schemes using a significance level of 0.1 %. MPB again produced the best results. Although the superiority of MPB over AdaBoost(C_3, C_4) is not statistically significant, it is an indication that MPB is capable of outperforming the AdaBoost version which uses several prototypes.

Please note that all algorithms using both prototypes produced good results despite the fact that half of the classifiers were produced from prototype C_4, which has a much lower performance than prototype C_3. It seems that the increase of diversity by adding classifiers produced out of prototype C_4 had a larger impact on the performance than the rather low individual performance of these additional classifiers.

6 Conclusions

In this paper, the generation of ensembles of classifiers from a set of prototype classifiers was studied. A new ensemble method, multi probabilistic boosting (MPB), has been proposed. The new ensemble method was experimentally evaluated together with previously introduced ensemble methods using several prototypes in complex handwritten word recognition tasks. The ensemble methods using several prototypes were also compared to classical ensemble methods.

The ensemble methods using two prototype classifiers were found to be superior to the ensemble methods using only one of the two prototype classifiers. The performance could be further improved by using the MPB method. These

findings were consistent for two independent data sets and for all of the three considered combination rules.

In future research we will focus on the use of more than two prototype classifiers. For example all four prototype classifiers C_1, C_2, C_3, C_4 described in this paper could be applied together.

Acknowledgment

This research was supported by the Swiss National Science Foundation (Nr. 20-52087.97). The authors thank Dr. Urs-Victor Marti for providing the handwritten word recognizer and Matthias Zimmermann for the segmentation of a part of the IAM database. Additional funding was provided by the Swiss National Science Foundation NCCR program "Interactive Multimodal Information Management (IM)2" in the Individual Project "Scene Analysis".

References

1. *Proc of the 7th Int. Conf. on Document Analysis and Recognition*, Edinburgh, Scotland, 2003.
2. A. Brakensiek, J. Rottland, A. Kosmala, and G. Rigoll. Off-line handwriting recognition using various hybrid modeling techniques and character n-grams. In *7th International Workshop on Frontiers in Handwritten Recognition*, pages 343–352, 2000.
3. Leo Breiman. Bagging predictors. *Machine Learning,* (2): 123–140, 1996.
4. T. G. Dietterich. Ensemble methods in machine learning. In *[17]*, pages 1–15.
5. T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artifical Intelligence Research*, 2:263–286, 1995.
6. T.G. Dietterich and E.B. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Departement of Computer Science, Oregon State University, 1995.
7. Yoav Freund and Robert E. Schapire. A decision-theoretic generalisation of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.
8. S. Günter and H. Bunke. Generating classifier ensembles from multiple prototypes and its application to handwriting recognition. In *[18]*, pages 179–188.
9. S. Günter and H. Bunke. New boosting algorithms for classification problems with large number of classes applied to a handwritten word recognition task. In *[28]*, pages 326 – 335.
10. S. Günter and H. Bunke. Optimizing the number of states, training iterations and Gaussians in an HMM-based handwritten word recognizer. In *[1]*, volume 1, pages 472–476.
11. T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
12. D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Research and Development in Information Retrieval*, pages 329–338. 1993.
13. S. Impedovo, P. Wang, and H. Bunke, editors. *Automatic Bankcheck Processing.* World Scientific Publ. Co, Singapore, 1997.

14. A. Kaltenmeier, T. Caesar, J.M. Gloger, and E. Mandler. Sophisticated topology of hidden Markov models for cursive script recognition. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 139–142, 1993.
15. G. Kim, V. Govindaraju, and S.N. Srihari. Architecture for handwritten text recognition systems. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 163–172. World Scientific Publ. Co., 1999.
16. M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns.* John Wiley and Sons, New York, 2001.
17. J. Kittler and F. Roli, editors. *First International Workshop on Multiple Classifier Systems*, Cagliari, Italy, 2000. Springer.
18. J. Kittler and F. Roli, editors. *Third International Workshop on Multiple Classifier Systems*, Cagliari, Italy, 2002. Springer.
19. D. Lee and S. Srihari. Handprinted digit recognition: A comparison of algorithms. In *Third International Workshop on Frontiers in Handwriting Recognition*, pages 153–162, 1993.
20. U. Marti and H. Bunke. The IAM-database: An English sentence database for off-line handwriting recognition. *Int. Journal of Document Analysis and Recognition*, 5:39–46, 2002.
21. U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Art. Intelligence*, 15:65–90, 2001.
22. D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4):869–893, 1996.
23. L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.
24. J.-C. Simon. Off-line cursive word recognition. *Special Issue of Proc. of the IEEE*, 80(7):1150–1161, July 1992.
25. C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Special Issue of Proc. of the IEEE*, 80(7):1162–1180, 1992.
26. A. Vinciarelli. *Offline Cursive Handwriting: From Word to Text Recognition.* PhD thesis, University of Bern, Switzerland, 2003.
27. A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of large vocabulary cursive handwritten text. In *[1]*, volume 2, pages 1101–1105.
28. T. Windeatt and F. Roli, editors. *4th Int. Workshop on Multiple Classifier Systems*, Guildford, United Kingdom, 2003. Springer.
29. L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics,* 22(3):418–435, 1992.
30. S. J. Young, J. Jansen, J. J. Odell, D. Ollason, and P. C. Woodland. *The HTK Hidden Markov Model Toolkit Book.* Entropic Cambridge Research Laboratory, http://htk.eng.cam.ac.uk/, 1995.
31. M. Zimmermann and H. Bunke. Hidden Markov model length optimization for handwriting recognition systems. In *Proc. of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 369–374, 2002.

Network Intrusion Detection by a Multi-stage Classification System*

Luigi Pietro Cordelia[1], Alessandro Limongiello[2], and Carlo Sansone[1]

[1] Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli "Federico II"
Via Claudio 21,I-80125 Napoli, Italy
{cordel,carlosan}@unina.it

[2]Dip. di Ingegn. dell'Informazione ed Ingegn. Elettrica, Università degli Studi di Salerno
Via Ponte Don Melillo, I,I-84084, Fisciano (SA), Italy
alimongiello@unisa.it

Abstract. A serial multi-stage classification system for facing the problem of intrusion detection in computer networks is proposed. The whole decision process is organized into successive stages, each one using a set of features tailored for recognizing a specific attack category. All the stages employ suitable criteria for estimating the reliability of the performed classification, so that, in case of uncertainty, information related to a possible attack are only logged for further processing, without raising an alert for the system manager. This permits to reduce the number of false alarms.
The proposed multi-stage intrusion detection system has been tested on two different services (http and ftp) of a standard database used for benchmarking intrusion detection systems. The experimental analysis highlights the effectiveness of the approach: the proposed system behaves significantly better than other multi-expert systems performing classification in a single stage.

1 Introduction

The increasing number of different services nowadays offered through the Internet determines a strong request for exploiting computer network security techniques that permit to protect Internet providers and/or commercial sites from malicious attacks (intrusions). A variety of approaches for facing the network intrusion detection problem has been proposed until now [1]. This notwithstanding, an Intrusion Detection Systems (IDS) can be basically ascribed to two different categories. The first one, that exploits signatures of known attacks for detecting when an attack occurs, is known as *misuse,* or *signature, detection.* IDSs that fall in this category are based on a model of all the possible misuses of the network resources. The completeness request is actually their major limit [2].

* This work has been partially supported by the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)".

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 324–333, 2004.

A dual approach tries to characterize the normal usage of the resources under monitoring. An intrusion is then suspected when a significant difference from the resource's normal usage is revealed. IDSs following this approach, known as *anomaly detection,* seem to be more promising because of their potential ability to detect unknown intrusions. However, in this case the major challenge is the need of acquiring a model of the normal use general enough to allow authorized users to work without raising false alarms, but specific enough to recognize unauthorized usages [3,4].

Different attack types can occur in a real network. Kendall [5] proposed a taxonomy of attacks, grouping them into four major categories: Probes, Denial of service (DoS), Remote to local (R2L) and User to root (U2R). The first category is made up of attacks that test a potential target to collect information about a possible intrusion. Therefore, they are usually harmless, unless a vulnerability is discovered and later exploited. DoS attacks prevent normal operation, causing the target host or a server to crash, or blocking network traffic; they, however, do not violate the target host. On the contrary, the last two categories group together attacks that permit the attacker to compromise the target host. In particular, in R2L attacks, an unauthorized user is able to bypass normal authentication and to execute commands on the target host, while in U2R attacks, a user with login access is able to bypass normal authentication to gain the privileges of another user, typically the *root* user.

With this taxonomy in mind, the network intrusion detection problem can be easily formulated as a typical pattern recognition problem [6]: given information about network connections between pairs of hosts, the task is to assign each connection to one out of five classes, that represent normal traffic conditions or one of the four different attack categories described above. Here the term "connection" refers to a sequence of data packets related to a particular service, as a file transfer via the ftp protocol. Since an IDS must detect connections related to malicious activities, each network connection can be viewed as a "pattern" to be classified.

This formulation implies the use of an IDS based on a misuse detection approach. The main advantage of the pattern recognition approach is the generalization capability exhibited by pattern recognition systems. They are able to detect some novel attacks, without the need of a complete description of all the possible attacks' signatures, so overcoming one of the main drawbacks of the misuse detection approach. In [7] the feasibility of the pattern recognition approach for the intrusion detection problem is addressed. Different pattern recognition systems have been proposed in the recent past for realizing an IDS, mainly based on neural network architectures [3,8,9]. In order to improve the performance, approaches based on multi-expert architectures have been also proposed [6,10].

However, it should be worth noticing that one of the main drawbacks occurring when using pattern recognition techniques in real environments is the high false alarm rate they often produce [6] (this drawback, indeed, is shared by a large number of commercial IDS). Moreover, the classification is usually performed in a feature space made up of all the features needed to detect the considered attack classes. Since the distributions of the classes is very unbalanced, it is necessary to employ a quite large set of features having different semantic meaning to reliably distinguish be-

tween patterns coming from distinct classes. This is not advisable, because the excessive size of the feature vectors could make more difficult the construction of reliable classifiers.

Starting from these considerations, in this paper we propose a multi-stage classification system for network intrusion detection. Each stage performs a binary classification, distinguishing between normal connections and a single type of attack, and utilizes features especially tailored for performing classification between these two classes. In order to decrement the number of missed detection, the proposed multi-stage system declares a connection as normal traffic only if all the stages do not detect an attack. Furthermore, in order to keep low the number of false alarms, some criteria have been established for evaluating the reliability of each classification act directly from the output of the classifiers. The reliability value is used to reject patterns that are unreliably attributed to an attack class. Reject, in this case, implies that the data about a 'rejected' connection are only logged for further processing, without raising an alert for the system manager. This should be done by using, for example, the proposed multi-stage system as an engine detection module of a public-domain IDS like *Snort* [11].

The organization of the paper is as follows: in Section 2 the proposed architecture is described, while in Section 3 several tests on a standard database used for benchmarking IDS are reported, together with a comparison of the proposed system with some parallel multi-expert systems. Finally, some conclusions are drawn.

2 The Proposed Approach

As anticipated in the introduction, if the detection of an intrusion is performed in a feature space made up of all the features needed to detect all the considered attack classes, the excessive size of the feature vectors could make difficult the construction of a reliable classifier. Moreover, the false alarm rate should be maintained low in order to make pattern recognition techniques appealing also for a system manager.

In order to address the first problem, the proposed architecture is made up of a cascade of stages; each stage considers a different set of features and is tailored for distinguishing only between normal traffic and one attack class. Therefore each stage behaves as a binary classifier on a (possibly) reduced set of features, so augmenting the probability of increasing the overall system performance. As regards the flow of the decision process, if a stage recognizes a pattern as normal traffic, the pattern is forwarded to the successive stage for further classification. In such a way, a pattern is recognized as normal traffic only if all the stages attribute it to the normal traffic class. This choice should permit to keep low the number of undetected attacks.

As regards the dual need of decreasing the false alarm rate, it must be noted that each stage is made up of an expert, devoted to the classification of an input pattern, and of a decider. This latter, on the basis of the output vector provided by the corresponding expert, estimates the reliability of the classification decision, so isolating all the patterns that can be reliably considered as attacks. If a stage was not able to reliably assigning the sample to an attack, the pattern is rejected. On the contrary, if an

expert recognizes a pattern as normal traffic, it is always forwarded to the successive stage for further classification, independently of reliability.

An overview of the decisional flow of the proposed system is given in Fig. 1. Note that the reliability parameters, whose values range from 0 to 1, are indicated with ψ, and the reliability thresholds (formally defined hereafter) with σ. These symbols have a subscript denoting the stage they refer to.

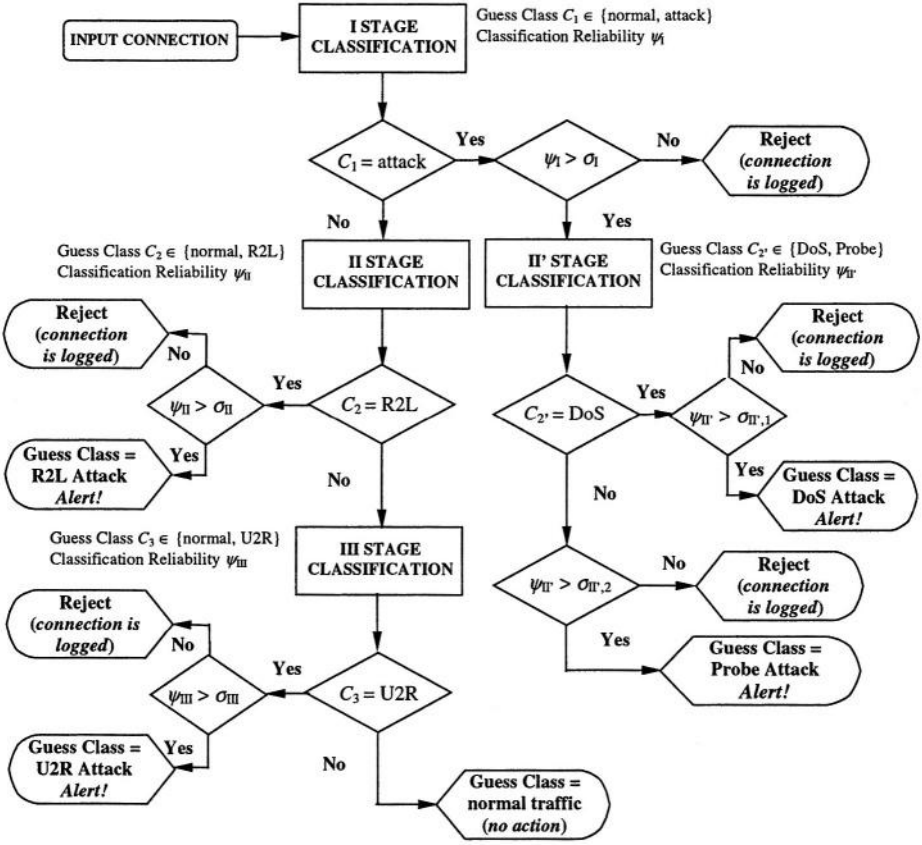


Fig. 1. The implemented decisional flow as a function of the classification decisions at the different stages and of the corresponding reliability evaluations.

The classification process starts by presenting the input pattern to the first stage. It is devoted to discriminate between normal traffic and attacks belonging to DoS or Probe classes. These two attack classes are considered together since their features should be quite similar. If the expert of this stage reliably recognizes the pattern as an attack, a second stage is activated for discriminating between DoS and Probe classes. However, if the classification of one of these two stages is under a suitable threshold, the input pattern is rejected, i.e. the data related to the connection are logged for fur-

ther processing and no alert is raised. The method used for fixing the thresholds for each stage will be illustrated in the next subsection.

On the other hand, if the first stage does not attribute the input connection to the attack class, the stage devoted to discriminate between normal connections and R2L attacks is activated. If the expert of this stage attributes the input pattern to the R2L class with high reliability, an alert is generated and the classification process stops; on the contrary, if the reliability is under a suitably fixed threshold the pattern is rejected and the data relative to the input connection are logged. Once again, the process continues, instead, if the second stage expert classifies the input pattern as belonging to normal traffic, no matter for the associated reliability. In this case, a third (and last) stage is activated. It must discriminate between normal connection and U2R classes, adopting the same decision rule of the previous stage. In summary, it is worth noticing that while the attribution to the attack class can be done by a single expert, a pattern can be assigned to the normal traffic class only after taking into account the results of three decision stages.

The choice of considering DoS and Probe attack together in the first stage is justified because connections related to these types of attacks are characterized by a behavior that is typically quite different from the one exhibited by normal connections. Thus, it is simpler to detect them at the first step. On the contrary, U2R and R2L attacks are more difficult to recognize and are even more dangerous, since their aim is to violate the target host. Obviously, the sequence the different stages are activated influences the overall system performance. In the next Section, tests will be made to confirm that the chosen activation sequence guarantees the best system performance.

2.1 Reliability Thresholds

The low reliability of a classification can be traced back to one of the following situations: *a*) the considered sample is significantly different from those present in the training set; *b)* the point which represents the sample considered in the feature space lies where the regions pertaining to different classes overlap. To distinguish between classifications which are unreliable because a sample is of type *a* or *b,* we define two reliability parameters, ψ^a and ψ^b whose values vary between 0 (completely unreliable) and 1 (very reliable). Each parameter is a function of the classifier output vector; in [12] the definition of the reliability parameters in case of some popular neural architectures are given.

A parameter ψ providing an inclusive measure of the reliability of a classification can be computed by combining the values of ψ^a and ψ^b. The form chosen for ψ is: $\psi = \min\{\psi^a, \psi^b\}$. This is certainly a conservative choice because it implies that a low value for just one of the parameters is sufficient to consider unreliable the whole classification. However, it is consistent with the kind of classification system considered, which is aimed at achieving the highest reliability.

Once the reliability of each classification has been evaluated, the optimal values of the thresholds can be determined by using the method described in [13]. It is assumed that an effectiveness function P is defined which, taking into account the require-

ments of the particular application, evaluates the quality of the classification in terms of correct recognition, misclassification and rejection rates. Under this assumption the optimal reject threshold value σ, determining the best trade-off between reject rate and misclassification rate, is the one for which the function P reaches its absolute maximum.

The requirements of the particular application domain are specified by attributing costs to misclassifications, rejects and correct classifications. As in [13], in our case the cost of an error is different as a function of the actual class. So a cost matrix C_{ij} has to be defined, whose generic element indicates the cost of misclassifying a pattern belonging to the i-th class by attributing it to the j-th class. We can assume that the gain for correct classification and the cost of a reject are not dependent on the class of the pattern. On the contrary, we can expect that the costs of a misclassification are quite different depending on the actual and the guess class.

Finally, it is worth noticing that misclassification costs are typically higher than the reject costs. This is true also in our domain, where a reject implies that the data relative to the 'rejected' connection are logged by the IDS for an off-line processing and so there is still the possibility of detecting an intrusion.

3 Experimental Results

The proposed system has been tested on a subset of the database created by DARPA in the framework of the 1998 *Intrusion Detection Evaluation Program.* It is made up of a large number of network connections related to normal and malicious traffic. This database was pre-processed by the Columbia University giving rise to a feature vector of 41 elements for each connection, according to the set of features defined in [14] and tailored for the intrusion detection problem. In the database each connection is labelled as belonging to one out of the five classes described in the previous Sections: normal traffic, Probe, DoS, U2R and R2L attacks. It is worth noticing that each attack class is made up of different attack variants, each one exploiting different vulnerabilities of a computer network.

Our results have been systematically compared with those obtained from some single classifiers operating in a single stage and some parallel Multi-Expert Systems (MES). As single experts we considered two different neural architectures: a Learning Vector Quantization (LVQ) and a three layered Multi-Layer Perceptron (MLP). In order to train them, the training data was split into two disjoint sets: a training set (in the following TRS) and a training-test set (in the following TTS), used to stop the learning process so as to avoid a possible overtraining. Different classifiers have been experimented, by varying the number of hidden nodes for the MLP nets and by using different numbers of prototypes for the LVQ nets. Also different feature sets have been used for training.

Different combining rules have been taken into account in order to build several MES, each one composed by two or three of the previously described neural architec-

tures. In particular, both "fixed" (Majority Vote, Min, Max, Average) and "trainable" (Naïve Bayes, Dempster-Shafer, BKS) combining rules have been considered.

The results obtained by all the considered classification systems are reported in terms of *i*) the overall classification error, *ii)* the sum of the false alarm and the missed detection rates, without taking into account class confusion among attack classes, and *iii)* the average classification cost calculated according to the cost matrix shown in Table 1. This type of evaluation has been proposed in [7] so as to have a more significant parameter for evaluating the effectiveness of an IDS. The average cost is computed by multiplying each entry of the confusion matrix with the corresponding entry in the cost matrix and dividing the result by the total number of test samples. The cost matrix of Table 1 is also used for evaluating the reliability thresholds by using the method illustrated in Section 2.1. In this case the cost of a reject is assumed to be 0.5.

Table 1. Cost matrix used to weight the confusion matrix related to each classifier and to calculate the reliability thresholds. The cost of a reject is assumed to be 0.5.

	Guess Class				
Actual Class	Normal	DoS	U2R	R2L	Probe
Normal	0	2	2	2	1
DoS	2	0	2	2	1
U2R	3	2	0	2	2
R2L	4	2	2	0	2
Probe	1	2	2	2	0

In the following we present the results obtained by our classification method applied to two different network services (ftp and http) among those present in the DARPA database. Other services have been also experimented, but the obtained results are not reported here for the sake of brevity. The choice of designing a different multi-stage classifier system for each service follows the so-called modular approach presented in [10], where the authors experimentally demonstrate the advantage, in terms of recognition performance, of an IDS that develops a different classification module for each one of the network services to be protected.

3.1 FTP Service

In this case the training data was made up of 798 patterns related to different attacks and to the normal class. These samples are not well balanced among the different classes: in particular, there are very few samples of U2R and Probe attacks. For this reason such samples have been duplicated before training, giving rise to 841 training data. This procedure was motivated by the fact that it is possible to easily retrieve on the Web the code for launching some type of attacks, and then it is reasonable to assume the possibility of having several identical connections in case of attacks. We use 65% of training data as TRS (549) and the remaining 35% for TTS (292). The Test Set (TS) for this service is made up of 825 patterns.

Table 2 reports the results of the best single classifiers and of the best parallel MES made up of three experts. As it is evident, the results in terms of overall error are very poor, while the percentage of false alarms and missed detections is quite acceptable. In this case the best MES does not significantly improve the performance obtained by the best single classifier.

Table 3 shows the characteristics of each stage of the proposed architecture. All the stages employ an LVQ net as expert, trained with the same modalities of the single classifiers described above. Note that different feature sets have been selected at different stages.

Table 2. Results obtained by the best single experts and by the best Parallel MES on the TS.

Classifier		*Overall Error*	*Cost*	*False+Missed alarm rates*
Single	MES			
LVQ (all features)		13.82 %	0.3358	3.76 %
LVQ (six features)		14.18 %	0.2861	1.09 %
	Average	13.70 %	0.2564	1.05 %
	Majority Vote	13.70 %	0.2564	1.05 %

Table 3. Details about each stage of the proposed system.

Stage	Neural Architecture	Features	Feature Normalization	Threshold
I	LVQ	duration, src_bytes, dst_bytes, same_srv_rate, dst_host_srv_diff_host_rate	Yes	$\sigma_I = 0.406$
II	LVQ	All	Yes	$\sigma_{II} = 0.210$
III	LVQ	All	No	$\sigma_{III} = 0.100$
II'	LVQ	All	Yes	$\sigma_{II',1} = 0.001$ $\sigma_{II',2} = 0.003$

The results obtained on the TS by the proposed system are reported in Table 4. For the sake of comparison, both the systems with and without the use of the reject option have been considered. The overall error is significantly lower with respect to the previous case. Moreover, the false alarm rate further decreases with the introduction of the reject option, as it should be.

It is worth noticing that the chosen activation sequence for the stages of the proposed architecture is really the most effective one. In fact, the overall error, without the reject option, raises to the 9.45% if the II Stage is chosen as the first one in our architecture, and to the 8.36% if the III Stage is the first to be activated.

Table 4. Results obtained on the TS by the proposed multi-stage classification system, with and without the reject option.

	Overall error	*Cost*	*False+Missed alarm rates*
Without reject option	2.79 %	0.0509	0.85 %
With reject option	0.72% (with 3.76 % of reject)	0.0309	0.24 %

3.2 Http Service

The training data for this service in the DARPA database are made up of 64292 patterns. However, in [9] it has been demonstrated that a dataset of about 15% of the whole http data is sufficient to training neural classifiers. Therefore, only 8866 samples have been considered as training data. Differently from the previous case, there are no attacks belonging to the U2R class. This implies that the proposed multi-stage classification system for this service do not present the III Stage. Also in this case samples of the R2L and Probe classes have been duplicated before training. The 70% of the whole training data has been used as TRS and the remaining 30% for the TTS. The TS for this service is made up of 40442 patterns.

Table 5 reports the results of the best single classifiers and of the best parallel MES made up of three experts. In this case the performance are quite good, especially as regards the number of false alarms. The best MES exhibits exactly the same performance of the best single expert.

Table 5. Results obtained by the best single experts and by the best Parallel MES on the TS.

Classifier		*Overall error*	*Cost*	*False +Missed alarm rates*
Single	**MES**			
LVQ		0.22 %	0.0027	0.09 %
MLP		0.27 %	0.0033	0.14 %
	Average	0.22 %	0.0027	0.09 %
	Majority Vote	0.22 %	0.0027	0.09 %

Table 6 shows the characteristics of each stage of the proposed architecture; all of them employ an LVQ net as expert. Once again, these experts were trained with the same modalities of the single classifiers reported in Table 5. In this case, the feature set is the same for all the three stages, even if in one case no normalization was performed. Since the expert present at each stage is very reliable, no thresholds were fixed by the proposed method in this case.

Table 6. Details about each stage of the proposed system.

Stage	Neural Architecture	Features	Feature Normalization	Threshold
I	**LVQ**	All	Yes	0.00
II	**LVQ**	All	Yes	0.00
II'	**LVQ**	All	No	0.00

Finally, table 7 shows the results of the proposed system on the TS. Also in this case the overall error decreases with respect to the previous case, keeping low the false alarm rate. This confirms the effectiveness of the proposed approach.

Table 7. Results obtained by the proposed multi-stage classification system on the TS.

Overall error	*Cost*	*False +Missed alarm rates*
0.11 %	0.0014	0.09 %

4 Conclusions

In this paper we have presented a multi-stage classification system for network intrusion detection. For this architecture, we have used some criteria for evaluating the reliability of the response and a method for the determination of an optimal reject option, in order to reduce the false alarm rate of the system. The effectiveness of our proposal has been experimentally evaluated on a standard database, where a significant improvement in the reliability of the system has been demonstrated.

References

1. S. Axelsson, Research in Intrusion Detection Systems: A Survey, TR 98-17, Chalmers University of Technology, 1999.
2. R. Kumar, E.H. Spafford, "A Software Architecture to Support Misuse Intrusion Detection", in Proceedings of the 18th National Information Security Conference, pp. 194-204, 1995.
3. A.K. Ghosh, A. Schwartzbard, "A Study in Using Neural Networks for Anomaly and Misuse Detection", Proc. 8'th USENIX Security Symposium, Aug. 26-29 1999, Washington DC.
4. T. Lane, C.E. Brodley, "Temporal Sequence learning and data reduction for anomaly detection", ACM Trans. on Inform. and System Security, vol. 2, no. 3, pp. 295-261, 1999.
5. K. Kendall, A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, Master's Thesis, Massachusetts Institute of Technology, 1998.
6. G. Giacinto, F. Roli, L. Didaci, "Fusion of multiple classifiers for intrusion detection in computer networks", Pattern Recognition Letters, vol. 24, pp. 1795-1803, 2003.
7. C. Elkan, "Results of the KDD99 classifier learning", ACM SIGKDD Explorations 1, 63–64, 2000.
8. S. C. Lee, D.V. Heinbuch, "Training a neural Network based intrusion detector to recognize novel attack", IEEE Trans. Syst, Man., and Cybernetic, Part-A, vol. 31, pp. 294-299, 2001.
9. M. Fugate, J.R. Gattiker, "Computer Intrusion Detection with Classification and Anomaly Detection, using SVMs", International Journal of Pattern Recognition and artificial Intelligence, vol. 17, no. 3, pp. 441-458, 2003.
10. G. Giacinto, F. Roli, L. Didaci, "A Modular Multiple Classifier System for the Detection of Intrusions", Lecture Notes in Computer Science vol. 2709, pp. 346-355, 2003.
11. J. Beale, J.C. Foster, Snort 2.0 Intrusion Detection, Syngress Publishing, Inc., Rockland, MA, 2003.
12. L.P. Cordella, C. Sansone , F. Tortorella, M. Vento, C. De Stefano, "Neural Networks Classification Reliability", in C.T. Leondes (ed.). Academic Press theme volumes on Neural Network Systems, Techniques and Applications. Academic Press, 5, pp. 161-199, 1998.
13. C. Sansone, F. Tortorella, M.Vento, "A Classification Reliability Driven Reject Rule for Multi-Expert Systems", International Journal of Pattern Recognition and Artificial Intelligence, vol. 15, no. 6, pp. 885-904, 2001.
14. W. Lee, S.J. Stolfo, "A framework for constructing features and models for intrusion detection systems", ACM Transactions on Inform. System Security, vol. 3, no. 4, pp. 227-261, 2000.

Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules

Vladimir Svetnik, Andy Liaw, Christopher Tong, and Ting Wang

Biometrics Research RY33-300, Merck & Co., Inc.
P.O. Box 2000, Rahway, NJ 07065, USA
{vladimir_svetnik,andy_liaw,Christopher_tong,ting_wang}@merck.com

Abstract. Leo Breiman's Random Forest ensemble learning procedure is applied to the problem of Quantitative Structure-Activity Relationship (QSAR) modeling for pharmaceutical molecules. This entails using a quantitative description of a compound's molecular structure to predict that compound's biological activity as measured in an *in vitro* assay. Without any parameter tuning, the performance of Random Forest with default settings on six publicly available data sets is already as good or better than that of three other prominent QSAR methods: Decision Tree, Partial Least Squares, and Support Vector Machine. In addition to reliable prediction accuracy, Random Forest provides variable importance measures which can be used in a variable reduction wrapper algorithm. Comparisons of various such wrappers and between Random Forest and Bagging are presented.

1 Introduction

In the drug discovery process, candidate drug compounds are assayed for potency against disease targets; but they must also be assayed for toxicity and pharmacokinetic properties involving absorption, distribution, metabolism, and excretion (ADME). These biological activities are usually measured in *in vitro* bioassays. The activities could be continuous numbers (like drug concentration producing 50% inhibition) or categorical labels (like active/inactive). We would like to model the relationship between a compound's biological activity and its molecular structure, where the latter is characterized quantitatively by a set of topological descriptors. This problem is an example of Quantitative Structure-Activity Relationship (QSAR) modeling [Eki00,Haw01]. The prediction of a continuous response is equivalent to regression; that of a categorical response is equivalent to classification.

The number of descriptors, p, usually exceeds the number of samples, n, and many of the descriptors may be irrelevant to predicting the activity of interest. Traditional statistical methods (multiple linear regression [MLR], linear discriminant analysis [LDA], and k-nearest neighbors [kNN]) cannot be used reliably without a sophisticated variable selection filter, such as, for example, a genetic algorithm. This approach is indeed taken by some investigators. Other approaches

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 334–343, 2004.

found in the literature include Decision Tree (recursive partitioning [RP]), Partial Least Squares (PLS), artificial neural networks (ANN), and support vector machines (SVM), although ANN and SVM again often require variable preselection if there are a large number of irrelevant descriptors. The Decision Tree is relatively free of the aforementioned limitations; however, it suffers from low accuracy. Ensembles of trees are a natural choice for QSAR modeling, since they combine the desirable properties of Decision Trees with high prediction performance. Among the most promising ensemble learning methods are boosting and Random Forest, and in this paper we study Random Forest, developed by Leo Breiman [Bre01]. We are currently investigating boosting, which will be the topic of a future report.

2 Random Forest

The Algorithm. Like Bagging, Random Forest is an ensemble of unpruned trees. Each tree is trained on a bootstrap sample of the training data, and predictions are made by majority vote of the trees (in classification) or averaging their outputs (in regression). Random Forest differs from Bagging in that at each node of each tree, the algorithm considers as splitting candidates a random sample of the variables instead of all the variables. The size of the variable subset is a fixed value, *mtry,* with default value $p^{1/2}$ for classification and $p/3$ for regression. The idea is to maintain the "strength" of the trees while reducing their correlation with each other. Breiman [Bre01] has shown that an upper bound on the generalization error of Random Forest is given by $r(1 - s^2)/s^2$, where r is a measure of the correlation between the trees, and s is a measure of their strength (see [Bre01] for the details). Since the unpruned trees are low-bias, high variance models, averaging over an ensemble of trees reduces variance while keeping low bias (see below). It is also thought that an ensemble of trees mitigates the semi-artificiality of the tree structure (hyper-rectangular partition of the descriptor space) and the greediness of the tree-growing algorithm, which are arguably the two drawbacks of the Tree approach.

Random Forest also provides additional features that increase its utility for QSAR modeling (see [Bre01,Sve03] for further discussion):

1. Out-of-bag predictions, useful for error estimation or threshold selection (see below for explanation);
2. A measure of variable importance, useful for model interpretation; and
3. A measure of intrinsic proximity between two compounds, useful for computing "neighbor molecules". (Not discussed further here.)

Out-of-Bag Predictions. Since each tree in the ensemble is grown on a bootstrap sample of the data, the molecules left out of the bootstrap sample, the "out-of-bag" (OOB) data, can be used as a legitimate test set for that tree. On average, one-third of the training data will be "out-of-bag" for a given tree [Has01]. Consequently, each molecule in the training data will be left out of (on average) 1/3 of the trees in the ensemble; we can compile the predictions for each

molecule when it is "out-of-bag" and use these OOB predictions to estimate the error rate of the full ensemble. This is similar to a cross-validation performance estimate, but at a much lower computational cost.

Variable Importance. Random Forest's variable importance measure is based on the following heuristic. When a descriptor that contributes to prediction accuracy is "noised up" (e.g., replaced with random noise), the accuracy of prediction should noticeably degrade. On the other hand, if a descriptor is irrelevant, "noising" it up should have little effect on the performance. A number of specific variable importance measures were proposed by Breiman based on this heuristic [Bre01]. We currently use an importance measure that he proposed recently to remedy a subtle problem with the previous meausres (Breiman, personal communication). The new measure is computed as follows: first compute the OOB error rate (or MSE) of each tree, and also compute the same for OOB data with one variable permuted; take the difference between these. The new measure is the mean difference (over all data) divided by the standard error of these differences. This variable importance measure could be used to select a subset of the most important descriptors, and partial dependence plots [Has01] can be produced for each of these, to see the trend, e.g., whether increasing a descriptor's value tends to increase the biological activity (in regression), or probability of activity (in classification, where probability is interpreted as proportion of votes in favor of the majority class).

3 Performance Assessment

In [Sve03], we selected six publicly available ADME data sets for the assessment of Random Forest in competition with two other popular QSAR methods, Recursive Partitioning (RP) and Partial Least Squares (PLS). The performance was evaluated by computing test set predictions from 5-fold cross-validation (CV), repeated on 50 different partitions of the data; we report median accuracy rates (percent of compounds correctly classified, in classification) and correlations between actual and predicted value (in regression). We review these results here and include new results from Support Vector Machine (SVM) using both a linear kernel and a radial-basis function (RBF) kernel with optimized parameters.

Briefly, the five classification data sets are the following: BBB (blood-brain barrier permeability) [Don02], ER (estrogen receptor binding activity) [Ton03], P-gp (P-glycoprotein transport activity) [Pen02], MDRR (multidrug resistance reversal activity) [Bak00], and COX-2 (inhibition of cyclooxygenase-2) [Kau01]. The COX-2 data were assigned to categorical labels based on a cutoff on numerical measurements of their $\log(\mathrm{IC}_{50})$; direct prediction of these numerical measurements was also done as a regression problem. (The number of molecules is greater in the classification case because there were some molecules whose activity was known only qualitatively, not quantitatively.) Another regression data set, D2, was related to predicting the $\log(\mathrm{IC}_{50})$ for binding affinity to the dopamine D_2 receptor [Gil92]. Please refer to [Sve03] for further details of these data sets, such as the types of descriptors used. The results of the performance

Table 1. Median accuracy rates for classification test data from 50 5-fold cross-validations (n = number of compounds; p = number of descriptors)

Data	n	p	RF	RP	PLS	SVM (linear)	SVM (RBF)
BBB	325	9	0.809	0.737	0.691	0.735	0.782
ER	232	197	0.828	0.759	0.813	0.765	0.815
P-gp	186	1522	0.806	0.712	0.769	0.801	0.804
MDRR	528	342	0.830	0.780	0.821	0.813	0.831
COX-2	314	135	0.780	0.736	0.774	0.774	0.774

Table 2. Median correlation between predicted and actual values for regression test data from 50 5-fold cross-validations (n = number of compounds; p = number of descriptors)

Data	n	p	RF	RP	PLS	SVM (linear)	SVM (RBF)
COX-2	272	135	0.658	0.525	0.573	0.608	0.646
D2	116	374	0.698	0.504	0.658	0.696	0.708

assessments are shown in Tables 1 and 2. The results show, over a range of data sets, that Random Forest *at its default settings* is already as good as or better than the other competing methods. Parameter tuning will be discussed in the next section.

For the P-gp data set, we used a set of 1522 in-house generated atom pair descriptors. As a "stress test" of the Random Forest algorithm, we also generated an in-house set of 43,928 three-dimensional fingerprint descriptors for the same data, and were able to obtain an median CV accuracy of 0.80 (although the calculations required considerably more time).

4 Parameter Optimization and Variable Reduction

Random Forest has a parameter, *mtry,* that could be considered a tuning parameter. In [Sve03] we showed empirically that the performance of Random Forest using a fixed set of descriptors is often relatively insensitive to the choice of *mtry,* specified as a function of the number of descriptors (like $mtry = p^{1/2}$), over a large range of choices, as long as *mtry* is far from its minimum or maximum possible values (1 or p, respectively). However, the sensitivity of the algorithm to the choice of *mtry* may depend on the proportion of irrelevant variables in the training data. Two other parameters in Random Forest include the number of trees and the minimum node size. The number of trees should simply be sufficiently large that the ensemble statistic of interest has stabilized. For instance, if one is interested in accuracy rates and variable importance measures, in the P-gp data set, we found that 1000 trees is adequate. However, if one is interested in the proportion of votes for a given class, we found that we have to grow at least 10,000 trees to get stable results. As for the minimum node size, which is

the minimum size of nodes below which no split will be attempted, in classification this has a default value of 1 (allowing trees of full depth to be grown) and in regression the default is 5. The latter case is somewhat arbitrary, but the performance of Random Forest is relatively insensitive to changes in its value (as long as it is kept small).

Variable reduction based on Random Forest's variable importance measure is another potential way to optimize the Random Forest algorithm. The removal of irrelevant variables may improve the performance of the algorithm upon retraining and may help improve the interpretability of the model. As an illustration, we implemented the following variable reduction "wrapper" algorithm:

1. Partition the data for 5-fold cross-validation (CV).
2. On each CV training set, train a model on all variables and use the variable importance measure to rank them. Record the CV test set predictions.
3. Use the variable ranking to remove the least important half of the variables and retrain the model, predicting the CV test set. Repeat removal of half of the variables until there are about 2 left.
4. Aggregate results from all 5 CV partitions and compute the error rate (in classification) or mean squared error (in regression) at each step of halving.
5. Replicate steps (1)-(4) 20 times to "smooth out" the variability.

It is vital to note that this procedure is non-recursive; that is, on each training run of CV, the variable importance is calculated just once, at the beginning, and is not recalculated repeatedly as the variables are reduced. (A recursive version of this procedure is much greedier and, in our experience, has much worse performance.) Note also that the CV error, and not the OOB error, is used to assess performance in the wrapper algorithm. (In the next section, we will show that the use of OOB data for this purpose leads to severe overfitting.)

On the P-gp data, the median error rates for the 50 replications, with medians connected by line segments, is shown in Fig. 1, for various choices of *mtry.* The cases of *mtry* equaling p (equivalent to bagging), $p/2$, $p/4$, and the default $p^{1/2}$ are considered. The plot shows that the default *mtry,* $p^{1/2}$, performs the best, but the other choices are only a few percent worse. Also, the performance remains about the same as irrelevant variables are removed, until reaching 191 variables; further variable reduction will degrade it.

The robustness of Random Forest's performance to the presence of irrelevant variables is illustrated here. It is certainly possible to improve the performance by a small amount by tuning parameters or reducing variables. For instance, in the COX-2 data set, we found that Bagging outperforms Random Forest by a couple percent [Sve03]. We have never yet observed a case where the performance actually improves as variables are reduced. This, and the relatively small change in performance due to different choices of *mtry,* show that Random Forest can perform reasonably well "off the shelf" without a lot of tuning or variable reduction.

Ambroise and McLachlan [Amb02] and Reunanen [Reu03] have argued compellingly that the assessment of performance of a learning machine with variable

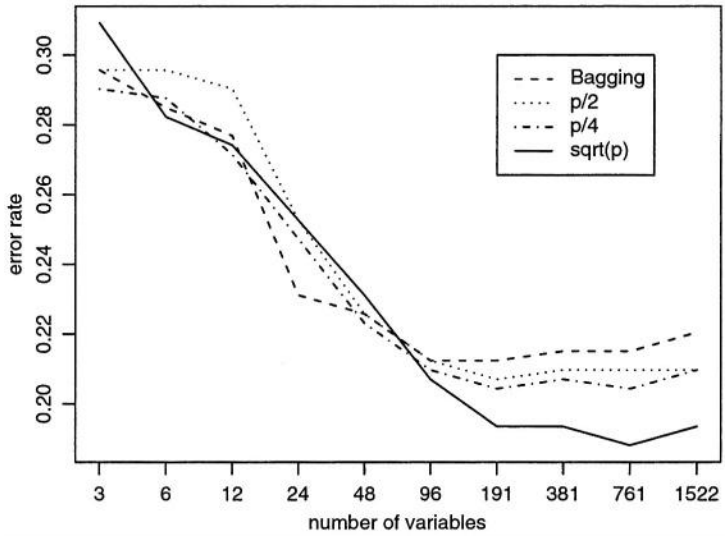


Fig. 1. Median CV test error rates at each step of halving the important variables, using different *mtry* functions, for the P-gp data. Line segments connect the medians of 20 5-fold CV error rates

selection requires great care. *Both* the variable selection and the supervised training should be embedded within a CV procedure in order to obtain an honest assessment of the total learning system's performance. If the variable selection is done outside of CV, a *selection bias* is introduced. Following this principle, a performance assessment of our procedure requires that the entire algorithm be nested within another CV loop [Reu03]. For the P-gp data set, the median nested CV error rate based on 10 replications of 5-fold CV embedded within 10 replications of 10-fold CV is 0.191. This performance is actually quite consistent with Fig. 1, indicating that the selection bias is negligible in this example.

5 Comparison with Other Wrappers

At first glance, a reasonable alternative to the variable reduction procedure outlined above is one that is exactly the same, except that the variable importance is *recalculated* at step 3, producing a new ranking of the variables. This corresponds to a recursive feature elimination procedure similar to the one used by Guyon *et al.* [Guy02]. Our limited experience shows that this approach performs poorly, since it is more prone to overfitting than the non-recursive approach. A comparison of the performance of these two approaches is shown in Fig. 2 for the P-gp data and Fig. 3 for the D2 data. As variable reduction proceeds onward, the recursive approach produces much higher error rates, especially in the D2 case, than the non-recursive approach. Because of its inferior performance, we recommend avoiding the recursive procedure.

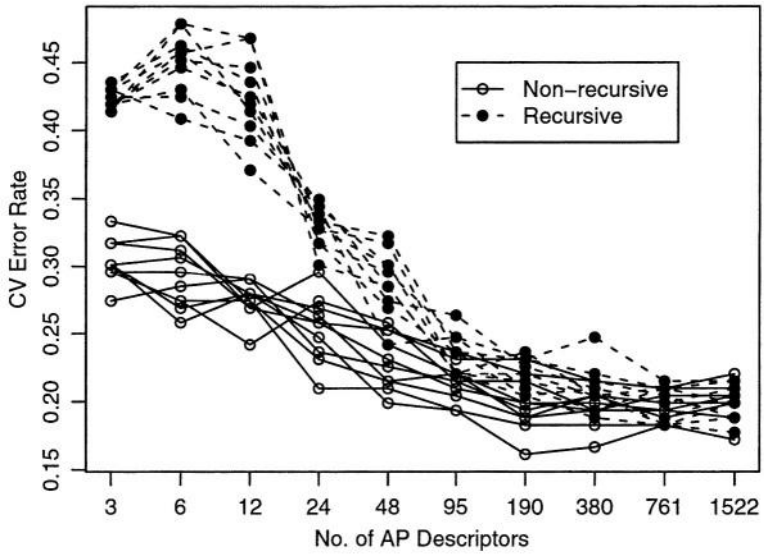


Fig. 2. Comparisons of the medians of CV test error rate performance of our wrapper variable selection procedure and a recursive version of it (P-gp data). The traces correspond to ten replications of 5-fold cross-validation

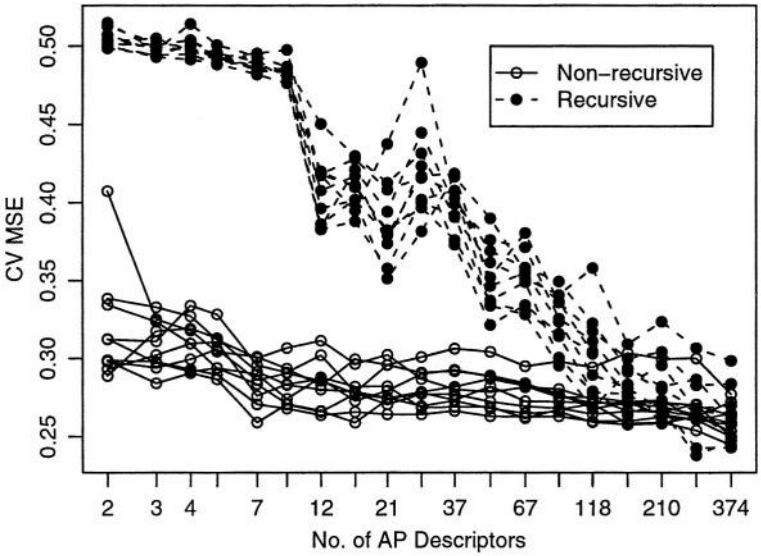


Fig. 3. Comparisons of the medians of CV test error rate performance of our wrapper variable selection procedure and a recursive version of it (D2 data). The traces correspond to ten replications of 5-fold cross-validation

Another variable reduction wrapper that could be attempted is one that uses the Random Forest OOB prediction error, rather than the CV error, to assess performance in step 4 of the wrapper algorithm. We tried this with the P-gp data with the class labels *randomly scrambled.* We compared the CV Test set error and the OOB prediction error during variable selection (see Fig. 4) The CV Test set error does reflect what we expect: performance close to that of random guessing, regardless of how many variables are used. On the other hand, the OOB prediction error rate incorrectly suggests that performance can be improved substantially by doing variable reduction. This shows that the use of out-of-bag error estimates for iterative performance assessment lends itself to severe overfitting. This is because the OOB error estimate for any reduced number of variables is contaminated by the initial variable ranking, which was based on *all* the data. The curve in Fig. 4 shows that with a sufficient number of variables remaining, this wrapper can do a good job of fitting to noise.

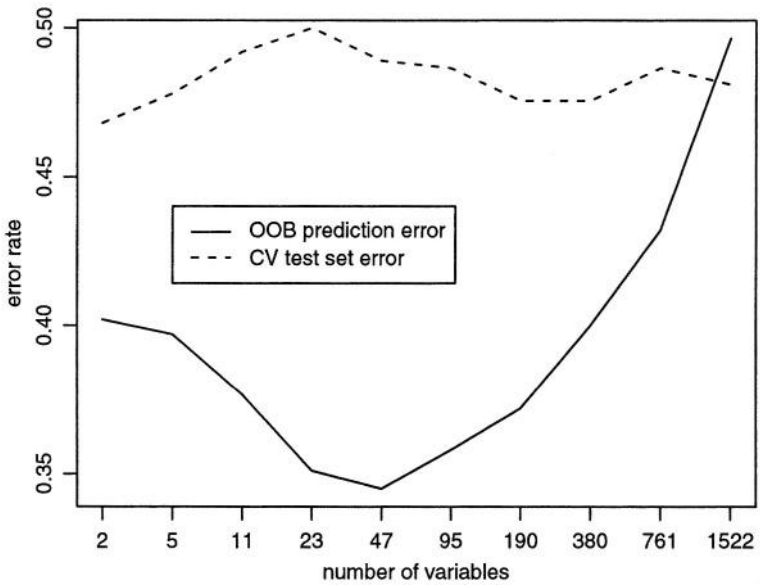


Fig. 4. Medians of CV test set and out-of-bag prediction error rates at each step of halving the important variables (P-gp data with randomized response vector). Line segments connect the medians of 20 5-fold CV error rates

6 Conclusion

Random Forest, as a tree ensemble learning method, has a combination of desirable properties for QSAR modeling and excellent prediction performance when used "off the shelf" (i.e., without parameter tuning or variable selection). We also showed how to use a variable selection wrapper with Random Forest, and how this wrapper is superior to two other possible wrappers. These results convince us that tree ensemble methods are very promising for QSAR modeling in

the drug discovery process, and we continue to explore Random Forest and other methods like Boosting for these applications.

7 Software

Open source software for Random Forest is publicly available. The Fortran code for the Random Forest software, written by L. Breiman and A. Cutler, is found at `http://www.stat.berkeley.edu/users/breiman/RandomForests/` and an R interface for it by A. Liaw and M. Wiener [Lia02] can be found at `http://cran.us.r-project.org/` by looking for the `randomForest` package. The SVM software we used is from SPIDER, found at `http://www.kyb.tuebingen.mpg.de/bs/people/spider/`.

Acknowledgments

We would like to thank Leo Breiman, Jerry Friedman, and Tom Dietterich for useful discussions; Bob Sheridan, Chris Culberson, and Brad Feuston for insights on QSAR modeling and descriptors; and Bruce Bush for improvements to the text. We thank Jason Weston for his help with the SVM sofware.

References

[Amb02] Ambroise, C., McLachlan, G. J.: Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. Natl, Acad. Sci. USA* **99** (2002) 6562–6566

[Bak00] Bakken, G. A., Jurs, P. C., Classification of multidrug-resistance reversal agents using structure-based descriptors and linear discriminant analysis. *J. Med. Chem.* **43** (2000) 4534–4541

[Bre98] Breiman, L.: Arcing classifiers. *Ann. Stat.* **26** (1998) 801–849

[Bre01] Breiman, L.: Random forests. *Machine Learning* **45** (2001) 5–32

[Don02] Doniger, S., Hofmann, T., Yeh, J.: Predicting CNS permeability of drug molecules: comparison of neural network and support vector machine algorithms. *J. Comput. Biol.* **9** (2002) 849–864

[Eki00] Ekins, S. et al.: Progress in predicting human ADME parameters in silico. *J. Pharmac. Toxic. Meth.* **44** (2000) 251–272

[Fri03] Friedman, J. H., Popescu, B. E.: Importance sampled learning ensembles. `http://www-stat.stanford.edu/~jhf/ftp/isle.pdf`

[Gil92] Gilligan, P. J. et al.: Novel piperidine σ receptor ligands as potential antipsychotic drugs. *J. Med. Chem.* **35** (1992) 4344–4361

[Guy02] Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. *Machine Learning* **46** (2002) 389–422

[Has01] Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer-Verlag (2001)

[Haw01] Hawkins, D. M., Basak, S. C., Shi, X.: QSAR with few compounds and many features. *J. Chem. Inf. Comput. Sci.* **41** (2001) 663–670

[Kau01] Kauffman, G. W., Jurs, P. C.: QSAR and k-nearest neighbor classification analysis of selective cyclooxygenase-2 inhibitors using topologically-based numerical descriptors. *J. Chem. Inf. Comput. Sci.* **41** (2001) 1553–1560

[Lia02] Liaw, A., Wiener, M.: Classification and regression by randomForest. *R News* **2/3** (2002) 18–22

[Pen02] Penzotti, J. E., Lamb, M. L., Evensen, E., Grootenhuis, P. D. J.: A computational ensemble pharmacophore model for identifying substrates of p-glycoprotein. *J. Med. Chem.* **45** (2002) 1737–1740

[Reu03] Reunanen, J.: Overfitting in making comparisons between variable selection methods. *J. Machine Learning Res.* **3** (2003) 1371–1382

[Sve03] Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., Feuston, B. P.: QSAR modeling using Random Forest, an ensemble learning tool for regression and classification. J. *Chem. Inf. Comput. Sci.* **43** (2003) 1947–1958

[Ton03] Tong, W.; Hong, H.; Fang, H.; Xie, Q.; Perkins, R.: Decision forest: combining the predictions of multiple independent decision tree models. *J. Chem. Inf. Comput. Sci.* **43** (2003) 525–531

Experimental Study on Multiple LDA Classifier Combination for High Dimensional Data Classification

Xiaogang Wang and Xiaoou Tang

Department of Information Engineering
The Chinese University of Hong Kong
{xgwang1,xtang}@ie.cuhk.edu.hk

Abstract. Multiple classifier systems provide an effective way to improve pattern recognition performance. In this paper, we use multiple classifier combination to improve LDA for high dimensional data classification. When dealing with the high dimensional data, LDA often suffers from the small sample size problem and the constructed classifier is biased and unstable. Although some approaches, such as PCA+LDA and Null Space LDA, have been proposed to address this problem, they are all at cost of discarding some useful discriminative information. We propose an approach to generate multiple Principal Space LDA and Null Space LDA classifiers by random sampling on the feature vector and training set. The two kinds of complementary classifiers are integrated to preserve all the discriminative information in the feature space.

1 Introduction

Multiple classifier combination is an effective way to improve pattern recognition performance. Random subspace [4] and bagging [5] are two popular techniques to combine weak classifiers into a powerful decision rule. In the random subspace method, a set of low dimensional subspaces are generated by randomly sampling from the high dimensional feature vector and multiple classifiers constructed in the random subspaces are combined in the final decision. In bagging, random independent bootstrap replicates are generated by sampling the training set. A classifier is constructed from each replicate, and the results of all the classifiers are finally integrated. Based on the two random sampling techniques, we propose an approach using multiple LDA classifier combination for high dimensional data classification.

Linear Discriminant Analysis (LDA) is a popular feature extraction technique for data classification. It determines a set of projection vectors maximizing the between-class scatter matrix (S_b) and minimizing the within-class scatter matrix (S_w) in the projective feature space. But when dealing with the high dimensional data, LDA often suffers from the small sample size problem. When there are not enough training samples, S_w is not well estimated and may become singular [3].

To address this problem, a two-stage PCA+LDA approach [1] is proposed. The high dimensional data is first projected to a low dimensional PCA subspace, in which S_w is non-singular, and then LDA is performed. We call it Principal Space LDA.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 344–353, 2004.

The eigenvectors with small eigenvalues removed from the PCA subspace may also encode some information helpful for recognition. Their removal may introduce a loss of discriminative information.

Chen et. al. [2] suggested that the null space spanned by the eigenvectors of S_w with zero eigenvalues contains the most discriminative information. However, as explained in [2], with the existence of noise, when the training sample number is large, the null space of S_w becomes small, so much discriminative information outside this null space will be lost.

Some random sampling based LDA classification approaches can be found in [7][8]. Different from the previous work, our method simultaneously samples on the feature space and training samples, and takes advantage of the discriminative information in both the principal and null spaces of S_w. We also explain that both Principal Space LDA (P-LDA) and Null Space LDA (N-LDA) encounter the overfitting problem, but for different reasons. So we will improve them in different ways accordingly. A more detailed description on the algorithm can be found in [9][10]. In this paper, we make an extensive experimental study on the XM2VTS database [12].

2 LDA for High Dimensional Data Classification

Two conventional LDA approaches, PCA+LDA and N-LDA are briefly reviewed in this section. The high dimensional data is represented as a vector $\vec{x}$ with length *N*. The training set contains *M* samples belonging to *L* classes.

2.1 PCA+LDA

Principal Component Analysis (PCA) computes a set of eigenvectors of the ensemble covariance matrix *C* of the training set. Eigenvectors are sorted by eigenvalues, which represent the variance of data distribution. There are at most *M-1* eigenvectors with non-zero eigenvalues. Normally *K* eigenvectors, $U = [\vec{u}_1, \ldots \vec{u}_K]$, with the largest eigenvalues, are selected to span the PCA subspace. Low dimensional features are extracted by projecting the high dimensional data $\vec{x}$ into the PCA subspace,

$$\vec{w} = U^T(\vec{x} - \vec{m}). \tag{1}$$

where $\vec{m}$ is the mean of the training set.

LDA tries to find a set of projecting vectors W maximizing the ratio of determinant of S_b and the determinant of S_w,

$$W = \arg\max \left| \frac{W^T S_b W}{W^T S_w W} \right|. \tag{2}$$

W can be computed from the eigenvectors of $S_w^{-1} S_b$ [6]. The rank of S_w is at most *M-L*. But when the training set is small and *M-L* is smaller than the vector length *N*, S_w may become singular and it is difficult to compute S_w^{-1}.

In the two-stage PCA+LDA approach [1], the data vector is first projected to a PCA subspace spanned by the *M-L* largest eigenvectors. LDA is then performed in the *M-L* dimensional subspace, such that S_w is nonsingular. But in many cases, *M-L* dimensionality is still too high for the training set. So the LDA classifier is often biased and unstable. Furthermore, much discriminative information outside the PCA subspace is discarded.

2.2 Null Space LDA

Chen et. al. [2] suggested that the null space of S_w also contains much discriminative information. It is possible to find some projection vectors *W* satisfying $W^T S_w W = 0$ and $W^T S_b W \neq 0$, thus the Fisher criteria in Eq. (2) definitely reaches its maximum value. The rank of S_w, $r(S_w)$, is bounded by min(*M* –*L, N*). Because of the existence of noise, $r(S_w)$ is almost equal to this bound. The dimension of the null space is max(0,*N*–*M* + *L*). As shown by experiments in [2], when the training sample number is large, the null space of S_w becomes small, thus much discriminative information outside this null space will be lost.

3 Multiple LDA Classifier Combination for High Dimensional Data Classification

Both P-LDA and N-LDA face the same problem: the constructed classifier is unstable and much discriminative information is discarded. But they are caused by different reasons. So we design different random sampling algorithms to improve the two LDA methods, and combine them in a multiple classifier structure.

3.1 Using Random Subspace to Improve P-LDA

In P-LDA, overfitting happens when the training set is relatively small compared to the high dimensionality of the feature vector. In order to construct a stable LDA classifier, we sample a small subset of features to reduce discrepancy between the training set size and the feature vector length. Using such a random sampling method, we construct a multiple number of stable LDA classifiers, and combine them into a powerful classifier covering the entire feature space without losing discriminative information.

We first apply PCA to the training set. All the eigenvectors with zero eigenvalues are removed, since all the training samples have zero projections on them. The *M-1* eigenvectors $U_0 = \{\vec{u}_1, \ldots, \vec{u}_{M-1}\}$ with positive eigenvalues are retained as candidates to construct random subspaces. Then, *K* random subspaces are generated. The dimen-

sion of random subspace is determined by the training set to make the LDA classifier stable. In each random subspace, the first N_0 dimensions are fixed as the largest eigenvectors, and the remaining N_1 dimensions are randomly selected from $\{\vec{u}_{M-N_0-1}, \ldots, \vec{u}_{M-1}\}$. The N_0 largest eigenvectors encode much data structural information. If they are not included in the random subspace, the accuracy of LDA classifiers may be too low. Our approach guarantees that the LDA classifier in each random subspace has satisfactory accuracy. The N_1 random dimensions cover most of the remaining small eigenvectors. So the ensemble classifiers also have a certain degree of error diversity.

3.2 Using Bagging to Improve N-LDA

In N-LDA, the overfitting problem happens when the training sample number is large, since the null space will be too small. It can be alleviated by bagging. In bagging, random independent bootstrap replicates are generated by sampling the training set, so each replicate has a smaller number of training samples. We Generate *K* replicates by randomly sampling the training set. A N-LDA classifier is constructed from each replicate and the multiple classifiers are combined using a fusion rule.

3.3 Integrating Random Subspace and Bagging for LDA Based Classification

While P-LDA is computed from the principal subspace of S_w, in which $W^T S_w W \neq 0$, N-LDA is computed from its orthogonal subspace in which $W^T S_w W = 0$. Both of them discard some discriminative information. Fortunately, the information retained by the two kinds of classifiers complements each other. So we combine them to construct the final classifier. Many methods on combining multiple classifiers have been proposed [11]. In this paper, we use two simple fusion rules: majority voting and sum rule. More complex combination algorithms may further improve the system performance.

4 Experiments

We apply the random sampling based LDA approach to face recognition and make a extensive experimental study on the XM2VTS face database [12]. There are 295 people, and each person has four frontal face images taken in four different sessions. In our experiments, two face images of each class are selected for training, and the remaining two for testing. In preprocessing, the face image is normalized by translation, rotation, and scaling, such that the centers of two eyes are in fixed positions. A 46 by 81 mask removes most of the background. So the face data dimension is $46 \times 81 = 3726$. We adopt the recognition test protocol used in FERET [13]. All the face classes in the reference set are ranked. We measure the percentage of the "correct answer in top 1 match".

4.1 Random Subspace LDA

We first compare random subspace LDA with the conventional PCA+LDA approach. Table 1 reports the accuracy of a single P-LDA classifier constructed from PCA subspace with different dimension. Since there are 590 face images of 295 classes in the training set, there are 589 eigenfaces with non-zero eigenvalues. According to [1], the PCA subspace dimension should be *M-L=295*. However, the result shows that the accuracy is only 79% using a single P-LDA classifier constructed from 295 eigenfaces, because this dimension is too high for this training set and S_w cannot be well estimated. We observe that P-LDA classifier has the best accuracy 92.9% when the PCA subspace dimension is set at 100. So for this training set 100 seems to be a suitable dimension to construct a stable P-LDA classifier. In the following experiments, we choose 100 as the dimension of random subspaces to construct the multiple P-LDA classifiers.

First, we generate the random subspaces by randomly selecting 100 eigenfaces from 589 eigenfaces with nonzero eigenvalues. The result of combining 20 P-LDA classifiers using majority voting is shown in Figure 1. The accuracy of each individual P-LDA classifier is low, between 50% and 70%. Using majority voting, the weak classifiers are greatly enforced, and 87% accuracy is achieved. This shows that P-LDA classifiers constructed from different random subspaces are complementary of each other. In Table 2, as we increase the classifier number *K,* the accuracy of the combined classifier improves, and even becomes better than the highest accuracy in Table 1. Although increasing classifier number and using more complex combining rules may further improve the performance, it will increase the system burden.

Table 1. Recognition accuracy of PCA+LDA classifier constructed from PCA subspace with different dimension.

Dim	30	50	70	100	150	200	250	295
Accuracy	0.870	0.925	0.927	0.929	0.898	0.864	0.820	0.792

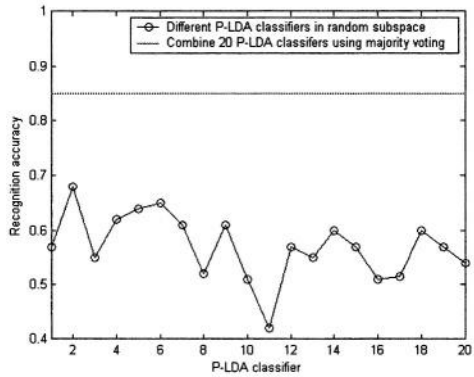


Fig. 1. Recognition accuracy of combing 20 P-LDA classifiers constructed from random subspaces using majority voting. Each random subspace randomly selects 100 eigenfaces from 589 eigenfaces with non-zero eigenvalues.

Table 2. Accuracy of combining different number (K) of P-LDA classifiers constructed from random subspaces using majority voting. Each random subspace randomly selects 100 eigenfaces from 589 eigenfaces with non-zero eigenvalues.

K	20	40	60	80	100	120	140	160
Accuracy	0.871	0.907	0.917	0.922	0.937	0.932	0.939	0.939

Table 3. Recognition accuracy of P-LDA classifiers constructed from different parts of eigenface sequence which has been sorted by eigenvalues. The first row is the index of eigenfaces spanning the subspace from which LDA classifier is constructed, and the second row is the recognition accuracy.

Index	1-100	101-200	201-300	301-400	401-500	501-589	vote
Accuracy	0.929	0.514	0.378	0.148	0.06	0.04	0.613

Table 4. Recognition accuracy of combining P-LDA classifiers using different number (K) of random subspaces (sum rule). In each random subspace, the first 50 dimensions are fixed as the 50 largest eignfaces, and another 50 dimensions are randomly selected from the remaining 593 eigenfaces with positive eigenvalues. We run ten times on the same training set and testing set, and record the accuracy means and variances.

K	5	10	15	20	25	30
Mean	0.954	0.958	0.959	0.961	0.961	0.962
Variance	0.0133	0.0127	0.0094	0.0101	0.0068	0.0049

Some largest eigenfaces encode much face structural information. If they are not included in the random subspace, the individual LDA classifier is poor. This can be further proved in Table 3, in which six LDA classifiers are constructed based on different parts of eigenface sequence. The first row is the index of eigenfaces spanning the subspace. Using only the eigenfaces with small eigenvalues, the recognition accuracy of LDA classifier is poor. But it doesn't mean these eigenfaces are not useful for recognition.

A better approach to improve the performance of the combined classifier is to increase the accuracy of each individual weak classifier. To improve the accuracy of each individual P-LDA classifier, as illustrated in Section 3.1, in each random subspace, we fix the first 50 basis as the 50 largest eigenfaces, and randomly select another 50 basis from the remaining 539 eigenfaces. As shown in Figure 2, individual P-LDA classifiers are improved significantly. They are similar to the LDA classifier based on the first 100 eigenfaces. These classifiers are also complementary of each other, so much better accuracy (96%) is achieved when they are combined. The recognition performance of using different number of random subspaces is shown in Table 4. We run 10 times on the same training set and testing set, recording the accuracy means and variances. Using more random subspaces, the accuracy is higher and more stable.

We also apply random subspace to N-LDA. Similar to the method in Section 3.1, the random subspaces with dimension D ($295<D<590$) are generated from PCA subspace and a N-LDA classifier is constructed from each random subspace. As shown in Figure 3, there is no improvement in recognition performance. When the random

subspace dimensionality D is low, the null space dimension (D-295) is small, so the recognition accuracy drops greatly. Random subspace further reduces the null space dimension and deteriorates the overfitting problem of N-LDA.

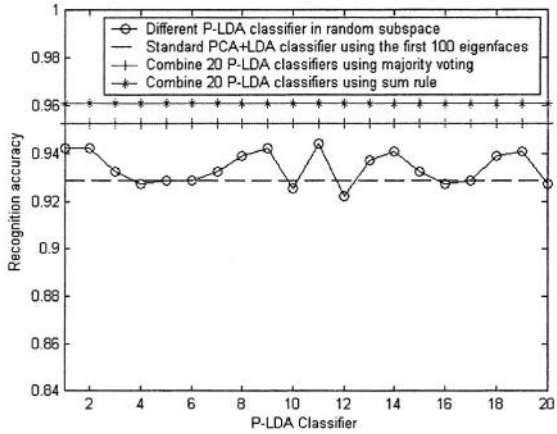


Fig. 2. Recognition accuracy of combing 20 P-LDA classifiers constructed from random subspaces. For each 100 dimensional random subspace, the first 50 dimensions are fixed as the 50 largest eigenfaces, and another 50 dimensions are randomly selected from the remaining 539 eigenfaces with non-zero eigenvalues.

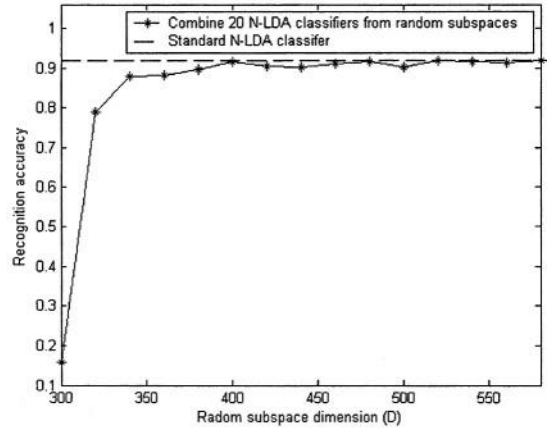


Fig. 3. Recognition accuracy of combining 20 N-LDA classifiers from random subspaces with different dimensions using majority voting.

4.2 Bagging LDA

Figure 4 reports the performance of bagging based N-LDA. We generate 20 replicates and each replicate contains 300 training samples. The individual N-LDA classi-

fier constructed from each replicate is less effective than the original classifier trained on the full training set. This is because that some intra-class variations are not included in each replicate. However, when the multiple classifiers are combined, the accuracy is significantly improved, and becomes much better than the standard N-LDA. Table 5 reports performance of bagging based N-LDA using different number of replicates, but fixing training sample number in each replicate as 300. As similar in Table 4, it is more stable using a relatively large number of replicates. In Figure 5, we fix the bagging replicates number as 20, but change the training sample number contained in the replicates from 100 to 500. The best performance is achieved using proper moderate training sample number in each replicate. When the training sample number in each replicate is too small, the null space cannot effectively remove the intra-class variation. When the training sample number in each replicate is too large, the null space dimension is too small to contain enough discriminative information, and different replicates are similar.

Table 5. Recognition accuracy of combining N-LDA classifiers using different number (K) of bagging replicates (sum rule). We run ten times on the same training set and testing set, and record the accuracy means and variances.

K	5	10	15	20	25	30
Mean	0.929	0.934	0.942	0.956	0.951	0.961
Variance	0.0120	0.0109	0.097	0.009	0.036	0.027

We also study using bagging to improve P-LDA classifiers. The PCA subspace is spanned by the 100 largest eigenfaces and 20 replicates are generated. The accuracies with the replicate containing different number of people are shown in Figure 6. As expected, the combined classifier shows no improvement over the original P-LDA classifier. In each replicate, the P-LDA classifier is constructed from an even smaller number of training samples. It deteriorates the small sample size problem.

4.3 Integrating Random Subspace and Bagging Based LDA

Integrating the multiple P-LDA classifiers generated by random subspace and N-LDA classifiers generated by bagging, the recognition accuracy can be further improved. We combine 10 P-LDA classifiers constructed from random subspaces and 10 N-LDA classifiers constructed from bagging replicates, and set an even better result as shown in Table 6.

Table 6. Compare random sampling based LDA with conventional LDA approaches. R-LDA (1): random subspace based LDA; R-LDA (2): bagging based N-LDA; R-LDA (3): integrating random subspace and bagging based LDA

PCA+LDA	N-LDA	R-LDA (1)	R-LDA (2)	R-LDA (3)
0.929	0.919	0.961	0.956	0.976

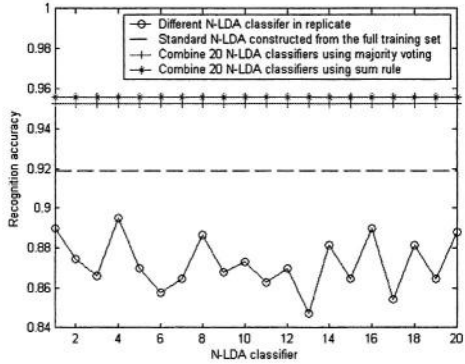


Fig. 4. Recognition accuracy of combining 20 N-LDA classifiers constructed from bagging replicates.

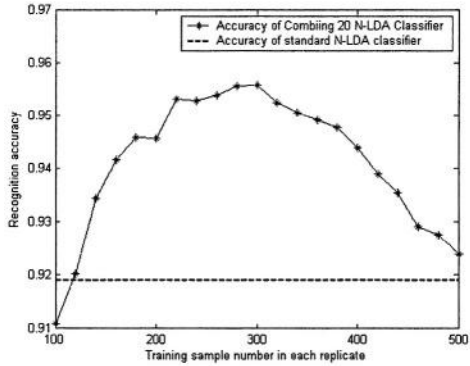


Fig. 5. Recognition accuracy of combing 20 N-LDA classifiers with different number of training samples contained in the bagging replicates (sum rule).

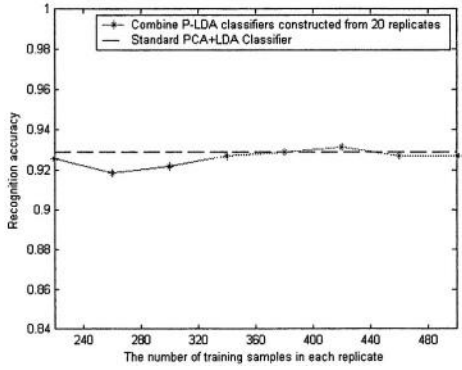


Fig. 6. Recognition accuracy of combining 20 P-LDA classifiers constructed from bagging replicates containing different number of training samples. The PCA space is spanned by 100 largest eigenfaces. The combining rule is majority voting.

5 Conclusion

Both P-LDA and N-LDA encounter the overfitting problems in when dealing with the high dimensional data classification, however, for different reasons. So we improve them using different random sampling approaches, sampling on feature for P-LDA and sampling on training samples for N-LDA. The two kinds of complementary classifiers are finally integrated in our system. The extensive experimental study on the XM2VTS database illustrates the effectiveness of our method and how it works.

Acknowledgement

The work described in this paper was fully supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region (Project no. CUHK 4190/01E and CUHK 4224/03E).

References

1. P.N. Belhumeur, J. Hespanda, and D. Kiregeman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection", *IEEE Trans, on PAMI,* Vol. 19, No. 7, pp. 711-720, July 1997.
2. L. Chen, H. Liao, M. Ko, J. Liin, and G. Yu, "A New LDA-based Face Recognition System Which can Solve the Samll Sample Size Problem", *Pattern Recognition,* Vol. 33, No. 10, pp. 1713-1726, Oct. 2000.
3. A. M. Martinez and A.C. Kak, "PCA versus LDA," *IEEE Trans, on PAMI,* Vol. 23, No. 2, pp. 228-233, 2001.
4. T. Kam Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE Trans, on PAMI,* Vol. 20, No. 8, pp. 832-844, August 1998.
5. L. Breiman, "Bagging Predictors," *Machine Learning,* Vol. 24, No. 2, pp. 123-140, 1996.
6. K. Fukunnaga, "Introduction to Statistical Pattern Recognition", Academic Press, second edition, 1991.
7. M. Skurichina and R.P.W. Duin, "Bagging and the Random Subspace Method for Linear Classifiers," Pattern Analysis and Application, Vol. 5, No. 2, pp. 121-135, 2002.
8. X. Lu and A.K. Jain, "Resampling for Face Recognition," *in Proceedings of AVBPA03,* 2003.
9. X. Wang and X. Tang, "Random Sampling Based LDA for Face Recognition," *in Proceedings of CVPR, 2004.*
10. X. Wang and X. Tang, "Random Subspace Based LDA for Face Recognition Integrating Multi-Features," in *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition,* 2004.
11. J. Kittler and F. Roli, (Eds): Multiple Classifier Systems.
12. K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre, "XM2VTSDB: The Extended M2VTS Database," *Proceedings of International Conference on Audio- and Video-Based Person Authentication,* pp. 72-77, 1999.
13. P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET Evaluation," in Face Recognition: From Theory to Applications, H. Wechsler, P. J. Phillips, V. Bruce, F.F. Soulie, and T. S. Huang, Eds., Berlin: Springer-Verlag, 1998.

Physics-Based Decorrelation of Image Data for Decision Level Fusion in Face Verification

Josef Kittler and Mohammad T. Sadeghi

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey, Guildford GU2 7XH, UK
{J.Kittler,M.Sadeghi}@surrey.ac.uk

Abstract. We consider the problem of face verification using multichannel image data where each channel serves as the input to a separate face verification expert. By decorrelating the information content of the respective data channels, we enhance the diversity of the resulting face verification experts as well as the performance of the multiple classifier system.

1 Introduction

One of the keys to success in constructing multiple classifier systems is to ensure that the component classifiers can provide complementary information for the fusion process. Such classifiers are deemed to exhibit diversity which is beneficial not only from the point of view of reducing the variance of the combined decision rule, but may even lead to the reduction of the inherent ambiguity between class populations.

There are a number of mechanisms that have been proposed to achieve this objective. In situations when the component classifiers are designed independently, their diversity can be maximised by the process of clustering [2, 6] which will help to identify experts with similar behaviour. The classifiers selected for fusion are picked as the representatives of the detected clusters. This procedure guarantees that the number of component classifiers used in the ultimate multiple classifier system is as small as possible and thus controlling the number of degrees of freedom available to the fusion decision rule.

The diversity of the component classifiers can be imposed more effectively by means of boosting [10] where the successive designs are based on resampled training sets which focus on the samples misclassified by the experts constructed earlier. This is a powerful concept but in practice its success depends on a careful trade-off between the performance of the component classifiers on the training set and their ability to generalise.

One of the easiest ways to achieve diversity is to deploy multiple sensors which themselves gather complementary information about the objects to be recognised. In the context of personal identity authentication, multiple biometric modalities such as face, voice characteristic, iris and finger print all bring to bear

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 354–363, 2004.

a completely independent piece of evidence on the authentication problem. The fusion of such multimodal experts has been shown to enhance the authentication performance significantly.

Some systems, such as a camera, are multiple sensor devices. Although they provide complementary information about the spectral properties of the imaged objects, the outputs of the respective sensor channels are not necessarily independent as they reflect not only the object's colour but also its shape. Nevertheless it has been demonstrated in a number of studies, that the decision level fusion of experts utilising such multispectral information can lead to performance improvement [11, 5].

This paper also addresses the problem of face verification using colour images. However, in contrast to the above work, here by analysing the underlying physical process of image formation, we shall show that the information content of the raw spectral images can be mapped into new image spaces which focus on complementary information content of the imaged scene. It will be demonstrated that by adopting the intensity image, intensity normalised green and opponent colour channels we will separate the imaging effects of object shape and object albedo, and create complementary image data channels that lead to face experts with an enhanced degree of diversity. The fusion of these experts will result in significant improvements in performance over the system in which the face experts work with the raw R,G,B channel data.

The paper is organised as follows. In the next section we develop the physics based transformation that orthogonalises the R,G,B channel content. The face verification system used in the study is described in Section 3. The experimental set up is detailed in Section 4. Section 5 presents the results of the experiments. Finally, in Section 6 the results are discussed and the paper is drawn to conclusion.

2 Physics Based Multichannel Image Conditioning

Let us consider an image acquisition system deploying a conventional colour camera. We assume that the imaged objects have a Lambertian surface. This assumption is reasonably well justified in the case of faces. We shall ignore the effect of interface reflection, which would distort only a small part of the face image due to the saturation of the camera. In any case, in regions giving rise to total reflection, the spectral content of the reflected light would be dominated by the illuminant, rather than the face skin, and would not provide useful information for discriminatory purposes.

Suppose the scene is illuminated by spatially invariant illumination source of spectral distribution $e(\lambda)$ where λ represents the wavelength of the incident light. Under the Lambertian assumption, the light emitted from the scene will be a function of the material properties of the scene objects, albedo, which we denote by $\alpha(x, y, \lambda)$ and the relative angles between the direction of illumination and the normal to the surface patch imaged by the (x, y) pixel of a camera sensor. The effect of the geometry will be to scale down the incident light by a factor $s(x, y)$. The output of the k^{th} sensor at pixel position (x, y) will then be given by

$$I_k(x,y) = \int_{\lambda_1}^{\lambda_2} \rho_k(\lambda)\alpha(x,y,\lambda)s(x,y)e(\lambda)d\lambda \tag{1}$$

where $\rho_k(\lambda)$ is the spectral response of the k^{th} sensor and λ_i, $i = 1,2$ are limits of the visible frequency spectrum. A typical colour camera will have three spectral channels, normally measuring the reflected light in the scene in the red, green and blue parts of the visible light spectrum. They are referred to as the R,G and B channels.

The majority of face recognition and verification systems use the intensity image, which is obtained by summing up the outputs of the R,G,B channels, i.e.

$$I(x,y) = s(x,y)\int_{\lambda_1}^{\lambda_2} \sum_{k=1}^{3} \rho_k(\lambda)\alpha(x,y,\lambda)e(\lambda)d\lambda \tag{2}$$

Assuming that $\sum_{k=1}^{3} \rho_k(\lambda) = c_1$, i.e. the combined response of the sensors is flat, and that the illuminant also has a broad flat spectrum which can be approximated by $e(\lambda) = c_2$, the intensity image acquired by the camera will given by

$$I(x,y) = c_1c_2s(x,y)\int_{\lambda_1}^{\lambda_2} \alpha(x,y,\lambda)d\lambda \tag{3}$$

which can be written as

$$I(x,y) = L(x,y)\mathcal{A}(x,y) \tag{4}$$

where $L(x,y)$ is the intensity of the incident light and $\mathcal{A}(x,y)$ is the reflectance property of the surface material.

Under the above assumptions, each channel will produce output

$$I_k(x,y) = L(x,y)\mathcal{A}_{\rho_k}(x,y) \tag{5}$$

where $\mathcal{A}_{\rho_k}(x,y) = \int_{\lambda_1}^{\lambda_2} \rho_k(\lambda)\alpha(x,y,\lambda)d\lambda$ is the reflectance property relative to the k^{th} sensor spectral band. If the sensor spectral characteristic is constant over the spectral band and zero elsewhere, $\mathcal{A}_{\rho_k}(x,y) = \mathcal{A}_k(x,y)$ will represent the intrinsic property of the material. If $\rho_k(\lambda)$ varies over the spectral band, the intrinsic albedo will be modulated by the sensor characteristic, i.e. the measured reflectance properties $\mathcal{A}_{\rho_k}(x,y)$ will be sensor specific.

It should be noted that the output of each channel is a function of the scene geometry. If geometry is the dominant factor, as it will be for 3D objects, the three channels will be highly correlated. Any decision making scheme which attempts the fuse the three data sources will be affected by these correlations. Comparing formulas (4) and (5), we can suppress the effect of geometry by dividing (5) with (4) which leads to intensity normalised channels (normalised chroma channels r, g, b)

$$\hat{I}_k(x,y) = \frac{I_k(x,y)}{I(x,y)} = \int_{\lambda_1}^{\lambda_2} \frac{\rho_k(\lambda)}{\sum_{j=1}^{3} \rho_j(\lambda)}\alpha(x,y,\lambda)d\lambda \tag{6}$$

Thus the normalised channels measure sensor specific albedo (i.e. properties of the material) uncorrupted by object geometry. For an object with slowly changing albedo, the normalised channel will show it as of constant intensity, defining its shape in terms of its occluding boundary. In contrast to the normalised channels, the intensity image manifests the changes in surface shape and provides, therefore, complementary information about scene objects.

The three normalised channels sum up to 1, hence only two of them contain useful information. More over, the information content of a pair of channels is likely to be correlated. In order to emphasise differences between the channels, rather than using the channels themselves, it should be beneficial to use one of the normalised colour channels and create one difference channel, by subtracting two normalised channel images, i.e.

$$\hat{I}_{kj}(x,y) = \hat{I}_j(x,y) - \hat{I}_k(x,y) \tag{7}$$

and shifting and rescaling to obtain non-negative pixel values. This corresponds to the idea of opponent colour spaces. The resulting representation should then maximise the complementary information content that should aid successful fusion.

3 Face Verification Process

The face verification process consists of three main stages: face image acquisition, feature extraction, and finally decision making. The first stage involves sensing and image preprocessing the result of which is a geometrically registered and photometrically normalised face image. Briefly, the output of a physical sensor (camera) is analysed by a face detector and once a face instance is detected, the position of the eyes is determined. This information allows the face part of the image to be extracted at a given aspect ratio and resampled to a pre-specified resolution. The extracted face image is finally photometrically normalised to compensate for illumination changes.

The raw colour camera channel outputs, $R(x,y)$, $G(x,y)$ and $B(x,y)$ are converted according to (2) and (6) into intensity image $I(x,y)$, the normalised chroma channels $r(x,y)$, $g(x,y)$, $b(x,y)$ and into opponent chromaticity channels by taking pairwise differences of the normalised chroma channels.

$$rg(x,y) = r(x,y) - g(x,y) \tag{8}$$

$$yb(x,y) = r(x,y) + g(x,y) - 2b(x,y) \tag{9}$$

The chromaticity and opponent images are appropriately shifted and rescaled.

In the second stage of the face verification process the face image data is projected into a feature space. We opted for the Linear Discriminant Analysis (LDA) feature space. The final stage of the face verification process involves matching and decision making. Basically the features extracted for a face image to be verified, $\mathbf{x}$, are compared with a stored template, that was acquired on enrolment, $\boldsymbol{\mu}_i$. The comparison is carried out using the isotropic gradient direction

metric, s, which was shown to outperform the Euclidean metric and normalised correlation function [91. It is defined as

$$s = \frac{||(\mathbf{x} - \boldsymbol{\mu}_i)^T \nabla_I P(i|\mathbf{x})||}{||\nabla_I P(i|\mathbf{x})||} \quad , \quad \nabla_I P(i|\mathbf{x}) = \sum_{\substack{j=1 \\ j \neq i}}^{m} p(\mathbf{x}|j)(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i) \tag{10}$$

where $\nabla_I P(i|\mathbf{x})$ denotes the gradient direction assuming an isotropic structure for the covariance matrix of the clients distribution, $p(\mathbf{x}|j)$.

The score, output by the matching process, s, which measures the degree of similarity between the test image and the template, is then compared to a threshold, η, in order to decide whether the claim is genuine (class ω_a) or impostor(class ω_b) i.e.

$$s(\mathbf{x}) \underset{\omega_b}{\overset{\omega_a}{\lessgtr}} \eta \tag{11}$$

where $(\mathbf{x})$ is the extracted LDA feature vector. If this final stage of processing is applied to each data channel separately, we end up with a number of scores, $s_k = s(\mathbf{x}_k), \ k = 1, 2, \ldots, N$ which then have to be fused to obtain the final decision. Any combination of the results of processing at this level is referred to as decision level fusion. In fact, the problem now is to find a function f so that the decision rule

$$f(s_1, s_2, \ldots, s_N) \underset{\omega_b}{\overset{\omega_a}{\lessgtr}} \eta \tag{12}$$

leads to a higher verification performance. This kind of fusion is also known as the confidence level or soft fusion [7]. Since the adopted experts in our case deliver a similar level of accuracy, their combination should either attach the same weight to all the scores or have a mechanism for selecting the best score. Thus in this study, two simple combining strategies have been considered. In the first method, we use samples average as the final score,

$$f(s_1, s_2, \ldots, s_N) = \frac{1}{N}[s_1 + s_2 + \ldots + s_N] \tag{13}$$

The second method is to select the best score as the final score, i.e.

$$f(s_1, s_2, \ldots, s_N) = \min(s_1, s_2, \ldots, s_N) \tag{14}$$

4 Experimental Design

The aim of the experiments is to show that by decorrelating the sensory data used by component experts, the performance of the multiple classifier system improves considerably. We use the BANCA database and its associated experimental protocols for this purpose.

4.1 BANCA Database

The BANCA database contains 52 subjects (26 males and 26 females). Each subject participated to 12 recording sessions in different conditions and with different cameras. Sessions 1-4 contain data under *Controlled* conditions while sessions 5-8 and 9-12 contain *Degraded* and *Adverse* scenarios respectively. Each session contains two recordings per subject, a true client access and an informed imposter attack. For the face image database, 5 frontal face images have been extracted from each video recording, which are supposed to be used as client images and 5 impostor ones. In order to create more independent experiments, images in each session have been divided into two groups of 26 subjects, 13 males and 13 females. Fig. 1 shows a few examples of the face data.

In the BANCA protocol, 7 different distinct experimental configurations have been specified, namely, Matched Controlled (MC), Matched Degraded (MD), Matched Adverse (MA), Unmatched Degraded (UD), Unmatched Adverse (UA), Pooled test (P) and Grand test (G). Table 1 describes the usage of the different sessions in each configuration. "T" refers to the client training while "C" and "I" depict client and impostor test sessions respectively. The decision function can be trained using only 5 client images per person from the same group and all client images from the other group. More details about the database and experimental protocols can be found in [1].

Fig. 1. Examples of the database images. *a*: Controlled, *b*: Degraded and *c*: Adverse scenarios.

Table 1. The usage of the different sessions in the BANCA experimental protocols.

	1	2	3	4	5	6	7	8	9	10	11	12
Mc	TI	CI	CI	CI								
Md					TI	CI	CI	CI				
Ma									TI	CI	CI	CI
Ud	T				I	CI	CI	CI				
Ua	T								I	CI	CI	CI
P	TI	CI	CI	CI	I	CI	CI	CI	I	CI	CI	CI
G	TI	CI	CI	CI	TI	CI	CI	CI	TI	CI	CI	CI

4.2 Experimental Setup

The original resolution of the image data is 720 × 576. The experiments were performed with a relatively low resolution face images, namely 64 × 49. The results reported in this article have been obtained by applying a geometric face registration based on manually annotated eyes positions. Histogram equalisation was used to normalise the registered face photometrically.

The feature selection process is performed using the linear Discriminant Analysis (LDA). The XM2VTS database [4] was used for calculating the LDA projection matrix. In this study, the isotropic Gradient Direction metric [9] (GD) was used as the scoring function. The thresholds in the decision making system were determined based on the Equal Error Rate criterion, i.e. where the false

Table 2. ID verification results using the GD metric in the R, G and B colour spaces.

		Evaluation			Test		
		FAR	FRR	TER	FAR	FRR	TER
Mc	R	0.3297	0.3333	0.663	0.8654	5.513	6.378
	G	0.2832	0.2821	0.5652	0.4808	3.205	3.686
	B	0.2811	0.2821	0.5631	0.7692	5.513	6.282
Md	R	1.156	1.154	2.31	1.058	7.821	8.878
	G	1.352	1.359	2.711	1.058	9.103	10.16
	B	1.547	1.513	3.06	2.019	4.615	6.635
Ma	R	0.9742	0.9744	1.949	2.019	7.692	9.712
	G	0.6889	0.6923	1.381	1.442	8.333	9.776
	B	0.5114	0.5128	1.024	1.923	5.641	7.564
Ud	R	13.75	13.59	27.34	13.65	14.74	28.4
	G	14.52	14.74	29.26	14.33	15.51	29.84
	B	21.15	20.9	42.05	20.58	21.03	41.6
Ua	R	15.58	16.03	31.6	16.83	17.56	34.39
	G	14.23	14.36	28.59	14.52	14.36	28.88
	B	17.5	17.56	35.06	17.02	18.08	35.1
P	R	10.58	9.957	20.53	11.31	10.85	22.17
	G	10.26	10.17	20.43	10.35	10.13	20.48
	B	13.49	14.23	27.72	13.94	14.19	28.13
G	R	1.169	1.145	2.314	2.596	1.239	3.835
	G	1.219	1.222	2.441	1.891	1.752	3.643
	B	1.526	1.538	3.064	2.179	2.65	4.829

rejection rate (FRR) is equal to the false acceptance rate (FAR). The thresholds were set either globally (GT) or using the client specific thresholding (CST) technique [3], As in the training sessions of the BANCA database, only 5 client images per person are available, in the case of global thresholding method, all these images are used for training of the clients template. The other group data is then used to set the threshold. While using the client specific thresholding strategy, only two images are used for the template training and the other three along with the other group data are used to determine the thresholds. Moreover, in order to increase the number of data used for training and to take the errors of the geometric normalisation into account, 24 additional face images per each image were generated by perturbing the location of the eyes position around the annotated positions.

5 Experimental Results

Table 2 shows the performance of the face verification system considering the individual R,G,B colour channels. Table 3 also contains the corresponding results using the intensity(i), chromaticity (r, g, b) and opponent chromaticity (rg,yb) spaces individually. In these tables, based on our previous study reported in [8], global (GT) and client specific (CST) thresholding techniques were used for unmatched (Ud, Ua, P) and matched (Mc, Md, Ma, G) protocols respectively.

Table 3. ID verification results on BANCA configurations using GD metric in the intensity (i), chromaticity (r,g,b) and opponent chromaticity (rg,yb) spaces.

	chromaticity							intensity and opponent chromaticity						
		Evaluation			Test				Evaluation			Test		
	I_i	FAR	FRR	TER	FAR	FRR	TER	I_i	FAR	FRR	TER	FAR	FRR	TER
Mc	r	0.414	0.410	0.824	0.673	3.974	4.647	i	0.304	0.308	0.612	1.058	4.744	5.801
	g	0.444	0.436	0.879	0.385	5.385	5.770	rg	0.359	0.359	0.718	0.673	4.103	4.776
	b	0.492	0.487	0.979	0.961	6.154	7.115	yb	0.129	0.128	0.257	0.0	12.82	12.82
Md	r	2.031	2.026	4.056	3.269	12.69	15.96	i	1.448	1.436	2.883	1.25	7.051	8.301
	g	2.538	2.538	5.077	2.885	28.97	31.86	rg	1.703	1.692	3.396	3.173	8.846	12.02
	b	2.439	2.462	4.9	4.327	25.9	30.22	yb	0.845	0.846	1.691	1.923	37.69	39.62
Ma	r	0.697	0.692	1.39	1.731	4.615	6.346	i	0.718	0.718	1.436	1.346	6.538	7.885
	g	1.253	1.256	2.51	2.596	7.179	9.776	rg	0.788	0.795	1.583	1.827	5.256	7.083
	b	0.999	1	2	1.923	6.026	7.949	yb	0.539	0.538	1.077	0.288	5.0	5.288
Ud	r	25.96	26.67	52.63	29.13	28.59	57.72	i	14.9	15.26	30.16	13.94	15.26	29.2
	g	20.19	20.38	40.58	20.58	21.03	41.6	rg	22.6	22.05	44.65	22.4	24.36	46.76
	b	35.67	35.64	71.31	37.12	34.1	71.22	yb	39.52	39.74	79.26	38.37	39.74	78.11
Ua	r	15.38	15.26	30.64	15.19	16.03	31.22	i	15.96	16.03	31.99	16.06	16.15	32.21
	g	10.67	11.15	21.83	10.29	12.44	22.72	rg	12.98	13.21	26.19	14.62	14.36	28.97
	b	19.42	19.23	38.65	20.1	19.36	39.46	yb	18.56	18.46	37.02	18.85	20.77	39.62
P	r	15.58	15.85	31.43	15.87	15.73	31.59	i	11.25	10.47	21.72	11.57	10.64	22.21
	g	11.03	10.81	21.84	11.15	11.84	22.99	rg	12.88	12.69	25.58	13.46	13.42	26.88
	b	19.9	21.07	40.97	20	21.28	41.28	yb	21.47	21.5	42.97	21.22	22.05	43.27
G	r	2.938	2.915	5.853	4.199	6.111	10.31	i	1.268	1.282	2.55	2.019	1.581	3.6
	g	2.175	2.154	4.328	2.853	5.684	8.536	rg	1.97	1.94	3.91	3.045	3.59	6.635
	b	5.187	5.171	10.36	6.154	13.55	19.7	yb	6.493	6.479	12.97	7.885	15.3	23.18

The values in the table indicate the FAR, FRR and Total Error Rates (TER), i.e. the sum of false rejection and false acceptance rates. In general as one would expect, for matched protocols the performance is better than for the unmatched protocols due to generalisation problem posed by the latter. There appears to be no outright winner among the considered image spaces. A comparison of the results in different spaces shows that although in some cases the chromaticity or opponent spaces are superior, the most consistent results are obtained in the intensity space. The main exception here is the Ua protocol where significantly better performance is obtained in the normalised green space (g). Moreover, among the chromaticity spaces, the (g) space gives the best performance overall. These results also demonstrate that among the opponent chromaticity spaces, the *rg* space leads to better results. Therefore, for the fusion study, we compare the original *R,G,B* spaces with the intensity (i), normalised green (g) and normalised red-green *(rg)* spaces.

To investigate the effect of combining different colour channel classifiers, we adopted the two simple methods of fusion, the average (Decision 1) and the best (Decision 2) rules, discussed in Section 3. The associated results are shown in Table 4. In this table $\mathcal{F}1$ and $\mathcal{F}2$ refer to combining i,g,rg and *R,G,B* spaces respectively.

Table 4. Decision level fusion results using the averaging and the best rules. $\mathcal{F}1$: i,g,rg, $\mathcal{F}2$:R,G,B channels.

		Decision 1						Decision 2					
		Evaluation			Test			Evaluation			Test		
		FAR	FRR	TER	FAR	FRR	TER	FAR	FRR	TER	FAR	FRR	TER
Mc	$\mathcal{F}1$	0.049	0.051	0.10	0.192	1.667	1.859	0.201	0.205	0.406	0.192	2.564	2.756
	$\mathcal{F}2$	0.228	0.231	0.459	0.288	4.103	4.391	0.205	0.205	0.410	0.288	4.872	5.16
Md	$\mathcal{F}1$	0.387	0.384	0.771	0.192	8.59	8.782	1.004	1.0	2.004	1.346	8.333	9.679
	$\mathcal{F}2$	0.955	0.949	1.904	0.769	6.154	6.923	1.086	1.103	2.189	0.865	6.282	7.147
Ma	$\mathcal{F}1$	0.179	0.179	0.359	0.480	1.667	2.147	0.503	0.513	1.016	0.865	3.333	4.199
	$\mathcal{F}2$	0.509	0.513	1.022	1.154	5.897	7.051	0.613	0.615	1.228	1.635	5.897	7.532
Ud	$\mathcal{F}1$	12.69	12.31	25	12.98	12.56	25.54	16.44	16.03	32.47	16.83	16.15	32.98
	$\mathcal{F}2$	14.62	14.36	28.97	14.33	14.1	28.43	15.96	16.41	32.37	15.96	16.79	32.76
Ua	$\mathcal{F}1$	6.923	6.923	13.85	7.212	8.974	16.19	9.519	9.103	18.62	10.67	10.9	21.57
	$\mathcal{F}2$	14.42	13.97	28.4	14.52	13.97	28.49	15.19	15.77	30.96	15.1	15.26	30.35
P	$\mathcal{F}1$	6.891	7.393	14.28	6.827	7.949	14.78	9.84	9.402	19.24	10.48	9.744	20.22
	$\mathcal{F}2$	9.968	10.04	20.01	10.19	10.09	20.28	11.89	11.07	22.96	12.02	10.98	23
G	$\mathcal{F}1$	0.357	0.359	0.716	0.513	0.769	1.282	0.783	0.778	1.561	1.378	1.068	2.447
	$\mathcal{F}2$	0.905	0.897	1.802	1.795	1.282	3.077	1.112	1.12	2.232	2.051	1.197	3.248

These results firstly show that a better performance is obtained by combining the different channel classifiers. Moreover, they clearly demonstrate that in all cases except the Md protocol a significantly better verification rate is achieved by fusing the intensity, chromaticity and opponent chromaticity spaces classifiers ($\mathcal{F}1$). This is due to the enhanced uncorrelatedness of the i,g,rg spaces. The poor quality of the $\mathcal{F}1$ system for the Md protocol is due to the very weak verification performance of the g channel in this scenario. Finally, one can see that between the adopted fusion methods, score averaging is superior.

6 Conclusions

We have investigated the benefits of decorrelating *R,G,B* colour camera channels by separating the effect of object shape and surface material properties. The study was carried out in the context of face verification. The Banca database and the associated protocols were used for performance characterisation.

Individually, the component face verification experts in the transformed colour space (intensity, chromaticity and opponent chromaticity channels) were not necessarily better than those using the original*R,G,B* images. Overall, the performance in the intensity space was more consistent than that of any other channel, but not sufficiently better to justify using a gray level rather than colour camera. However, the advantage of using the physics based methods of decorrelating the sensory data became apparent in the context of multiple expert fusion. Using the i, g and rg component experts in a multiple classifier system improved the performance over the *R,G,B* colour channel expert fusion significantly. Taking the intensity space as the baseline, in the case of the controlled scenario (Mc) an improvement by a factor of three was obtained. Similar gains were achieved for the G protocol.

Acknowledgements

The financial support from the EU project VAMPIRE is gratefully acknowledged.

References

1. E. Bailly-Baillière, S. Bengio, P. Bimbot, M. Hamouz, J. Kittler, J. Mariéthoz, J. Matas, K. Messer, V. Popovici, F. Porée, B. Ruiz, and J.-P. Thiran. The BANCA database and evaluation protocol. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication, AVBPA,* pages 625–638. Springer-Verlag, 2003.
2. G. Giacinto, F. Roli, and G. Fumera. Design of effective multiple classifier systems by clustering of classifiers.; In *15th International Conference on Pattern Recognition, ICPR00,* pages Vol II: 160–163, 2000.
3. K. Jonsson, J. Kittler, Y. Li, and J. Matas. Support vector machines for face authentication. In *T. Pridmore and D. Elliman, editors, Proceedings of BMVC'99,* pages 543– 553, 1999.
4. K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre. XM2VTSDB: The extended M2VTS database. In *Second International Conference on Audio and Video-based Biometric Person Authentication,* pages 72–77, March 1999.
5. C. Nastar and M. Mitschke. Real-time face recognition using feature combination. In *International Conference on Automatic Face and Gesture Recognition, AFGR98,* pages 312–317, 1998.
6. F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. In *Second International Workshop on Multiple Classifier Systems,* pages 78–87, Cambridge, UK, 2001.
7. A. Ross and A. K. Jain. Information fusion in biometrics. *Pattern Recognition Letters,* 24(13):2115–2125, Sep 2003.
8. M. Sadeghi and J. Kittler. Data fusion in face verification. In *Second COST 275 Workshop, Biometrics on the Internet: Fundamentals, Advances and Applications,* Vigo, Spain, 25-26 March 2004.
9. M. Sadeghi and J. Kittler. Decision making in the LDA space: Generalised gradient direction metric. In *the 6th International Conference on Automatic Face and Gesture Recognition,* Seoul, Korea, May 2004.
10. R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning,* pages 322–330. Morgan Kaufmann, 1997.
11. L. Torres, J.Y. Reutter, and L. Lorente. The importance of the color information in face recognition. In *International Conference on Image Processing, ICIP99,* pages 627–631, 1999.

High Security Fingerprint Verification by Perceptron-Based Fusion of Multiple Matchers

Gian Luca Marcialis and Fabio Roli

Department of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
{marcialis,roli}@diee.unica.it

Abstract. Recent works about perceptron-based fusion of multiple fingerprint matchers showed the effectiveness of such approach in improving the performance of personal identity verification systems. However, to the best of our knowledge, no previous work investigated such fusion approach when stringent requirements in terms of verification errors are given, and the number of available samples for perceptron training is small. Such investigation can allow to understand for which kind of applications such fusion rule can be useful. Reported experiments, based on two benchmark data sets, show that perceptron-based fusion can be useful for high security fingerprint verification applications, and it is effective in small-sample-size realistic cases.

1 Introduction

Fingerprints are widely used to automatically grant or deny the access to restricted areas (personal identity authentication or verification) [1]. The person to be authenticated submits to the system her/his fingerprint and declares her/his identity. The system matches the given fingerprint with the one stored in its data base and associated to the claimed identity. A matching score, i.e., a similarity degree among the two fingerprints, is computed. The higher such score, the higher the degree of similarity. The final decision about the claimed identity is performed by a threshold-based approach. In particular, if the score is higher than the given acceptance threshold, the claimed identity is accepted, and the person is classified as a genuine user. Otherwise, she/he is rejected, and classified as an impostor.

The so-called verification errors strictly depend on such threshold. In particular, the rate of impostors accepted by the system, also called false acceptance rate (FAR), and the rate of genuine users rejected by the system, also called false rejection rate (FRR), are the two performance evaluation parameters by which the effectiveness of the acceptance threshold is checked.

It is worth noting that high security applications, as the access control to nuclear power stations, require that the FRR/FAR value has to be the lowest as possible for a fixed value of FAR/FRR, usually very low too. As an example, the FAR value could be fixed to 1%. The FRR value should be minimized by keeping FAR=1%. However, it is very difficult to design a fingerprint matcher able to meet such stringent requirement. Usually, the resulting FAR or FRR value, by keeping the other parameter fixed to 1%, is unacceptable for the given application.

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 364–373, 2004.

In order to improve the performance of automatic fingerprint verification systems [2], the decision-level fusion of multiple fingerprint matchers has been recently proposed, and such research topic is still very active [3-7]. Among the other approaches, the decision-level fusion by perceptron neural net has been investigated [4-6]. In particular, it has been shown that such kind of fusion can outperform the best individual matcher, and also some fixed fusion rules [5-6]. However, verification performances, when a small data set is available for perceptron training, have not yet been investigated in detail. It is worth noting that the small sample size problem is a realistic condition for any automatic verification system. This becomes a crucial issue under high security requirements.

In this paper, we experimentally investigated the performance improvement achievable by perceptron-based fusion of two well-known fingerprint matchers, when stringent requirements in terms of FAR and FRR are used, and the number of available training samples decreases. Results on two widely used benchmark data sets, namely, the FVC2000-DB1 and the FVC2000-DB2 data sets [2], are reported.

The paper is organized as follows. Section 2 describes the selected individual fingerprint matchers. Section 3 describes our fusion approach, by focusing on the perceptron-based fusion rules. Section 4 reports experimental results. Section 5 concludes the paper.

2 The Selected Individual Algorithms for Fingerprint Verification

A fingerprint verification algorithm, also called "matcher", performs the comparison between the fingerprint submitted by the person to be authenticated, and the one stored in its data base (the so-called "template"). The comparison is performed by the matching algorithm, which provides a similarity degree among fingerprints. Such similarity degree is a real value in [0,1], named "score".

The literature reports two main kinds of approaches to fingerprint verification: the minutiae-based ones and the filterbank-based ones [3, 8, 9]. The first ones try to exploit the local features of fingerprints, while the second ones are focused on global features. As results achieved in other application fields support the hypothesis that combining information coming from different algorithms can substantially improve fingerprint verification performances, we selected one algorithm for each kind. For a review about other fingerprint verification approaches, the reader is referred to Ref. 1.

The minutiae-based matchers are based on the location and orientation of the so-called minutiae points. The minutiae-points are the terminations and the bifurcations of the ridge lines (Figure 1). The orientation of a minutia point is defined as the local orientation of the ridge which the minutia belongs to.

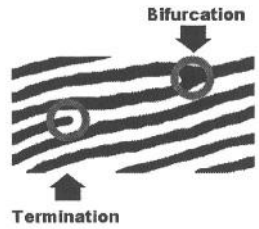


Fig. 1. The so-called minutiae-points: bifurcations and terminations of the ridge lines.

The so-called "String" algorithm is a widely used minutiae-based approach [8]. In the following, we summarize such algorithm. Let X be the template minutiae set. Let Y be the input minutiae set. For each minutia $x \in X$, the following algorithm is performed. For each $y \in Y$, x is aligned to y. Such alignment implies the roto-translation of the input minutiae-set while x and y match perfectly. Let $A(x, y) = \{(x_i, y_i), x_i \in X, y_i \in Y : aligned(x_i, y_i) = true\}$ be the set of other couples of aligned minutiae. x_i and y_i are considered as aligned on the basis of a pre-defined "minutiae distance" not exceeding a certain fixed threshold. At the end of such loops, the value $\max_{x,y}\{|A(x, y)|\}$ is converted into a matching score by the formula:

$$score = \frac{\left(\max\{|A(x, y)|\}\right)^2}{|X| \cdot |Y|} \tag{1}$$

The filterbank-based matchers try to describe the fingerprint texture by processing the fingerprint image with bank of filters aimed to enhance the different orientations of the ridge flow (Figure 2).

Fig. 2. (a) Original fingerprint image, (b) Filtered fingerprint images enhancing three different ridge lines orientations by three Gabor filters.

In particular, we used the so called "Filter" algorithm [9], which is based on a set of Gabor filters. A Gabor filter is a band-pass filter with orientation-selective characteristics. It has been shown that such filters are very effective to enhance the fingerprint texture [10]. In [9], a tessellation of the fingerprint image is firstly defined. The tessellation is constituted by a circle subdivided in a certain number of sectors. Such circle is centered on the fingerprint "core"[1](Figure 3).

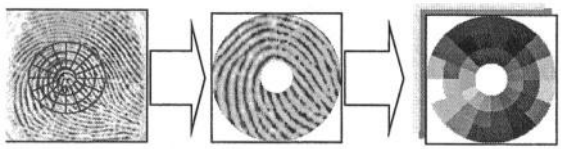


Fig. 3. The main steps of the fingercode feature vector generation. A circle decomposed in a certain number of sectors is centered on the core of the fingerprint image. A set of Gabor filters is applied to such tessellated image. The standard deviation of the gray level values of each sector of the filtered image are computed, so generating the fingercode feature vector [9].

[1] The fingerprint core is the point around which many ridge lines converge (Figure 3).

A set of Gabor filters is applied to such tessellation, so producing a set of filtered and tessellated images. The standard deviation of the grey values in each sector of such images is computed. The obtained values constitutes a feature vector named "fingercode" [9]. By such approach, both template and input fingerprints are represented by a fingercode. So, the matching phase can be performed by simply computing the Euclidean distance among such fingercodes. The obtained distance is finally converted into a matching score.

3 Decision-Level Fusion of the Selected Fingerprint Matchers

The proposed fusion scheme is outlined in Figure 4.

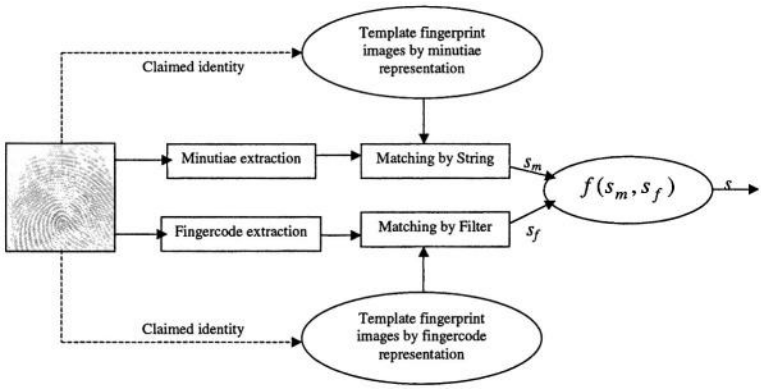


Fig. 4. The proposed decision-level fusion scheme of the two selected fingerprint matchers.

Let s_m and s_f be the matching scores provided by the minutiae-based and the filter-bank-based matching algorithms, respectively:

- Apply the following transformation to the above scores s_m and s_f to implement the fusion:

$$s = f(s_m, s_f) \tag{2}$$

- Compare the obtained score value s with a threshold. The claimed identity is classified as "genuine user" if:

$$s > threshold \tag{3}$$

otherwise it is classified as "impostor".

It is easy to see that the above methodology can be also used for the case of more than two matchers.

Among the fusion rules which can implement eq. (2), the most simple ones are the so-called fixed fusion rules [11]. This term derives from the observation that such rules require no training parameters. In other words, eq. (2) can be implemented by simply combining the given matching scores without other kind of information.

Among fixed rules, we selected the mean rule:

$$s = \frac{s_m + s_f}{2} \tag{4}$$

and the product rule:

$$s = s_m \cdot s_f \tag{5}$$

The perceptron-based fusion rules belong to the class of the so-called trained rules. In fact, eq. (2) is implemented by the logistic transformation:

$$s = \frac{1}{1+\exp\left[-\left(w_0 + w_1 s_m + w_2 s_f\right)\right]} \tag{6}$$

The logistic transformation parameters (w_0, w_1, w_2), also called weights, are usually computed by a gradient descent algorithm with a cross-entropy loss function, or with a sum square error loss function [12].

Recently, Marcialis and Roli [6] proposed a novel approach to address the weights optimisation problem by using the following loss function:

$$FD = \frac{\left(\mu_{gen} - \mu_{imp}\right)^2}{\sigma^2_{gen} + \sigma^2_{imp}} \tag{7}$$

The above loss function is called Fisher distance. In eq. (7), μ_{gen}, σ^2_{gen} and μ_{imp}, σ^2_{imp} are the mean and the variances defined as follows:

$$\mu_i = \frac{1}{N_i}\sum_{j=1}^{N_i} s_j^{(i)}\ ,\ \sigma_i^2 = \frac{1}{N_i - 1}\sum_{j=1}^{N_i}\left(s_j^{(i)} - \mu_i\right)^2\ ,\ i \in \{gen, imp\} \tag{8}$$

being $s_j^{(i)}$ the j-th output of eq. (6) for class i. Our learning algorithm looks for weights of logistic transform that maximise the Fisher distance by a gradient-descent algorithm. Accordingly, the distributions of genuine and impostor classes will be separated in terms of distance between their means, while their variances will be reduced. The rationale behind such optimisation approach can be summarised as follows. If genuine and impostor classes are well separated, errors that usually affect estimates of class distributions have a smaller impact on threshold selection, and consequently, on system performances. Further details and experiments showing the benefits of this fusion approach can be found in [6].

4 Experimental Results

4.1 The Data Sets

We used the FVC2000-DB1 and DB2 data sets, proposed for the Fingerprint Verification Competition [2]. Each data set is made up of 800 fingerprint images acquired by a different fingerprint image capture device. The number of identities is 100, and the number of images per identity is eight.

The capture device used for collecting the DB1 data set is an optical sensor. The optical sensor is characterised by a LED light source and a CCD placed on the side of a glass platen, on which the fingerprint to acquire is placed. The LED illuminates the fingerprint and the CCD captures the light reflected from the glass, enhancing ridges and valleys of the given fingerprint. The image size is 300x300 pixels at 500 dpi.

The capture device used for the DB2 data set is a capacitive sensor. The core of the capacitive sensors is the sensing surface, which is made up of a two-dimensional array of silicon capacitor plates. The second plates are considered to be the finger skin. The capacitance is dependent on the distance between the finger skin, i.e. ridges and valleys, and the plates. The captured image is derived from the capacitance measures from each array element. The image size is 256x364 pixels at 500 dpi.

Figures 5 and 6 show examples of fingerprint images from the DB1 and DB2 data sets.

Fig. 5. Examples of fingerprint images from the FVC2000-DB1 data set.

Fig. 6. Examples of fingerprint images from the FVC2000-DB2 data set.

4.2 Experimental Protocol

Our experimental protocol is very similar to the one proposed in the FVC competition [2]:

- For each matcher, or any combination of the two selected matching algorithms, we computed two sets of scores. The first one is the "genuine-matching scores" set G, made up of all comparisons among fingerprints of the same identity (in our experiments, all images were compared with all the other images of a given identity). The second set is the "impostor matching scores" set I, made up of all comparisons among fingerprints of different identities.
- We randomly subdivided the above sets in four parts, so that: $G=G1 \cup G2$, $I=I1 \cup I2$. $G1$ and $G2$, as well as $I1$ and $I2$, are disjoint sets.
- The training set $Tr=\{G1, I1\}$ was used to compute the weights of the perceptron-based fusion rules.
- The test set $Tx=\{G2, I2\}$ was used to assess the performance of the individual and combined algorithms on unknown patterns.
- We performed our experiments on five different training-test couples (i.e. on five different partitions of I and G): $\{Tr_1, Tx_1\},\ldots,\{Tr_5, Tx_5\}$, and averaged the results.

Let $|Tr|$ and $|Tx|$ be the size of the Tr and Tx sets, respectively. In order to simulate the effect of decrease of training set size, the perceptron has been trained and tested in three experimental conditions:

(1) $|Tr|/|Tx|=1$, that is, Tr and Tx sets have the same size;
(2) $|Tr|/|Tx|=1/2$, so that the Tx set size is twice as big as that of Tr;
(3) $|Tr|/|Tx|=1/3$, so that the Tx set size is three times as big as that of Tr.

In all cases, $TrUTx = IUG$. Individual algorithms and fixed fusion rules have also been tested according to the above protocol.

The high security fingerprint verification performances of the individual fingerprint matchers, and the different combinations of the two selected algorithms, were assessed and compared in terms of the following parameters:

(a) 1%FAR, which is the rate of rejected genuine users (false rejection rate, FRR) when the rate of accepted impostors (false acceptance rate, FAR) is fixed to 1%;
(b) 1%FRR, which is the FAR when the FRR is fixed to 1%.

4.3 Results

Tables 1-2 show the average 1%FAR and 1%FRR for the individual algorithms (second and third rows), the fixed fusion rules (fourth and fifth rows) and the perceptron-based fusion rules on the FVC2000-DB1 data set. Each column is related to the size ratio between training set and test set.

It is worth noting that the best individual algorithm and the fixed fusion rules perform worst by decreasing the training set size (second, third, fourth columns). In particular, fixed fusion rules perform progressively worse than the best individual algorithm. On the contrary, the perceptron-based fusion allows to obtain a performance definitely better than that of the best individual matcher, even if the training set/test set size ratio decreases. As an example, the perceptron trained by cross-entropy loss function exhibits the most notable trade-off between 1%FAR and 1%FRR. A performance improvement of 2% over the best individual algorithm is pointed out for $|Tr|/|Tx| = 1/3$ (Tables 1-2).

Table 1. Average 1%FAR percentage values on the FVC2000-DB1 test set for decreasing values of $|Tr|/|Tx|$. In the first column, "Perceptron-CE" refers to the use of a cross-entropy loss function, "Perceptron-SSE" refers to the use of a sum square error loss function, "Perceptron-FD" refers to the use of the Fisher distance loss function.

	1%FAR		
	$\|Tr\|/\|Tx\| = 1$	$\|Tr\|/\|Tx\| = ½$	$\|Tr\|/\|Tx\| = 1/3$
String	2.86	3.21	3.22
Filter	16.89	17.25	17.35
Fusion by Mean	3.69	3.97	4.13
Fusion by Product	*1.78*	3.97	4.13
Fusion by Perceptron-CE	*1.72*	***1.85***	***1.70***
Fusion by Perceptron-SSE	***1.69***	*1.88*	*1.76*
Fusion by Perceptron-FD	*1.72*	*1.86*	*2.15*

Table 2. Average 1%FRR percentage values on the FVC2000-DB1 test set for decreasing values of $|Tr|/|Tx|$. In the first column, "Perceptron-CE" refers to the use of a cross-entropy loss function, "Perceptron-SSE" refers to the use of a sum square error loss function, "Perceptron-FD" refers to the use of the Fisher distance loss function.

	1%FRR		
	$\|Tr\|/\|Tx\| = 1$	$\|Tr\|/\|Tx\| = ½$	$\|Tr\|/\|Tx\| = 1/3$
String	3.76	5.01	5.09
Filter	45.56	46.56	43.87
Fusion by Mean	17.54	16.85	13.54
Fusion by Product	***2.38***	16.85	13.54
Fusion by Perceptron-CE	*3.65*	*3.80*	*2.91*
Fusion by Perceptron-SSE	*4.74*	*4.07*	***2.65***
Fusion by Perceptron-FD	*3.06*	***3.05***	*3.13*

Table 3. Average 1%FAR percentage values on the FVC2000-DB2 test set for decreasing values of $|Tr|/|Tx|$. In the first column, "Perceptron-CE" refers to the use of a cross-entropy loss function, "Perceptron-SSE" refers to the use of a sum square error loss function, "Perceptron-FD" refers to the use of the Fisher distance loss function.

	1%FAR		
	$\|Tr\|/\|Tx\| = 1$	$\|Tr\|/\|Tx\| = ½$	$\|Tr\|/\|Tx\| = 1/3$
String	4.42	5.10	5.37
Filter	35.76	37.83	37.47
Fusion by Mean	14.45	14.85	15.30
Fusion by Product	*4.27*	14.85	15.30
Fusion by Perceptron-CE	*4.09*	***4.36***	***4.54***
Fusion by Perceptron-SSE	***4.06***	***4.36***	*4.62*
Fusion by Perceptron-FD	*4.12*	*4.38*	*4.60*

Table 4. Average 1%FRR percentage values on the FVC2000-DB2 test set for decreasing values of $|Tr|/|Tx|$. In the first column, "Perceptron-CE" refers to the use of a cross-entropy loss function, "Perceptron-SSE" refers to the use of a sum square error loss function, "Perceptron-FD" refers to the use of the Fisher distance loss function.

	1%FRR		
	$\|Tr\|/\|Tx\| = 1$	$\|Tr\|/\|Tx\| = ½$	$\|Tr\|/\|Tx\| = 1/3$
String	31.70	43.90	44.04
Filter	88.73	87.40	84.72
Fusion by Mean	60.47	59.08	56.46
Fusion by Product	*18.89*	59.08	56.46
Fusion by Perceptron-CE	***18.16***	*20.62*	***21.57***
Fusion by Perceptron-SSE	*18.17*	*20.18*	*22.85*
Fusion by Perceptron-FD	*19.24*	***19.83***	*27.91*

Tables 3-4 show the average 1%FAR and 1%FRR for the individual algorithms (second and third rows), the fixed fusion rules (fourth and fifth rows) and the perceptron-based fusion rules on the FVC2000-DB2 data set. Each column is related to the size ratio between training set and test set.

Even in this case, the performance improvement obtained by perceptron-based fusion rules are notable with respect to that of the fixed rules, which are not able to outperform the best individual matcher when the training set size decreases (third and fourth columns). In particular, perceptron-based fusion achieved an 1%FRR improvement of 20% over the best individual matcher (Table 4). Such results are particularly relevant because the 1%FRR assesses the verification performance when only the 1% of genuine users can be wrongly rejected.

It is worth noting that the small sample size issue should advise against the use of more complex, trained, fusion rules for performances improvement, as remarked in [3]. Therefore, three main observations can be made on the basis of this remark and the reported results:

- the individual algorithms do not exhibit satisfying performances under high security requirements (see in particular Table 4, second and third rows);
- the use of fixed fusion rules can be inappropriate for improving performances, especially under the increase of the user population, that is, the reduction of the training set size w.r.t. the test set size (Table 1-4, fourth and fifth columns);
- on the other hand, the perceptron-based fusion appears to exhibit the best trade-off among high security requirements, the increase of the user population, and the available sample size for weights optimisation. In addition, a perceptron-based fusion approach is very simple, so that the overall system complexity does not significantly increase.

5 Conclusions

In this paper, we experimentally investigated the performance improvement achievable by perceptron-based fusion of two well-known fingerprint matchers when stringent requirements in terms of FAR and FRR are required and the available training set is small. Experiments with two widely used benchmark data sets have been performed. Reported results showed that the perceptron-based fusion rules allow outperforming the best individual matcher and some fixed fusion rules. This result is notable because realistic conditions for fingerprint verification systems, that is, small training sets, have been simulated.

Although further experiments are needed to draw definitive conclusions, the perceptron-based fusion approach appears to exhibit the best trade-off among high security requirements, the increase of the user population and the available sample size for weights optimisation.

Acknowledgments

This work was partially supported by the Italian Ministry of University and Scientific Research (MIUR) in the framework of the research project on distributed systems for multisensor recognition with augmented perception for ambient security and customisation.

References

1. D. Maltoni, D. Maio, A.K Jain, and S. Prabhakar, Handbook of Fingerprint Recognition, Springer Verlag, 2003.
2. D. Maio, D. Maltoni, R.Cappelli, J.L. Wayman, and A.K. Jain, FVC-2000: Fingerprint Verification Competition. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) 402-412,2002.
3. S. Prabhakar and A.K. Jain, Decision-level Fusion in Fingerprint Verification. Pattern Recognition 35 (4) 861-874, 2002.
4. A.K. Jain, S. Prabhakar, and S. Chen, Combining Multiple Matchers for a High Security Fingerprint Verification System. Pattern Recognition Letters, 20 (11-13) 1371-1379,1999.
5. G.L. Marcialis and F. Roli, Experimental results on Fusion of Multiple Fingerprint Matchers. Proc. 4th Int. Conf. on Audio and Video-Based Person Authentication AVBPA03, J. Kittler and M.S. Nixon Eds., Springer LNCS2688, pp. 814-820, 2003.
6. G.L. Marcialis and F. Roli, Perceptron-based Fusion of Multiple Fingerprint Matchers. Proc. First Int. Work. on Artificial Neural Networks in Pattern Recognition ANNPR03, M. Gori and S. Marinai Eds., pp. 92-97, 2003.
7. A. Ross, A.K. Jain, and J. Reisman, A Hybrid Fingerprint Matcher. Pattern Recognition, 36 (7) 1661-1673, 2003.
8. A.K. Jain, L. Hong, and R. Bolle, On-line Fingerprint Verification. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4) 302-314, 1999.
9. A.K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, Filterbank-based Fingerprint Matching. IEEE Transactions on Image Processing, 9 (5) 846-859, 2000.
10. J. Yang, L. Liu, T. Jiang, and Y. Fan, A modified Gabor filter design method for fingerprint image enhancement, Pattern Recognition Letters, 24 1805-1817, 2003.
11. J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20 (3) 226-239, 1998.
12. C.M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995.

Second Guessing a Commercial 'Black Box'Classifier by an 'In House' Classifier: Serial Classifier Combination in a Speech Recognition Application

Fuad Rahman*, Yuliya Tarnikova, Aman Kumar, and Hassan Alam

BCL Technologies Inc. 990 Linden Dr., Suite #203
Santa Clara CA 95050, USA
{fuad,yuliyat,amank,halam}@bcltechnologies.com
http://www.bcltechnologies.com

Abstract. We describe how an *'in house'* classifier can enhance the performance of a commercial *'black box'* classifier using the classic serial multiple classifier combination scheme. It is now acknowledged by the classifier combination community that parallel or hybrid decision fusion algorithms, in general, outperform serial combination schemes. However, classifier combination using techniques that use class labeling, ranking or probability estimators need access to low level information supplied by all of the participating classifiers. Unfortunately, in many commercial applications the classifier is often a 'black box', which implies that it is not possible to manipulate the low level information regarding classification for these classifiers. In many such cases, a serial classifier combination model provides the only practical method to improve classification. In this paper, we present such an application in speech recognition.

1 Introduction

Commercial classifiers use a variety of ways to rank the input patterns to maximize recognition efficiency. Unfortunately, the optimization details are often not available, and in many cases are targeted to offer a performance best suited for an average user scenario. In other words, they are optimized to work *reasonably* well in a varied user environment. In many cases, optimizing these commercial recognizers for specific user requirements is extremely difficult, as most of the low level handles of the algorithms are not publicly available for manipulation. This scenario calls for using secondary and preferably tertiary classifiers to enhance overall recognition performance. In many large commercial systems, unfortunately, it is not economically viable to design classifiers from scratch due to time constraints, unavailability of skills and/or spiraling development costs. In most of these cases, the solution is to acquire one reasonably good Commercial-Off-the-Shelf (COTS) classifier. These second (and may be the third) classifiers are usually selected from what is freely available, and most often than not, are derived from academic research. Combining these classifiers with the COTS classifier then provides a major challenge.

In this paper we discuss such a practical scenario. The focus of the paper is to offer insights into how the applied research of the commercial world differs in its ap-

* Corresponding author

F. Roli, J. Kittler, and T. Windeatt (Eds.): MCS 2004, LNCS 3077, pp. 374–383, 2004.

proaches from the rigorous research of the academic world. This paper is an attempt to show how it is possible to apply some of the classifier combination techniques, albeit within restricted domains, to streamline an often ad-hoc approach. The chosen scenario is treated as typical, since many commercial systems face similar demands for performance enhancement without the benefit of detailed access to low level parameters.

Specifically, we demonstrate how it is possible to use Support Vector Machines (SVMs) to optimize the performance of a commercial voice recognition engine (IBM ViaVoice™: http://www-3.ibm.com/software/voice/viavoice/) by using phonetic and other features (Fig. 1).

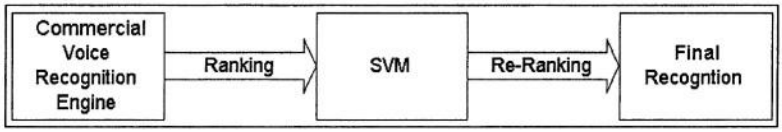


Fig. 1. Re-ranking top choices from a commercial voice recognition engine using Support Vector Machine classifiers

The rest of the paper is organized as follows. Section 2 introduces some terminology within the framework of a multiple classifier system (MCS) paradigm. Section 3 presents a concise summary of previous research on commercial Automatic Speech Recognizers (ASRs). Section 4 introduces the proposed serial MCS. Section 5 presents some results and finally, in Section 6, some conclusions are drawn and future work is discussed.

2 MCSs: Some Terminology

Within the framework of Multiple Classifier Systems (MCS), attempts have been made to categorize these systems based on their hierarchy. Three different decision combination topologies widely reported in the literature [1] are as follows:

- Serial (Vertical) Combination Scheme:
  This realizes to a physical structure where the classifiers are applied sequentially.
- Horizontal Combination Scheme:
  This realizes to a physical structure where the classifiers are applied concurrently and independently.
- Hybrid Combination Scheme:
  In this case, a combination of serial and parallel combinations is constituted.

In this paper, we focus of serial combination schemes.

3 Previous Research

Since this paper is focused on commercial Automatic Speech Recognizers (ASRs), we chose to review this area. IBM® Embedded ViaVoice™ is the leading Automatic Speech Recognition (ASR) software in the market (http://www-3.ibm.com/software/

voice/viavoice/). The voice recognition engine is impressive, and the ability to use Via Voice™ to develop applications for handheld PDAs makes it a valuable tool. However, there is room for improvement in the ViaVoice™ software and in the development model. SRI International offers an embedded ASR, DynaSpeak (http://www.dynaspeak.com/). Some of the newer ASRs have been designed specifically for small screen devices with very optimized and therefore very small footprint. These include Fonix (http://www.fonix.com/), Dragon (http://www.caere.com/naturallyspeaking/) and others.

In this paper, we focus on the ranking of utterances that are candidates to be the top choices by the IBM® ViaVoice™ engine. Internally ViaVoice™ uses a set of parameters to optimize this, and some of these parameters are exposed for manipulation with their SDK. However, these parameters are too generic and are not able to adapt themselves to specific user environments. In the rest of the paper we offer an alternate solution of post-processing using SVM classifiers as black-box solutions to re-rank the top choice offered by the ViaVoice™ solution and demonstrate that in many cases it is possible to produce significant improvements over the response of the ViaVoice™ working on it's own. We also demonstrate how this can result in much improved performance from the ViaVoice™ engine and subsequently to the overall recognition task.

There has been significant amount of work done on serial combination of multiple classifiers. The space restrictions do not allow a full review of this area and the readers are referred to [2] for a comprehensive treatment.

4 Serial Combination: IBM ViaVoice™ and Support Vector Machines (SVM)

We propose a serial combination of the COTS IBM ViaVoice and a Support Verctor Machine (SVM).

4.1 IBM ViaVoice™

IBM® Embedded ViaVoice™ is the leading Automatic Speech Recognition (ASR) software in the market. Currently, all expected input must be pre-specified in a .bnf file in Speech Recognition Command Language (CRCL) format. There are several disadvantages to this development model. The process of designing a grammar requires the programmer to use production rules to generate all of the possible allowable phrases for the application. While this approach works well with simple predetermined commands, ViaVoice™ is not able to recognize speech that varies slightly from the grammar. Furthermore, it is difficult to design a grammar in CRCL format that supports Natural Language (NL) input. Therefore, programmers using ViaVoice™ to develop an NL speech enabled application must have an in-depth background in linguistics,

The process of using ViaVoice™ to create a speech-enabled application is complex, requiring the programmer to complete many steps in order to compile a simple "Hello World" program. Between setting environment variables, keeping track of annotation markers, and defining the AppEvent in 4 or more places, the ViaVoice™

learning curve is quite steep. Also, ViaVoice™ is not easily compatible with the Microsoft® Visual Studio environment, nor does ViaVoice™ provide any development automation software.

Fig. 2 shows a sample set of responses of IBM® ViaVoice™ to a typical input sentence. The input utterance is "Dial 5581230" and the top nine candidates are shown. The first part of the response shows the text part, the second number is an IBM-internal confidence score and the third number (0/1) is a Boolean response of acceptance, where '0' means rejection and '1' means acceptance. It is interesting to see that although the top choice was correct, the engine chose to reject it. In this example, no choice was accepted.

Sentence:
1 Dial 5581230
Results:
dial five five eight one two three zero, 4842842, 0
dial five five eight one two three zero, 4841823, 0
dial five five eight one two two zero, 4859335, 0
dial five five eight one two two zero, 4860354, 0
dial five nine eight one two three zero, 4903933, 0
dial five nine eight one two three zero, 4904952, 0
dial five one eight one two three zero, 4911739, 0
dial five one eight one two three zero, 4912758, 0
dial five four eight one two three zero, 4913812, 0

Fig. 2. Typical input utterance and response from the voice recognition engine

4.2 Support Vector Machine (SVM)

SVM is being extensively studied in recent years because of its ability to create very accurate decision boundary in multi-dimensional feature spaces [3-6]. We chose SVM as our secondary classifier because of its ability to generate robust global minima in a multi-dimensional feature space. The following is a generalized description of the principal concepts in an SVM based largely on [7].

A function $f:\mathrm{R}^N \rightarrow \{\pm 1\}$ is estimated from the training data, i.e. for N-dimensional patterns x_i and class labels $y_1, (x_1, y_1),..,(x_l, y_l) \in R_N X \{(+/-)1\}$, such that f will correctly classify new examples (x,y) - that is, $f(x) = y$ for examples (x,y), which were generated from the same underlying probability distribution $P(x,y)$ as the training data. If no restriction is put on the class of functions that f is chosen from, a function that does well on the training data may not generalize well to unseen examples. Assuming no prior knowledge about f, the values on the training patterns carry no information whatsoever about values on novel patterns. Therefore learning is not possible, and minimizing the training error does not necessarily imply a small test error. Statistical learning theory, or VC (Vapnik-Chervonenkis) theory, shows that it is crucial to restrict the class of functions that the learning machine can implement to one with a capacity that is suitable for the amount of available training data.

To design solid learning algorithms, a class of functions is required whose capacity can be computed. Support Vector classifiers are based on the class of hyperplanes $(w.x) + b = 0$, $w \in \mathrm{R}_N$, $b \in \mathrm{R}$ corresponding to decision functions $f(x) = sign((w.x) + b)$. It is possible to show that the optimal hyperplane, defined as the one with the

maximal margin of separation between the two classes, has the lowest capacity. It can be uniquely constructed by solving a constrained quadratic optimization problem whose solution *w* has an expansion w $\sum_i v_i x_i$ in terms of a subset of training patterns that lie on the margin. These training patterns, called support vectors (SV), carry all relevant information about the classification problem. One crucial property of the algorithm is that both the quadratic programming problem and the final decision function depend only on dot products between patterns. This makes it easier to generalize this solution to the nonlinear case. SV machines map the data into some other dot product space (called the feature space) F via a nonlinear map $\Phi: R^N \rightarrow F$, and perform the above linear algorithm in *F*, which only requires the evaluation of dot products $k(x, y)=(\Phi(x).\Phi(y))$. If F is high dimensional, the right-hand side of this equation is very expensive to compute. In some cases, however, there is a simple kernel k that can be evaluated efficiently.

For instance, the polynomial kernel $k(x, y)=(x.y)^d$ can be shown to correspond to a map F into the space spanned by all products of exactly d dimensions of R^N. For $d=2$ and x, $y \in R^2$, for example, $(x.y)^2 = \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\begin{pmatrix} y \\ y_2 \end{pmatrix}\right)^2 = \left(\begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}\begin{pmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{pmatrix}\right) = (\Phi(x)\Phi(y))$, defining $\Phi(x)=\left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$. More generally, it can be shown that for every kernel that gives rise to a positive matrix $\left(k(x_i, x_j)\right)_{ij}$, a map F can be constructed such that this equation holds. It is also possible to use radial basis function (RBF) kernels such as $k(x, y)=\exp\left(-\|x-y\|^2/(2\sigma^2)\right)$ and sigmoid kernels (with gain κ and offset Θ), $k(x,y)=\tanh(\kappa(x.y)+\Theta)$. Support Vector Machines use a nonlinear decision function of the form $f(x)=sign\left(\sum_{i=1}^{l} v_i.k(x, x_i)+b\right)$. We have adopted an implementation of ***SVM^{light}***, which is an implementation of Vapnik's Support Vector Machine [8]. The optimization algorithm used in ***SVM^{light}*** is described in [9]. The algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently. All SVMs discussed in this paper assumes a Dot Product function implementation.

4.3 Manipulating SVM Features

What we are looking for here is a post-processing black-box that either confirms or rejects the decision of the ViaVoice™ engine. In case of rejection, it should re-rank the choices according to its best judgment. Fig. 3 shows an example utterance and the response generated by the engine.

4.3.1 Signal Features

The first column in Fig. 3 shows the candidates. The second column shows the Boolean decision by the engine. All the candidates are rejected in this case. The third column shows internal confidence scores of the engine. The fourth column shows the normalized scores. Finally, the fifth column shows the differences between two successive scores. Fig. 4 shows the graphical representation of the differences. These scores and their combinations are used as signal features for the SVM.

view Jennifer Peters				
Results:				
view Jennifer Peters	0	2600051	0	0.699648
view Jennifer DAVIDSON	0	2785921	0.699648425	0.117657
view Julie Fruit	0	2817178	0.817305448	0.024497
view JASON Curtis	0	2823686	0.841802742	0.051178
call Jennifer Peters	0	2837282	0.892980554	0.03279
view JULIE Jones	0	2845993	0.92577034	0.041632
view Dennis Dent	0	2857053	0.967402188	0.012
view Stacy Perkins	0	2860241	0.979402399	0.020598
view Jamie Jaggers	0	2865713	1	

Fig. 3. SVM Features

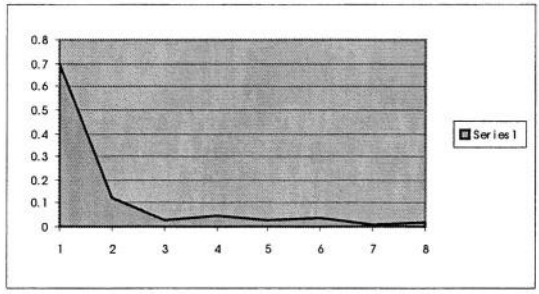


Fig. 4. SVM Features

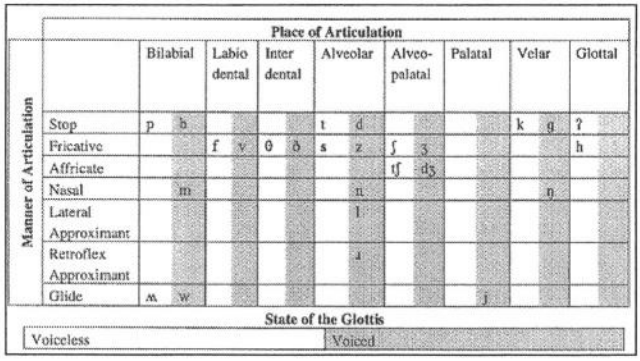


Fig. 5. IPA definition of Standard American English - Consonants

4.3.2 Phonetic Features

We use International Phonetic Alphabet (IPA) definitions (http://www.ic.arizona.edu/~lsp/IPA/SSAE.html) for generating phonetic features. Fig. 5 and Fig. 6 concentrate on the sounds of Standard American English. These charts are done in the standard IPA. Our phonetic features are based on manipulating these notations and values. More information is in Section 4.3.3.

4.3.3 Final Combined Features

Our features combined the signal and the phonetic features. The features we used included:

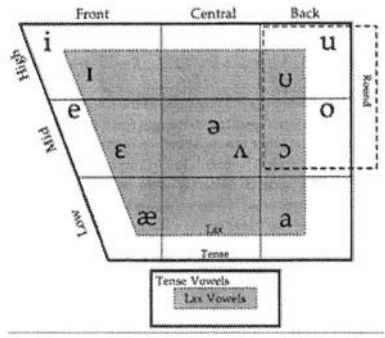


Fig. 6. IPA definition of Standard American English - Vowels

1. $(Score_1 - Score_2) - (Score_2 - Score_3)$, where $(Score_1$ is the highest score, the peak), $score_2$ is the next highest score etc. after normalization.
2. $(Score_1 - Score_2)$ / <phonetic difference between 1^{st} and 2^{nd} choice>

In order to calculate phonetic distance we find arrays of phonemes for both phrases, and then find a mapping of one array to the other, which gives minimal score. The overall score of a mapping is equal to the following:

Σ distances between all pairs of phonemes mapped to each other + 0.5 * all unmapped vowels + all unmapped consonants

The distance between two phonemes is calculated as follows:

- If one of them is vowel, and the other is consonant, then the distance is 1.
- If both same, then the distance is derived from

0.5 * sqrt (sum of squared differences of features)

The vowel features include the 'Front-Back' and 'Low-High' (from Fig. 6). The consonant features include the Place of Articulation (POA) and the Manner of Articulation (MOA) (from Fig. 5). Each feature is mapped to a number from [0, 1] interval. For the "front-back" feature, the 'Front' is mapped to 0, and the 'Back' to 1. For the "low-high" feature, the 'Low' is mapped to 0, and the 'High' to 1. For the case of 'POA', the feature 'Voiceless biliabial' is mapped to 0 and the feature 'Voiced glottal' is mapped to 1. Finally, for 'MOA', the feature 'Stop' is mapped to 0, and 'Glide' - to 1.

The scoring and mapping process described here is totally novel and have not been used in other work that we have come across in our survey of related methods.

5 Experimental Results

We carried out tests with 400 (4 people x 2 modes x 50 sample each) commands. There were three distinct objectives for the tests:

a. Can we improve the overall recognition rate by using the secondary SVM classifier - using the performance of the IBM engine as the baseline?
b. Can we distinguish cases where the two classifiers are better off on their own?
c. Can we identify groups based on their demography on which the classifiers perform better?

Table 1. Overall recognition rates

	Tester 1	Tester 2	Tester 3	Tester 4	Average
ViaVoice™	56%	83%	88%	49%	69%
SVM+ViaVoice™	79%	77%	70%	80%	76%

Table 1 shows the overall recognition rates for the combination versus the ViaVoice™ working on its own. It is very clear that the combination is able to perform at a higher rate. On the whole, this pair-wise testing shows the following main themes:

i. On the whole, the combination outperforms the IBM ViaVoice™ working on its own.
ii. The combination is consistently better than IBM ViaVoice™ in confirming the top choice of the rank table.

When the combination says something is correct, it is more frequently correct.

Table 2. Definitions of false positives and negatives

	+	-
+	---	False -ve
-	False +ve	---

But this is not the complete picture. In practical commercial systems, there are two other aspects of performance that need to be addressed. The first is the analysis of false +ve and false -ve. Generally, users judge systems on the following two criteria:

i. How many times a command is not accepted?
ii. How many times a command is confused with another command?

Table 2 defines how false positives and false negatives are counted. Table 3 summarizes the performance of the combination and the IBM ViaVoice™ engine focusing on false positives. Table 4 summarizes the performance of the combination and the IBM ViaVoice™ engine focusing on false negatives.

Table 3. False positive result analysis

SVM/IBM	Tester 1	Tester 2	Tester 3	Tester 4
Tester 1	----	34/52	4/8	48/17
Tester 2	16/50	----	0/8	28/17
Tester 3	8/50	8/52	----	10/17
Tester 4	10/50	12/52	0/8	----

Table 4. False negative result analysis

SVM/IBM	Tester 1	Tester 2	Tester 3	Tester 4
Tester 1	----	4/0	80/0	13/0
Tester 2	16/2	----	86/0	28/17
Tester 3	34/2	24/0	----	15/0
Tester 4	32/2	16/0	4/0	----

Based on these comparisons, the following observations can be made.

i. The combination has much lower rate of false +ves than IBM ViaVoice™ on its own.
   - IBM ViaVoice™ confuses too many commands.
   - IBM ViaVoice™ rejects too many commands even though they are correctly placed at the top of the ranking table.

ii. However IBM ViaVoice™ has performed better than the combination with respect to false negatives.
   When ViaVoice™ says something is incorrect, frequently it is incorrect.

The discussion so far shows that in a practical application, the straightforward recognition performance may not be the most appropriate yardstick. If reducing the false +ves is deemed more important from the user point of view, then the combined system is obviously the preferred option. However, if reducing false negatives is considered to be the priority, then the combination is not the obvious choice any more, although the combination does offer better overall recognition performance.

In addition, it was also found that the combination performs better than the COTS system when the speakers are non-native English speakers. This might be due to the fact that an SVM is a global expected risk minimizer, and not a local optmizer based on training samples [10]. An SVM's effectiveness depends on the number of Support Vectors (SVs) within a close proximity to the separating hyperplane. Local training may provide too many SVs. This might be an important aspect when choosing to use the combination based on the target user demographic in commercial marketing.

The final point to address is the cost. Deployment of any commercial system must address the effect of choosing a computation-heavy solution in place of the default solution, which in this case is the COTS IBM ViaVoice™ engine. There are two types of costs, the training cost and the execution cost. Since the cost of training is directly connected with the length of the ranking list (in Fig. 2, this length is 9), we chose three different scenarios for this comparison. Our experiments show that the cost of training goes up by 8.9% if the length is changed from 1 to 3. If the length is 9, meaning that 9 options are used for retraining, then the training cost is increased by 35.2%. In terms of execution cost, it has been found from our experiments that the combined system loses throughput by 14.2% when a 3-length ranked list is used. If the rank list has 9 entries, then the throughput decreases by 112.3%. In our experience, reducing the length of the ranked list from 9 to 3 does not produce any change in recognition performance. So for all practical purposes, the combined system will be 14.2% slower compared to the case when IBM ViaVoice™ is working on its own. This slowdown is not considered to be significant in terms of the user experience.

6 Conclusions

This paper has presented a novel method of calculating secondary ranking using confidence scores from a commercial voice recognition engine using SVM classifiers. The method is generic and since it does not depend on the way the engine works, is applicable to a wide variety of commercial voice recognition engines. The experimentation shows that adoption of SVM as secondary post-processing stage significantly increases the overall performance of the chosen voice recognition engine.

References

1. A. Rahman and M. C. Fairhurst. Multiple Classifier Decision Combination Strategies for Character Recognition: A Review. Special Issue on Multiple classifiers for document analysis applications. Int. Jour. on Document Analysis and Recognition (IJDAR), in press.
2. A. F. R. Rahman and M. C. Fairhurst, "Serial combination of multiple experts: A unified evaluation". Int. Journal of Pattern Analysis and Applications, 2:292-311, 1999.
3. T. Joachims, "Transductive Inference for Text Classification using Support Vector Machines", International Conference on Machine Learning (ICML), 1999.
4. K. Morik, P. Brockhausen, and T. Joachims, "Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring", Proc. 16th Int'l Conf. on Machine Learning (ICML-99), 1999.
5. Y.-J. Lee, O. L. Mangasarian and W. H. Wolberg, "Survival-Time Classification of Breast Cancer Patients", Data Mining Institute Technical Report 01-03, 2001.
6. H. Alam, Y. Tarnikova and F. Rahman, "Exploring a Hybrid of Support Vector Machines (SVMs) and a Heuristic Based System in Classifying Web Pages", Document Recognition and Retrieval X, 15th Annual IS&S/SPIE Symposium, 2003.
7. A. Smola and B. Schölkopf, http://www.kernel-machines.org/publications.html.
8. B. Scholkopf, S. Dumais, E. Osuna and J. Platt, "Support Vector Machine", in IEEE Intelligent Systems Magazine, Trends and Controversies, Marti Hearst, ed., 13(4), pp. 18-28, July/August 1998.
9. V. Vapnik, "The Nature of Statistical Learning Theory", Springer, 1995.
10. T. Joachims, in "Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning", B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

This page intentionally left blank

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–2984

please contact your bookseller or Springer-Verlag

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages, 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3066: S. Tsumoto, R. S lowiński, J. Komorowski, J.W. Grzymala-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.). Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.),Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3056: H. Dai.R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.

Vol. 3053: C. Bussler, J. Davies, D. Fensel, R. Studer (Eds.), The Semantic Web: Research and Applications. XIII, 490 pages. 2004.

Vol. 3052: W.Zimmermann, B. Thalheim (Eds.), Abstract State Machines 2004. Advances in Theory and Practice. XII, 235 pages. 2004.

Vol. 3051: R. Berghammer, B. Möller, G. Struth (Eds.), Relational and Kleene-Algebraic Methods in Computer Science. X, 279 pages. 2004.

Vol. 3050: J. Domingo-Ferrer, V. Torra (Eds.), Privacy in Statistical Databases. IX, 367 pages. 2004.

Vol. 3047: F. Oquendo, B. Warboys, R. Morrison (Eds.), Software Architecture. X, 279 pages. 2004.

Vol. 3046: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1016 pages. 2004.

Vol. 3045: A. Laganà M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1040 pages. 2004.

Vol. 3044: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1140 pages. 2004.

Vol. 3043: A. Laganà, M.L. Gavrilova, V. Kumar, Y. Mun, C.K. Tan, O. Gervasi (Eds.), Computational Science and Its Applications - ICCSA 2004. LIII, 1180 pages. 2004.

Vol. 3042: N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos (Eds.), NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications. XXXIII, 1519 pages. 2004.

Vol. 3039: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1271 pages. 2004.

Vol. 3038: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 1311 pages. 2004.

Vol. 3037: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 745 pages. 2004.

Vol. 3036: M. Bubak, G.D.v. Albada, P.M. Sloot, J.J. Dongarra (Eds.), Computational Science - ICCS 2004. LXVI, 713 pages. 2004.

Vol, 3035: M.A. Wimmer(Ed.), Knowledge Management in Electronic Government. XII, 326 pages. 2004. (Subseries LNAI).

Vol. 3034: J. Favela, E. Menasalvas, E. Chávez (Eds.), Advances in Web Intelligence. XIII, 227 pages. 2004. (Subseries LNAI).

Vol. 3033: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVIII, 1076 pages. 2004.

Vol. 3032: M. Li, X.-H. Sun, Q. Deng, J. Ni (Eds.), Grid and Cooperative Computing. XXXVII, 1112 pages. 2004.

Vol. 3031: A. Butz, A. Krüger, P. Olivier (Eds.), Smart Graphics. X, 165 pages. 2004.

Vol. 3030: P. Giorgini, B. Henderson-Sellers, M. Winikoff (Eds.), Agent-Oriented Information Systems. XIV, 207 pages. 2004. (Subseries LNAI).

Vol. 3029: B. Orchard, C. Yang, M. Ali (Eds.), Innovations in Applied Artificial Intelligence. XXI, 1272 pages. 2004. (Subseries LNAI).

Vol. 3028: D. Neuenschwander, Probabilistic and Statistical Methods in Cryptology. X, 158 pages. 2004.

Vol. 3027: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004. XI, 628 pages. 2004.

Vol. 3026: C. Ramamoorthy, R. Lee, K.W. Lee (Eds.), Software Engineering Research and Applications. XV, 377 pages. 2004.

Vol. 3025: G.A. Vouros, T. Panayiotopoulos (Eds.), Methods and Applications of Artificial Intelligence. XV, 546 pages. 2004. (Subseries LNAI).

Vol. 3024: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3023: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 611 pages. 2004.

Vol. 3022: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 621 pages. 2004.

Vol. 3021: T. Pajdla, J. Matas (Eds.), Computer Vision - ECCV 2004. XXVIII, 633 pages. 2004.

Vol. 3019: R. Wyrzykowski, J.J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics. XIX, 1174 pages. 2004.

Vol. 3016: C. Lengauer, D. Batory, C. Consel, M. Odersky (Eds.), Domain-Specific Program Generation. XII, 325 pages. 2004.

Vol. 3015: C. Barakat, I. Pratt (Eds.), Passive and Active Network Measurement. XI, 300 pages. 2004.

Vol. 3014: F. van der Linden (Ed.), Software Product-Family Engineering. IX, 486 pages. 2004.

Vol. 3012: K. Kurumatani, S.-H. Chen, A. Ohuchi (Eds.), Multi-Agnets for Mass User Support. X, 217 pages. 2004. (Subseries LNAI).

Vol. 3011: J.-C. Régin, M. Rueher (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 415 pages. 2004.

Vol. 3010: K.R. Apt, F. Fages, F. Rossi, P. Szeredi, J. Váncza (Eds.), Recent Advances in Constraints. VIII, 285 pages. 2004, (Subseries LNAI).

Vol. 3009: F. Bomarius, H. Iida (Eds.), Product Focused Software Process Improvement. XIV, 584 pages. 2004.

Vol. 3008: S. Heuel, Uncertain Projective Geometry. XVII, 205 pages. 2004.

Vol. 3007: J.X. Yu, X, Lin, H. Lu, Y. Zhang (Eds.), Advanced Web Technologies and Applications. XXII, 936 pages. 2004.

Vol. 3006: M. Matsui, R. Zuccherato (Eds.), Selected Areas in Cryptography. XI, 361 pages. 2004.

Vol. 3005: G.R. Raidl, S. Cagnoni, J. Branke, D.W. Corne, R. Drechsler, Y. Jin, C.G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G.D. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing. XVII, 562 pages. 2004.

Vol. 3004: J. Gottlieb, G.R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization. X, 241 pages. 2004.

Vol. 3003: M. Keijzer, U.-M. O'Reilly, S.M. Lucas, E. Costa, T. Soule (Eds.), Genetic Programming. XI, 410 pages. 2004.

Vol. 3002: D.L. Hicks (Ed.), Metainformatics. X, 213 pages. 2004.

Vol. 3001: A. Ferscha, F. Mattern (Eds.), Pervasive Computing. XVII, 358 pages. 2004.

Vol. 2999: E.A. Boiten, J. Derrick, G. Smith (Eds.), Integrated Formal Methods. XI, 541 pages. 2004.

Vol. 2998: Y. Kameyama, P.J. Stuckey (Eds.), Functional and Logic Programming. X, 307 pages. 2004.

Vol. 2997: S. McDonald, J. Tait (Eds.), Advances in Information Retrieval. XIII, 427 pages. 2004.

Vol. 2996: V. Diekert, M. Habib (Eds.), STACS 2004. XVI, 658 pages. 2004.

Vol. 2995: C. Jensen, S. Poslad, T. Dimitrakos (Eds.), Trust Management. XIII, 377 pages. 2004.

Vol. 2994: E. Rahm (Ed.), Data Integration in the Life Sciences. X, 221 pages. 2004. (Subseries LNBI).

Vol. 2993: R. Alur, G.J. Pappas (Eds.), Hybrid Systems: Computation and Control. XII, 674 pages. 2004.

Vol. 2992: E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, E. Ferrari (Eds.), Advances in Database Technology - EDBT 2004. XVIII, 877 pages. 2004.

Vol. 2991: R. Alt, A. Frommer, R.B. Kearfott, W. Luther (Eds.), Numerical Software with Result Verification. X, 315 pages. 2004.

Vol. 2990: J. Leite, A. Omicini, L. Sterling, P. Torroni (Eds.), Declarative Agent Languages and Technologies. XII, 281 pages. 2004. (Subseries LNAI).

Vol. 2989: S. Graf, L. Mounier (Eds.), Model Checking Software. X, 309 pages. 2004.

Vol. 2988: K. Jensen, A. Podelski (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XIV, 608 pages. 2004.

Vol. 2987:I. Walukiewicz (Ed.), Foundations of Software Science and Computation Structures. XIII, 529 pages. 2004.

Vol. 2986: D. Schmidt (Ed.), Programming Languages and Systems. XII, 417 pages. 2004.

Vol. 2985: E. Duesterwald (Ed.), Compiler Construction. X, 313 pages. 2004.