Neamat El Gayar
Josef Kittler
Fabio Roli (Eds.)

LNCS 5997

Multiple Classifier Systems

9th International Workshop, MCS 2010
Cairo, Egypt, April 2010
Proceedings

Springer

Lecture Notes in Computer Science 5997

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

Neamat El Gayar Josef Kittler
Fabio Roli (Eds.)

Multiple Classifier Systems

9th International Workshop, MCS 2010
Cairo, Egypt, April 7-9, 2010
Proceedings

Springer

Volume Editors

Neamat El Gayar
Nile University
Center for Informatics Science
12677 Giza, Egypt
E-mail: nelgayar@nileuniversity.edu.eg

Josef Kittler
University of Surrey
Centre for Vision, Speech and Signal Processing
Guildford, Surrey GU2 7XH, UK
E-mail: ees1jk@ee.surrey.ac.uk

Fabio Roli
University of Cagliari
Department of Electrical and Electronic Engineering
Piazza d'Armi, 09123, Cagliari, Italy
E-mail: roli@diee.unica.it

Library of Congress Control Number: 2010922797

CR Subject Classification (1998): I.4, H.3, I.5, F.1, J.3, H.4

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

ISSN 0302-9743
ISBN-10 3-642-12126-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-12126-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

These proceedings are a record of the Multiple Classifier Systems Workshop, MCS 2010, held at the Nile University, Egypt in April 2010. Being the ninth in a well-established series of meetings providing an international forum for discussion of issues in multiple classifier system design, the workshop achieved its objective of bringing together researchers from diverse communities (neural networks, pattern recognition, machine learning and statistics) concerned with this research topic. From more than 50 submissions, the Program Committee selected 31 papers to create an interesting scientific program. Papers were organized into sessions dealing with classifier combination and classifier selection, diversity, bagging and boosting, combination of multiple kernels, and applications. The workshop program and this volume were enriched by two invited talks given by Gavin Brown (University of Manchester, UK), and Friedhelm Schwenker (University of Ulm, Germany).

As usual, the workshop would not have been possible without the help of many individuals and organizations. First of all, our thanks go to the members of the MCS 2010 Program Committee, whose expertise and dedication helped us create an interesting event that marks the progress made in this field over the last year and aspire to chart its future research. The help of James Field from the University of Surrey, who administered the submitted paper reviews, and of Giorgio Fumera who managed the MCS website deserve a particular mention. Special thanks are due to the members of the Nile University Organizing Committee, Ahmed Salah, Amira El Baroudy, Esraa Aly, Heba Ezzat, Nesrine Sameh, Rana Salah and Mohamed Zahhar for their indispensable contributions to the registration management, local organization, and proceedings preparation.

This workshop was supported by the Center for Informatics Science at the Nile University, Egypt, the Center for Vision, Speech and Signal Processing, University of Surrey, UK and the Department of Electrical and Electronic Engineering of the University of Cagliari, Italy. We also thank the International Association for Pattern Recognition for endorsing the MCS 2010. We wish to express our appreciation to our two financial sponsors: The Information Technology Industry Development Agency (ITIDA), which was the main sponsor of the event, and the Microsoft Innovation Laboratory in Cairo (CMIC).

April 2010

Neamat El Gayar
Josef Kittler
Fabio Roli

Organization

MCS 2010 was organized by the Center for Informatics Science (CIS) at the Nile University, Egypt in association with the Center for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK and the Department of Electrical and Electronic Engineering(Diee) of the University of Cagliari, Italy.

Program Committee

Conference Chairs	Neamat El Gayar (Nile University, Egypt) Josef Kittler (University of Surrey, UK) Fabio Roli (University of Cagliari, Italy)

Scientific Committee

J. Benediktsson (Iceland)
G. Brown (UK)
L. Bruzzone (Italy)
H. Bunke (Switzerland)
L.P. Cordella (Italy)
R.P.W. Duin (The Netherlands)
G. Fumera (Italy)
C. Furlanello (Italy)
J. Ghosh (USA)
V. Govindaraju (USA)
L.Hall (USA)
T.K. Ho (USA)
A. Jain (USA)
L.I. Kuncheva (UK)
V. Mottl (Russia)
N. Oza (USA)
R. Polikar (USA)
A. Ross (USA)
C. Sansone (Italy)
A. Sharkey (UK)
D. Tax (The Netherlands)
G.Valentini (Italy)
T. Windeatt (UK)
D. Windridge (UK)
Z.-H. Zhou (China)

Sponsoring Institutions

International Association of Pattern Recognition (IAPR)
The Information Technology Industry Development Agency (ITIDA)
Microsoft Innovation Laboratory in Cairo (CMIC)
Nile University
University of Surrey
University of Cagliari

Table of Contents

Classifier Ensembles(I)

Classifier Ensembles(II)

Classifier Diversity

Classifier Selection

Combining Multiple Kernels

Boosting and Bootstrapping

Handwriting Recognition

Applications

Invited Papers

Erratum

Weighted Bagging for Graph Based One-Class Classifiers

Santi Seguí[1,2], Laura Igual[1,2], and Jordi Vitrià[1,2]

[1] Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Spain
[2] Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, 08007, Barcelona, Spain

Abstract. Most conventional learning algorithms require both positive and negative training data for achieving accurate classification results. However, the problem of learning classifiers from only positive data arises in many applications where negative data are too costly, difficult to obtain, or not available at all. Minimum Spanning Tree Class Descriptor (MST_CD) was presented as a method that achieves better accuracies than other one-class classifiers in high dimensional data. However, the presence of outliers in the target class severely harms the performance of this classifier. In this paper we propose two bagging strategies for MST_CD that reduce the influence of outliers in training data. We show the improved performance on both real and artificially contaminated data.

1 Introduction

One-class classification tries to distinguish one class of objects (*target class*) from all other possible objects (*outlier class*). In contrast to traditional classifier methods, the one-class classifier methods only use examples of the target class in the learning process and any information of the outlier class is applied [1]. Its goal is to model the target class to accept as much number of target examples as possible while minimizing the number of outliers.

There are three main approaches for building one-class classifiers: density estimators (Gaussian model [2], mixture of Gaussians [3], Parzen density [4]), boundary methods (k-centers [5], NN-d [3] and OCSVM [6]) and reconstruction methods (k-mean clustering [2], self-organizing maps [7]). Density methods estimate the density of the training set and set a threshold on this density. These methods have excellent results when a good probability model is provided and the sample size is sufficient. Boundary methods estimate the boundary that encloses target class examples. These methods heavily rely on the distance between objects, so they tend to be very sensitive to the scale of features. Most of the reconstruction methods make assumptions about the clustering characteristics of the data or their distribution in subspaces. Then, a set of prototypes or subspaces are defined and a reconstruction error is minimized.

Recently, a graph-based one-class classifier method, Minimum Spanning Tree Class Descriptor (MST_CD), has been proposed in [8]. This one-class classifier computes the Minimum Spanning Tree (MST) on the training set, and the classification rule relies on the distance to the closest edge of this constructed

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 1–10, 2010.

tree. This method achieves better accuracy than others methods in problems with high dimensional spaces (small amount of samples in comparison with data dimensionality). MST_CD method constructs the entire MST with all data without ruling out any example of the training set in the graph definition. This may result in a high data sensitivity, affecting the performance when outliers are present in the target class. We propose to combine multiple MST_CD classifiers to achieve robustness to the presence of outliers to the original MST_CD.

Ensemble methods combine outputs of several classifiers using diversified data to improve the performance. The diversity of classifier outputs is a critical requirement to assure success of ensemble methods. In [9], Kuncheva et.al. note that the ensemble methods which focuss on creating diversity in an intuitive way seem to have very good results, however, those methods which measure diversity and use it explicitly in the process of creating the ensemble, apparently, do not have the same improvement.

There are three main ways to create diversity: the first one consists of considering a different pool of examples in each classifier of the ensemble; the second way consists of considering different set of features; and the last way consists of using different classifier methods to create each classifier in the ensemble.

Bagging [10] trains each classifier in the ensemble with a resampled version of the training set. This method is useful for unstable classifiers in which small changes in the training set cause large changes in the estimated boundary [11]. Weighted bagging [12] is an extension of original bagging which establishes a different weight for each example to be included in the bootstrap samples in order to create classifiers robust to outliers.

In this paper, we propose the use of two different ensemble methods for one-class classification based on MST_CD method. In particular, the proposed methods are bagging MST_CD and weighted bagging MST_CD which vary the pool of examples for building each classifier. We show that the combinations of MST_CD dramatically improve the results, even for small size data problems where the original MST_CD behaves well.

The rest of the paper is organized as follows: Section 2 reviews MST_CD method; Section 3 introduces the two proposed methodologies for combining multiple one-class classifiers: bagging MST_CD and weighted bagging MST_CD; Section 4 presents the experimental results; and Section 5 finalizes the paper with the conclusions and future work.

2 Minimum Spanning Tree Class Descriptor

MST_CD is a classifier which is based on a graph representation of the training data. This classifier was originally proposed in [8]. The graph is computed to capture the structure of the training data set. In particular, this structure is based on the Minimum Spanning Tree (MST) [13].

Training process of the classifier is formulated as solving the standard MST problem. Several algorithms have been proposed for finding the graph, being Prim's [14] and Kruskal's [15] the most popular. The task of these methods

consists of finding the graph without loops which connects all the vertices, such that the total length of the edges is minimum. The length of an edge that connects two target samples x_i and x_j is usually measured as the Euclidian distance between these points.

Basic elements of this classifier are not only vertices but also edges of the graph, giving a much richer representation of the data. Considering edges of the graph as target class objects, additional virtual target objects are generated. The classification of a new object $\mathbf{x}$ is based on the distance to the nearest vertex or edge. The projection of $\mathbf{x}$ onto a line defined by the vertices $\{x_i, x_j\}$ is:

$$p_{e_{ij}}(\mathbf{x}) = x_i + \frac{(x_j - x_i)^T(x - x_i)}{(\|x_j - x_i\|)}(x_j - x_i). \tag{1}$$

If the projection $p_{e_{ij}}$ lies on the edge, then the distance $d(\mathbf{x}\|e_{ij})$ between $\mathbf{x}$ and the edge e_{ij} is computed as the Euclidian distance. Otherwise, $d(\mathbf{x}\|e_{ij})$ is derived as the shortest Euclidian distance from one of the vertices $\{x_i, x_j\}$

The distance of the new object $\mathbf{x}$ to the target class is defined as the minimum distance to the set of $(n-1)$ edges of the MST:

$$d_{MST\_CD}(\mathbf{x}, X) = \min_{e_{ij} \in MST} d(\mathbf{x}\|e_{ij}). \tag{2}$$

The decision whether $\mathbf{x}$ belongs to the *target* or *outlier* class is based on a threshold, θ, set on the distances d_{MST_CD}. This threshold cannot be derived as an error on the training set since the distance of all target objects is equal to zero according to the definition of the distance. Therefore, θ is determined by a quantile function of the distribution of the edge weights in the given MST. The quantile function is defined as:

$$\theta \equiv \vartheta_\alpha(\tilde{\mathbf{e}}) = \|e_{([\alpha n])}\|, \tag{3}$$

where $\tilde{\mathbf{e}} = (e_{(1)}, e_{(2)}, ..., e_{(n)})$ is the sorted sequence of scalar weights of the edges in the MST, such that $\|e_{(1)}\| \leq \|e_{(2)}\| \leq, ..., \leq \|e_{(n)}\|$, $[a]$ returns the nearest integer of a and $\alpha \in [0, 1]$. Thus, $\vartheta_0(\tilde{\mathbf{e}})$ returns the minimum distance between two edges of the MST, $\vartheta_1(\tilde{\mathbf{e}})$ is maximum, and $\vartheta_{0.5}(\tilde{\mathbf{e}})$ is median weight of edges.

3 Ensemble of Minimum Spanning Tree Class Descriptors

The main idea of ensemble methods is to combine the output of several classifiers, each one trained with diversified data set. The goal of these classifiers consists of creating more robust, precise and accurate classification systems. It has been proved that ensemble classifiers improve the classification performance in many applications, compared to single classifier methods [16]. The use of this methodology for one-class classification problem is justified when the classifier method is sensitive to the data, which is the case of MST_CD, where the presence of outlier examples in the training set could be critic.

The proposed methods are bagging MST_CD (B_MST_CD) and weighted bagging MST_CD (WB_MST_CD).

3.1 Bagging Minimum Spanning Tree Class Descriptors

Bagging was introduced by Breiman in [10], and since then is one of the most popular ensemble methods. This method trains each classifier in the ensemble with a resampled version of the training set.

Let $X = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ be a N x n matrix corresponding to the training data set, where N is the number of examples, and $\mathbf{x}_i = [x_1, ..., x_n]^T$ are data examples described by n features. Bagging generates L new training data sets X'_i of size N' ($N' \leq N$) by sampling from X uniformly and with replacement. Some instance of X may not appear in some X'_i, and some may appear duplicated in X'_i. In [11], it was demonstrated that bagging is useful for unstable classifiers in which small changes in the training set cause large changes in the estimated boundary. MST_CD method is very sensitive to changes in the training data examples, since we use the entire MST (See Figure 1 (a)). Therefore, bagging can be applied to MST_CD in order to increase robustness of this one-class classifier method (See Figure 1 (b)). Nevertheless, although bagging has a positive influence in the classifier the outlier contribution persists in the final classifier.

3.2 Weighted Bagging Minimum Spanning Tree Class Descriptors

In classical bagging, all examples have the same probability to appear in one of the classifiers of the ensemble. Each bootstrap sample contains the same number of examples as the training set, but with repeated examples. If all examples have the same probability, outlier samples are likely to be included in the most part of the bootstrap samples. Weighted bagging was introduced in [12] to overcome this problem. The authors proposed to give probability weights to points based on how close are them to the target class by using a kernel density estimator. The proposed method implicitly imposes that outlier points (points far from the target class) should have lower probability weights.

To estimate the probability weights they used the iterative method proposed in [17]. This method, initially, gives the same probability weight to each sample. At iteration k, the probability weight is estimated by:

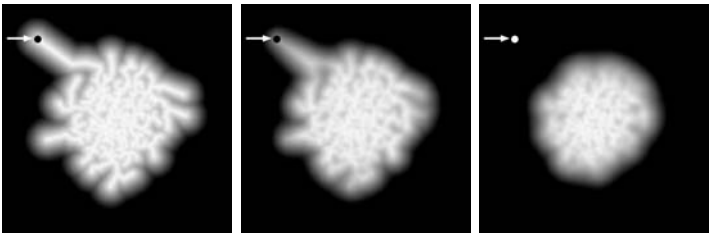
$$w_k(i) = w_{k-i}(i) + log(\frac{f_{k-1}(x_i)}{f'_{k-i}(x_i)}), \tag{4}$$

where f_k is the weighted kernel density estimator defined as:

$$f_k(\mathbf{x}_i) = \sum_{j=1}^{m} \frac{w_k(i)}{(2\pi)^{d/2}\sigma^d} k(x_i, x_j). \tag{5}$$

And f'_k is defined as:

$$f'_k(\mathbf{x}_i) = \sum_{j=1}^{m} \frac{w_k(i)}{(2\pi)^{d/2}\sigma^d} k(x_i, x_j) I(j \neq i), \tag{6}$$



(a) MST_CD (b) B_MST_CD (c) WB_MST_CD

Fig. 1. Results of MST_CD (a), B_MST_CD (b) and WB_MST_CD (c) methods on synthetic Gaussian data and one outlier (indicated by the arrow)

where I is the pulse function and $k(x_i, x_j)$ is a Gaussian kernel function defined as:

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2}). \tag{7}$$

Finally, the probability weights are inverted and normalized as follows:

$$w(i) = (\frac{1}{w_n(i)})/\Big(\sum_{j=1}^{m} \frac{1}{w_n(j)}\Big). \tag{8}$$

The final probability will be low for samples in low density regions and high for samples in high density regions.

We consider a toy problem example to illustrate the difference between the original MST_CD and the two proposed ensemble methods: B_MST_CD and WB_MST_CD. Synthetic data set was generated using a Gaussian function and one outlier was included in the target class. Figure 1 shows the qualitative performance of the three methods. In particular, distances computed in the learning process (Eq. (2)) are depicted using gray level: white means distance 0 and black means distance larger than 1. The boundary was defined using threshold θ equal to 1. In the figures, it can be appreciated that MST_CD and B_MST_CD are not able to reject the included outlier, whereas, the boundary defined by WB_MST_CD does not include the outlier.

4 Experiments

In order to evaluate the performance of the one-class ensemble classifier methods, the area under the receiver operator characteristics (ROC) curve (AUC) is computed in the experiments [18]. The AUC measure is the total performance of a one-class classifier integrated over all possible thresholds. As thresholds we refer the distance to the graph (Eq. (2)). A large AUC value means better performance of the one-class classifier, a value lower than 50% means that the classifier is worst than random guessing. Additionally, we use true positive rate (TPR)

Table 1. Data sets used in the experiments

Data Set	Features	#Target class	#Outliers class
Heart	9	239	444
Iris	13	150	120
Ecoli	7	52	284
Ionosphere	34	126	225
Spectf	44	95	254
Sonar	60	111	97
Arrythmia	278	237	183
Colon	2000	22	44

and false positive rate (FPR) as a performance metric. Varying the threshold θ of the graph we can get an specific TPR for a fixed FPR. As was done in [12], different target of FPRs are fixed; these are, typically, 1%, 5% and 10%.

All the experiments were evaluated using 50% of target class as training set, and the other 50% of target class together with the outlier class as test set. The experiments were repeated 20 times and the mean value of the results is presented. Both, in B_MST_CD and in WB_MST_CD, L was fixed to 50. Moreover, the σ parameter of the kernel density function of weighed bagging was fixed previously by cross-validation for each database. The complexity parameter, γ, in MST_CD was fixed to the maximum, and the threshold θ was fixed to 0.1. The MATLAB version of MST_CD available in *DD_TOOL* toolbox [19] [1] for prtools library was used.

Eight databases are used, all of them obtained from UCI repository [20], except for Colon data which is an example of gene expression data. Data sets with more examples than features are referred as *low-dimensional* data and data sets with more feature than examples are referred as *high dimensional* data.

Table 1 contains the list of data sets with the corresponding number of features and number of examples of the *target* and *outlier* class. We sorted them from data set with highest (first row) until lowest (last row) relation between examples and features. The *low dimensional* databases are: Heart, Iris, Ecoli and Ionosphere, whereas the *high dimensional* databases are: Spectf, Sonar, Arrythmia and Colon. Two experiments were performed, one using the original data sets (*clean data set*) and another using a noisy version of the same data sets (*noisy data set*).

Clean Data Set. Table 2 presents the results obtained on the 8 data sets. The AUC value for each method is presented and the rank of the methods are printed in parenthesis. In each row, the best and not significantly worse than the best result are marked bold. To measure the significance of the obtained results the statistical Tukey's test is used [21] with a confidence level fixed to 95%.

We observe that WB_MST_CD achieved the best result on 5 of the data sets, and, on the other 3, it obtained a non statistically worse result than the best

[1] http://ict.ewi.tudelft.nl/~davidt/dd_tools.html

Table 2. Performance using original data sets. AUC measure and rank is presented for every databases. Best and not significantly worse results than the best are marked bold. Performance variation (gain) between the ensemble methods (B_MST_CD and WB_MST_CD)and original MST_CD is presented.

Data Set	MST_CD	B_MST_CD	Gain	WB_MST_CD	Gain
Heart	79.80% (3.0)	81.30% (2.0)	+3.7%	**82.70%** (1.0)	+6.5%
Iris	97.90% (3.0)	98.70% (2.0)	+6.5%	**99.00%** (1.0)	+9.0%
Ecoli	89.30% (3.0)	92.50% (2.0)	+5.8%	**93.30%** (1.0)	+7.4%
Ionosphere	96.90% (3.0)	97.40% (2.0)	+5.9%	**97.80%** (1.0)	+8.5%
Spectf	**90.10%** (2.5)	**90.20%** (1.0)	+0.4%	**90.10%** (2.5)	+1.3%
Sonar	**70.10%** (1.0)	68.20% (3.0)	-0.9%	**69.00%** (2.0)	-0.5%
Arrythmia	**79.30%** (3.0)	**79.60%** (1.0)	+0.7%	**79.50%** (2.0)	+2.3%
Colon	**70.30%** (3.0)	71.90% (2.0)	+1.6%	**72.00%** (1.0)	+1.7%
Rank	**(2.4)**	**(1.8)**		**(1.4)**	

method. In particular, for the 4 *low dimensional* data sets, the best results were obtained by WB_MST_CD. These results were significantly better than the results obtained by the other two methods. However, on the 4 *high dimensional* data sets no clear conclusions can be extracted. MST_CD obtained the best result on Sonar data set, and in the other 3, this method obtained a result not significantly worse than the best.

Table 3 shows the results of TPR obtained for fixed values of FPRs. We can observe that the results are equivalent to the ones obtained in Table 2. For the most part of the data sets the method which gets the highest AUC value also obtains the highest TPR for all the analyzed $FPRs$.

Noisy Data Set. The second experiment is performed in a noisy version of the data sets generated by swapping 10% of the examples in the target class used as training by examples of the outlier class.

The obtained results are presented in Table 4. We observe that WB_MST_CD obtained the best rank, achieving better results than MST_CD and B_MST_CD on 7 of the used data sets, and on the other one the obtained result by WB_MST_CD is not significantly worse than the best result. In particular, if we analyze the *low dimensional*, WB_MST_CD scores much better than the other methods for all data sets. The average increase of the AUC value is higher than 6 points.

In the case of the *high dimensional* data sets, WB_MST_CD method achieves the best result on 3 of the 4 data sets. The gain of WB_MST_CD with respect to MST_CD is substantially lower than in the case of *low dimensional* data.

Table 5 shows the performance variation of using the noisy data instead of clean data. We observe that WB_MST_CD suffers a lower reduction of the AUC value than MST_CD: the mean values are 7.0%, 4.8% and 3.7% by MST_CD, ensemble B_MST_CD and ensemble WB_MST_CD respectively. For example, in the case of Iris and Ionosphere data sets, the decrease value of the AUC by MST_CD is 9.1% and 10.2% while by ensemble WB_MST_CD is only 1.2% and 2.6% respectively.

Table 3. Performance of MST_CD, B_MST_CD and WB_MST_CD. TPR for some fixed values of $FPRs$ are presented. Best results are marked bold

Data Set	FPR	TPR		
		MST_CD	B_MST_CD	WB_MST_CD
Heart	1%	19.67 %	23.03 %	**23.68** %
	5%	36.18 %	41.58 %	**41.91** %
	10%	46.58 %	49.80 %	**52.83** %
Iris	1%	64.81 %	75.19 %	**84.04** %
	5%	92.12 %	93.08 %	**93.65** %
	10%	96.73 %	97.12 %	**98.85** %
Ecoli	1%	13.33 %	41.48 %	**51.11** %
	5%	50.37 %	74.07 %	**80.56** %
	10%	68.33 %	82.59 %	**85.74** %
Ionosphere	1%	64.82 %	68.86 %	**73.82** %
	5%	75.31 %	78.99 %	**84.52** %
	10%	88.64 %	89.87 %	**92.02** %
Spectf	1%	**48.27** %	47.14 %	47.76 %
	5%	56.63 %	**57.04** %	**57.04** %
	10%	64.39 %	**66.43** %	65.10 %
Sonar	1%	**24.21** %	17.89 %	22.81 %
	5%	**36.84** %	31.93 %	36.05 %
	10%	**50.35** %	48.95 %	49.82 %
Arrythmia	1%	12.08 %	**12.83** %	12.63 %
	5%	27.33 %	**27.75** %	27.54 %
	10%	40.29 %	**41.00** %	40.87 %
Colon	1%	27.08 %	26.25 %	**27.92** %
	5%	43.75 %	**45.00** %	**45.00** %
	10%	54.58 %	**56.25** %	**56.25** %

Table 4. Performance using the noisy data sets in the training step. AUC measure is presented for all databases. Best and not significantly worse results than the best are marked bold. Performance variation (gain) between the ensemble methods and original MST_CD is presented.

Data Set	MST_CD	B_MST_CD	Gain	WB_MST_CD	Gain
Heart	69.90% (3.0)	73.60% (2.0)	+3.7%	**76.40**% (1.0)	+6.5%
Iris	88.80% (3.0)	95.30% (2.0)	+6.5%	**97.80**% (1.0)	+9.0%
Ecoli	82.90% (3.0)	88.70% (2.0)	+5.8%	**90.30**% (1.0)	+7.4%
Ionosphere	86.70% (3.0)	92.60% (2.0)	+5.9%	**95.20**% (1.0)	+8.5%
Spectf	87.10% (3.0)	87.50% (2.0)	+0.4%	**88.40**% (1.0)	+1.3%
Sonar	**61.60**% (1.0)	60.70% (3.0)	-0.9%	**61.10**% (2.0)	-0.5%
Arrythmia	73.00% (3.0)	73.70% (2.0)	+0.7%	**75.30**% (1.0)	+2.3%
Colon	**67.70**% (3.0)	69.30% (2.0)	+1.6%	**69.40**% (1.0)	+1.7%
Rank	**(2.7)**	**(2.1)**		**(1.1)**	

Table 5. Performance variation with the presence of 10% of outliers on the target data

Data Set	MST_CD	B_MST_CD	WB_MST_CD
Heart	-9.9%	-7.7%	-6.3%
Iris	-9.1%	-3.4%	-1.2%
Ecoli	-6.4%	-3.8%	-3.0%
Ionosphere	-10.2%	-4.8%	-2.6%
Spectf	-3.0%	-2.7%	-1.4%
Sonar	-8.5%	-7.5%	-7.9%
Arrythmia	-6.3%	-5.9%	-4.2%
Colon	-2.6%	-2.6%	-2.6%
Mean	**-7.0%**	**-4.8%**	**-3.7%**

5 Conclusions

In this paper, we proposed two ensemble methods for one-class classification: B_MST_CD and WB_MST_CD. Method B_MST_CD slightly improves the performance of the original MST_CD method, however the statistical tests are not significant. Method WB_MST_CD, by estimating the probability weights of the training examples with a kernel density function, achieves much better results. Moreover, in this case, the obtained results are significant from a statistical point of view.

An experimental study was performed using 8 data sets, 4 of them *low dimensional* and 4 *high dimensional*. Both ensemble methods achieve better accuracies in *low dimensional* data sets and similar results in *high dimensional* data sets than original MST_CD. Experiments with noisy versions of the data were also performed and statistical test confirms the superiority of WB_MST_CD for dealing with outliers. WB_MST_CD results are better in *low dimensional* data than in *high dimensional* data sets. This could be due to the difficulty of estimating the probability density function in *high dimensional* data.

Future work will be study alternative ways, preferably non-parametric strategies, to estimate the weights for WB_MST_CD. Moreover, we will analyze other graph-representations for building the classifier instead of using MST. Additionally, Random Projections or other graph-based transformations will be considered to perform the classification. Finally, the sparseness nature of the chosen graph-based representation makes it feasible to represent large-scale data sets of high dimensional data. To this aim we should explore the use of approximate algorithms for building the different components of this classifier.

Acknowledgment

This work has been partially supported by MICINN Grants TIN2009-14404-C02 and CONSOLIDER-INGENIO 2010 (CSD2007-00018).

References

1. Tax, D.M.J.: One-class classification. PhD thesis, Delft. University of Technology (2001)
2. Bishop, C.M.: Neural Networks for Pattern Recognition, 1st edn. Oxford University Press, USA (1996)
3. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons Inc., Chichester (1973)
4. Parzen, E., Hoti, F.: On estimation of a probability density function and mode (1962)
5. Ypma, A., Ypma, E., Duin, R.P.: Support objects for domain approximation. In: ICANN 1998, Skovde, Sweden, pp. 2–4. Springer, Heidelberg (1998)
6. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13, 2001 (1999)
7. Kohonen, T., Schroeder, M.R., Huang, T.S. (eds.): Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus (2001)
8. Juszczak, P., Tax, D.M.J., Pekalska, E., Duin, R.P.W.: Minimum spanning tree based one-class classifier. Neurocomput. 72(7-9), 1859–1869 (2009)
9. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
10. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
11. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants (1998)
12. Shieh, A.D., Kamm, D.F.: Ensembles of one class support vector machines. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 181–190. Springer, Heidelberg (2009)
13. Graham, R.L., Hell, P.: On the history of the minimum spanning tree problem. IEEE Ann. Hist. Comput. 7(1), 43–57 (1985)
14. Prim, R.C.: Shortest connection networks and some generalizations. Bell System Technology Journal 36, 1389–1401 (1957)
15. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society 7(1), 48–50 (1956)
16. Dietterich, T.G.: Ensemble methods in machine learning, pp. 1–15. Springer, Heidelberg (2000)
17. Marzio, M.D.: Boosting kernel density estimates: A bias reduction technique? Biometrika 91(1), 226–233 (2004)
18. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
19. Tax, D.: Ddtools, the data description toolbox for matlab (December 2009), version 1.7.3
20. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
21. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)

Improving Multilabel Classification Performance by Using Ensemble of Multi-label Classifiers

Muhammad Atif Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan

Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford, GU2 7XH, UK
{m.tahir,j.kittler,k.mikolajczyk,f.yan}@surrey.ac.uk

Abstract. Multilabel classification is a challenging research problem in which each instance is assigned to a subset of labels. Recently, a considerable amount of research has been concerned with the development of "good" multi-label learning methods. Despite the extensive research effort, many scientific challenges posed by e.g. highly imbalanced training sets and correlation among labels remain to be addressed. The aim of this paper is use heterogeneous ensemble of multi-label learners to simultaneously tackle both imbalance and correlation problems. This is different from the existing work in the sense that the later mainly focuses on ensemble techniques within a multi-label learner while we are proposing in this paper to combine these state-of-the-art multi-label methods by ensemble techniques. The proposed ensemble approach (EML) is applied to three publicly available multi-label data sets using several evaluation criteria. We validate the advocated approach experimentally and demonstrate that it yields significant performance gains when compared with state-of-the art multi-label methods.

1 Introduction

A traditional multi-class classification system assigns each instance x a single label l from a set of disjoint labels L. However, in many modern applications such as music categorisation [1], text classification [2,3], image/video categorisation [4,5] etc, each instance is to be assigned to a subset of labels $Y \subseteq L$. This problem is known as multi-label learning.

There is considerable amount of research concerned with the development of "good" multi-label learning methods. Despite the extensive research effort, there exist many scientific challenges. They include highly imbalanced training sets, as very limited data is available for some labels, and capturing correlation among classes. Interestingly, most state-of-the-art multi-label methods are designed to focus mainly on the second problem and very limited effort has been devoted to handling imbalanced data populations. In this paper, we focus on the first problem of multi-label learning, and tackle highly imbalanced data distributions using ensemble of multi-label classifiers.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 11–21, 2010.

Ensemble techniques are becoming increasingly important as they have repeatedly demonstrated the ability to improve upon the accuracy of a single-classifiers [6]. Ensembles can be homogeneous, in which every base classifier is constructed using the same algorithm, or heterogeneous in which base classifiers are constructed using different algorithms. In fact, some state-of-the-art multi-label learners use homogeneous or heterogeneous ensemble techniques to improve the overall performance. For example, in [7] Logistic Regression and Nearest Neighbour classifiers are combined, in [8] random subsets of training data are used. The aim of this paper is to use heterogeneous ensemble of multi-label learners to improve the performance. This is different from the existing work in the sense that the latter mainly focuses on ensemble techniques within a multi-label learner while we are proposing to combine these state-of-the-art multi-label methods by ensemble techniques.

Interestingly another advantage of combining multi-label classifiers is that both imbalance and correlation problems can be tackled simultaneously. The imbalance problem can be handled by using ensemble of multi-label classifiers while the correlation problem can be solved by using state-of-the-art multi-label classifiers as base classifiers that inherently consider correlation among labels. The proposed ensemble approach (EML) is applied to three publicly available multi-label data sets from different domains (Scene, Yeast, and Enron) using 18 different multi-label classification measures. We validate the advocated approach experimentally and demonstrate that it yields significant performance gains when compared with individual state-of-the art multi-label methods.

The paper is organised as follows. In Section 2, we review state-of-the-art multi-label methods. Section 3 discusses the proposed ensemble of multi-label classifiers. Experiments are discussed in Section 4 followed by the results and discussion in Section 5. Section 6 concludes the paper.

2 Related Work

The sparse literature on multi-label classification driven by problems in text classification, bioinformatics, music categorisation, and image/video classification, has recently been summarised by Tsoumakas et al [9]. This research can be divided into two different groups: i) *problem transformation* methods, and ii) *algorithm adaptation* methods. The problem transformation methods aim to transform multilabel classification task into one or more single-label classification [10,11], or label ranking [12] tasks. The algorithm adaptation methods extend traditional classifiers to handle multi-label concepts directly [13,14,7]. In this section, we review the state-of-the-art multi-label learners that are used as base classifiers in our ensemble approach namely RaKEL [11], Calibrated Label Ranking (CLR) [12], Multi-label KNN (MLKNN) [13], Instance Based Logistic Regression (IBLR) [7] and Ensemble of Classifier Chains (ECC) [8].

RaKEL: Multilabel classification can be reduced to the conventional classification problem by considering each unique set of labels as one of the classes.

This approach is referred to as *label powerset* (LP) in the literature. However, this approach leads to a large number of label subsets with the majority of them with a very few examples and it is also computationally expensive. Many approaches have been proposed in the literature to deal with the aforementioned problem [11,15]. One state-of-the-art approach is RaKEL (Random k-Labelsets) [11] that constructs an ensemble of LP classifiers where each LP classifier is trained using a different small random subset of the set of labels. In order to get near-optimal performance, appropriate parameters (subset size, number of models, threshold etc) must be optimised using internal cross validation. However, these parameters are hard to optimised when the number of training samples is insufficient.

Ensemble of Classifier Chains (ECC): Multilabel classification can be reduced to the conventional binary classification problem. This approach is referred to as *binary relevance* (BR) learning in the literature. In BR learning, the original data set is divided into $|Y|$ data sets where $Y = \{1, 2, ..., N\}$ is the finite set of labels. BR learns one binary classifier $h_a : X \rightarrow \{\neg a, a\}$ for each concept $a \in Y$. BR learning is theoretically simple and has a linear complexity with respect to the number of labels. Its assumption of label independence makes it attractive to situations where new examples may not be relevant to any known subset of labels or where label relationships may change over the test data [8]. However, BR learning is criticised for not considering correlations among the labels [7,12]. The work in [8] is a state-of-the-art BR approach. Their classifier chain (CC) approach only requires a single training iteration like BR and uses labels directly from the training data without any internal classification. Classifiers are linked in a chain where each classifier deals with the BR problem associated with label $y_j \in Y$. However, the order of the chain can clearly have an effect on accuracy. An ensemble framework (ECC) is used to create different random chain orderings. This method was shown to perform well against BR and other multi-label classifiers.

Calibrated Label Ranking (CLR): Like *one-vs-all* approach in BR learning, the binary pairwise *one-vs-one* approach has also been employed for multi-label classification, therefore requiring $|Y|^2$ classifiers as opposed to $|Y|$. Calibrated label ranking (CLR) [12] is an efficient pairwise approach for multilabel classification. The key idea in this approach is to introduce an artificial calibration label that, in each example, separates the relevant label from the irrelevant labels. This method was shown to perform well against other multi-label classifiers but mainly on ranking measures.

Multi-label KNN (MLKNN): Instance-based approach is also quite popular in multilabel classification. In [13], a lazy learning approach (MLKNN) is proposed. This method is derived from the popular k-Nearest Neighbour (kNN) algorithm. It consists of two main steps. In the first step, for each test instance, its k nearest neighbours in the training set are identified. Next, in the second step, the maximum a posteriori probability label set is identified for a test instance based on the statistical information gained from the label sets of these

neighbouring instances. This method was shown to perform well in some domains e.g. in predicting the functional classes of genes in the Yeast Saccharomyces cerevisiae [7].

Instance Based Logistic Regression (IBLR): IBLR is also a novel approach to instance-based learning with main idea to combine instance-based learning (ILR) and logistic regression [7]. The key idea is to consider the labels of neighboring instances as "features" of unseen samples and thus reduce ILR to logistic regression. This approach captures interdependencies between labels for multilabel classification.

3 Ensemble of Multi-label Classifiers (EML)

Let X denote a set of images (instances) and let $Y = \{1, 2, ..., N\}$ be a set of labels. Given a training set $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_i \in X$ is a single instance and $y_i \subseteq Y$ is the label set associated with x_i, the goal is to design a multi-label classifier H that predicts a set of labels from an unseen example.

As discussed previously, ensembles methods are well-known for overcoming over-fitting problems especially in highly unbalanced data sets. Ensemble of multi-label classifiers train q multi-label classifiers H_1, H_2, ..., H_q. Thus, all q models are diverse and able to give different multi-label predictions. For an unseen instance x_j, each kth individual model (of q models) produces an N-dimensional vector $P_{jk} = [p_{1k}, p_{2k}, ..., p_{Nk}]$, where the value p_{bk} is the probability of the b^{th} class label assigned by classifier k being correct.

There are many ways of combining the outputs of these q classifiers. Among them, nontrainable combiners such as MEAN, MAX, MIN are the simplest and most popular way to combine the scores of classifiers with probabilistic outputs [16]. These combiners have no extra parameters to be trained. In this paper, nontrainable combiners are used to combine the scores from multi-label classifiers. The only exception is the adjustment of the thresholds using the method described below and proposed by [8]. It is reported in [17] that properly adjusting the decision thresholds (instead of the traditional value of 0.5) can improve the performance for a multi-label classifier. Let the sum of probabilities from q models be stored in a vector $W = (\theta_1, ..., \theta_N) \in \mathbb{R}^{\mathbb{N}}$ such that $\theta_b = \sum_{k=1}^{q} p_{bk}$. W is then normalised to W^{norm}, which represents a distribution of scores for each label in [0,1]. Let X_T be the training set and X_S the test set. A threshold t is then selected using Equation 1 to choose the final predicted multi-label set Z.

$$t = \arg \min_{\{t \in 0.00, 0.001, ..., 1.00\}} |LCard(X_T) - LCard(H_t(X_s))| \tag{1}$$

where LCard (Label Cardinality) is the standard measure of "multi-labelledness" [9]. It is the average number of labels relevant to each instance and is defined as $LCard(X) = \frac{\sum_{i=1}^{|X|} |E_i|}{|X|}$ where E_i is the actual set of labels for the training set and a predicted set of labels under threshold t for the test set. Equation 1 measures the difference between the label cardinality of the training

set and the predictions made on the test set. It avoids intensive internal cross-validation. Hence, the relevant labels in Z under threshold t represent the final predicted set of labels. It should be clear that the actual test labels are never seen by the presented threshold selection method. The threshold t is calculated using the predicted set of labels only.

4 Experiments

Datasets: We experimented with 3 multi-label datasets from a variety of domains. Table 1 shows certain standard statistics of these datasets. The publicly available feature vectors are used in this paper for all datasets [1]. The image dataset "scene" is concerned with semantic indexing of images of still scenes [10]. The "yeast" data set contains functional classes of genes in the Yeast Saccharomyces cerevisiae [11,7]. The "enron" is a subset of the Enron email corpus [15].

Table 1. Standard and multilabel statistics for the data sets used in the experiments

Datasets	Domain	Train	Test	Features	Labels	LCard
Enron	Text	1123	579	1001	53	3.38
Scene	Vision	1211	1196	294	6	1.07
Yeast	Biology	1500	917	103	14	4.24

Evaluation Measures: Multi-label classification requires different evaluation measures than traditional single-label classification. The details can be found in [9]. They are not shown here due to the lack of space. These measures can be categorised into three groups: example based, label-based and ranking-based. In this paper, 18 different evaluation measures are used to compare the proposed approach. These measures include Hamming Loss, Accuracy, Precision, Recall, F_1, and Classification Accuracy from the example-based category, and Micro Precision/Recall/F_1/AUC, Macro Precision/Recall/F_1/AUC from the label-based group. Additionally, we use One-error, Coverage, Ranking Loss and Average Precision from the ranking-based group.

Benchmark Methods: The proposed EML method is compared with the state-of-the-art multi-label classifiers discussed in Section 2: RaKEL [11], ECC [8], CLR [12] , MLKNN [13], and, IBLR [7]. Since all these multi-label classifiers are quite diverse, they are selected as base classifiers in the proposed EML method. MLKNN and IBLR are from the algorithm adaptation group while ECC, RaKEL and CLR are from the problem transformation group. Further, C4.5 is used as a base classifier in RaKEL while Linear SVM is used as a base classifier in ECC and CLR. For the training of MLKNN, IBLR, CLR and Rakel, the Mulan[1] open-source library in Java for multi-label classification is used. For the training of ECC, the MEKA [2] open-source library is used with the default parameters. Both

[1] http://mlkd.csd.auth.fr/multilabel.html
[2] http://www.cs.waikato.ac.nz/ jmr30/software

libraries are an extension of WEKA [18]. All multi-label classifiers are trained using default parameters which are also the best reported parameters e.g. the number of neighbours is 10 for IBLR and MLKNN; the number of iterations is 10 for ECC. The multi-label classifiers are compared with the following variants of the proposed method.

- EML_M: An ensemble of multilabel classifiers using the MEAN rule. It should be noted that we have also tried several others rules such as MAX, MIN and only the best is reported here due to lack of space.
- EML_T: An ensemble of multi-label classifiers using the threshold selection method discussed in Section 3.

5 Results and Discussion

In this section, we discuss the results obtained using EML for the Enron, Medical, Scene and Yeast datasets using Example, Label and Rank based measures.

5.1 Enron Dataset

Table 2 shows the comparison of EML with the state-of-the-art multilabel classifiers for the Enron dataset. First, when the individual multi-label classifiers are compared with each other using various performance measures, it is hard to pick between CLR and RakEL for this dataset. RakEL delivers excellent performance in the majority of Example-based measures while CLR gives the best performance in the majority of Label and Rank-based measures. Overall, CLR and RakEL show the best performance in eight and six evaluation measures respectively. MLKNN and ECC are also the winners in 2 measures. However, by using an ensemble of multi-label classifiers, significant performance gains have been observed in almost all measures. It is also observed that the presented

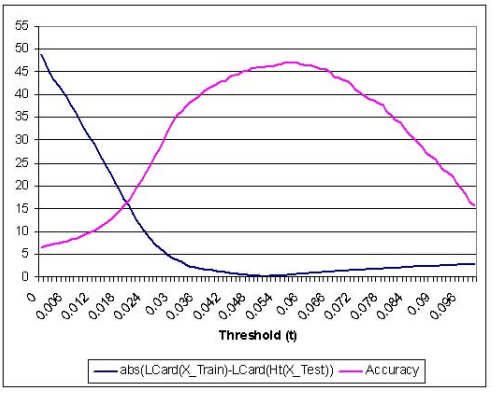


Fig. 1. Threshold t vs $\{|LCard(X_T) - LCard(H_t(X_s))|$,Accuracy$\}$ for Enron

Table 2. Comparison of proposed ensemble method (EML) with the state-of-the-art multi-label classifiers for *Enron* dataset. For each evaluation criterion, ↓ indicates "the smaller the better" while ↑ indicates "the higher the better".

	CLR	RakEL	MLKNN	IBLR	ECC	EML_M	EML_T
HammingLoss ↓	0.062	0.050	0.052	0.057	0.063	**0.047**	0.051
Accuracy ↑	0.401	0.429	0.352	0.337	0.405	0.411	**0.461**
Precision ↑	0.539	0.640	0.630	0.557	0.523	**0.698**	0.608
Recall ↑	0.577	0.509	0.390	0.396	0.589	0.447	**0.626**
Fmeasure ↑	0.557	0.567	0.482	0.463	0.554	0.545	**0.617**
SubsetAccuracy ↑	0.083	0.131	0.093	0.086	0.071	**0.133**	0.071
Micro Precision ↑	0.516	0.651	0.680	0.594	0.507	**0.746**	0.603
Micro Recall ↑	0.553	0.485	0.363	0.379	0.558	0.416	**0.603**
Micro F_1 ↑	0.534	0.556	0.473	0.463	0.531	0.534	**0.603**
Macro Precision ↑	**0.297**	0.213	0.163	0.220	0.217	0.212	0.256
Macro Recall ↑	**0.269**	0.137	0.078	0.130	0.229	0.099	0.169
Macro F_1 ↑	**0.257**	0.149	0.088	0.144	0.210	0.118	0.175
Micro AUC ↑	0.904	0.816	0.900	0.880	0.856	**0.918**	0.916
Macro AUC ↑	0.723	0.592	0.632	0.619	0.662	**0.724**	0.712
One-error ↓	0.309	0.287	0.269	0.378	0.307	**0.238**	**0.238**
Coverage ↓	11.957	24.073	12.751	14.534	19.411	**11.218**	**11.218**
Ranking Loss ↓	0.085	0.195	0.092	0.109	0.148	**0.075**	**0.075**
AvgPrecision ↑	0.649	0.612	0.641	0.612	0.624	**0.699**	**0.699**
# Wins (Ind. Classi.)	8/18	6/18	2/18	0/18	2/18	-	-
# Wins (All)	3/18	0/18	0/18	0/18	0/18	10/18	9/18

threshold selection technique (EML_T) has a significant effect on some evaluation measures. For example, there is a 12% increase in Accuracy while 48% increase in Macro F_1. In summary, a significant improvement is observed by using both variants of the proposed ensembles of multi-label classifier techniques (EML_M and EML_T).

Figure 1 shows the graph for different values of threshold t in the X-axis and $\{|LCard(X_T) - LCard(H_t(X_s))|$,Accuracy$\}$ in the Y-axis for the Enron data set. For clarity, only values with $t < 0.1$ are shown here as optimal value of the threshold selection measure is between 0 and 0.1 for this dataset. It is clear from this graph that the threshold selection method is able to deliver a near-optimal value of accuracy. The optimal value of accuracy is 47.2 under threshold $t = 0.056$ (accuracy curve is plotted using actual test labels for demonstration only). In contrast, the best value of accuracy obtained by Equation 1 is 46.1 under $t = 0.051$ which is very close to the optimal one. In summary, this graph clearly shows the merit of the presented threshold selection method as this simple approach attains near-optimal values without expensive internal cross validation.

5.2 Scene Dataset

Table 3 shows the comparison of EML with the state-of-the-art multilabel classifiers for the Scene dataset. It is interesting to observe that for this data set, IBLR and ECC achieve the highest performance in most of the measures when compared with the individual multi-label classifiers. Overall, IBLR, ECC and CLC

Table 3. Comparison of proposed ensemble method (EML) with the state-of-the-art multi-label classifiers for *Scene* dataset

	CLR	RakEL	MLKNN	IBLR	ECC	EML_M	EML_T
HammingLoss ↓	0.122	0.112	0.099	0.091	0.109	**0.084**	0.095
Accuracy ↑	0.577	0.571	0.629	0.647	0.683	**0.699**	0.694
Precision ↑	0.600	0.598	0.661	0.676	0.716	**0.730**	0.725
Recall ↑	0.669	0.612	0.655	0.655	0.727	0.716	**0.754**
Fmeasure ↑	0.632	0.605	0.658	0.665	0.722	0.723	**0.740**
SubsetAccuracy ↑	0.474	0.503	0.573	0.609	0.605	**0.651**	0.602
Micro Precision ↑	0.666	0.732	0.779	**0.824**	0.696	0.812	0.737
Micro Recall ↑	0.659	0.600	0.634	0.635	0.708	0.696	**0.737**
Micro F_1 ↑	0.663	0.660	0.699	0.717	0.702	**0.750**	0.737
Macro Precision ↑	0.680	0.729	0.784	**0.827**	0.713	0.817	0.764
Macro Recall ↑	0.662	0.609	0.647	0.642	0.715	0.701	**0.741**
Macro F_1 ↑	0.669	0.663	0.692	0.719	0.712	**0.754**	0.748
Micro AUC ↑	0.916	0.894	0.924	0.931	0.901	**0.949**	0.943
Macro AUC ↑	0.910	0.886	0.911	0.927	0.898	**0.943**	0.938
One-error ↓	0.261	0.293	0.242	0.234	0.273	**0.216**	**0.216**
Coverage ↓	0.543	0.691	0.569	0.551	0.639	**0.451**	**0.451**
Ranking Loss ↓	0.088	0.117	0.093	0.090	0.106	**0.070**	**0.070**
AvgPrecision ↑	0.845	0.817	0.851	0.856	0.831	**0.873**	**0.873**
# Wins (Ind. Classi.)	2/18	0/18	0/18	10/18	6/18	-	-
# Wins (All)	0/18	0/18	0/18	2/18	0/18	12/18	8/18

give the best performance in ten, six and two evaluation measures respectively. However, by using the proposed ensemble of multi-label classifiers (EML), significant performance gains have been observed in all measures except Micro/Macro Precision. In summary, in addition to performance gains, fusion of multi-label classifiers also has overcome some limitations of individual multi-label classifiers as the performance of these individual multi-label classifiers may vary in evaluation measures and from one data set to another.

5.3 Yeast Dataset

Table 4 shows the comparison of EML with the state-of-the-art multilabel classifiers for the Yeast dataset. First, when the individual multi-label classifiers are compared with each other using various performance measures, ECC, MLKNN and IBLR demonstrate very good performance on this dataset. IBLR ranks first in the ranked-based measures while the performance vary for ECC and MLKNN in Example and Label based measures. Overall, ECC, IBLR and MLKNN report the best performance in seven, six and four evaluation measures respectively. As before, the fusion of multi-label classifiers has significantly improved the overall performance in all except classification accuracy. It is also observed that the presented threshold selection technique (EML_T) makes a significant impact on the performance especially on example-based measures e.g. 11% and 8.5% increase in Accuracy and Fmeasure respectively when compared with an ensemble using the MEAN rule (EML_M) for fusion.

Table 4. Comparison of proposed ensemble method (EML) with the state-of-the-art multi-label classifiers for *Yeast* dataset

	CLR	RakEL	MLKNN	IBLR	ECC	EML_M	EML_T
HammingLoss ↓	0.210	0.244	0.198	0.199	0.212	**0.193**	0.197
Accuracy ↑	0.497	0.465	0.492	0.506	0.535	0.500	**0.553**
Precision ↑	0.674	0.601	0.732	0.712	0.654	**0.738**	0.682
Recall ↑	0.596	0.618	0.549	0.581	0.669	0.553	**0.690**
Fmeasure ↑	0.633	0.609	0.628	0.640	0.661	0.633	**0.686**
SubsetAccuracy ↑	0.158	0.091	0.159	0.176	**0.197**	0.166	0.190
Micro Precision ↑	0.681	0.596	0.736	0.714	0.648	**0.750**	0.676
Micro Recall ↑	0.585	0.616	0.543	0.576	0.656	0.548	**0.677**
Micro F_1 ↑	0.629	0.605	0.625	0.637	0.652	0.633	**0.677**
Macro Precision ↑	0.447	0.430	0.600	0.560	0.460	**0.689**	0.504
Macro Recall ↑	0.364	0.420	0.308	0.342	0.423	0.315	**0.428**
Macro F_1 ↑	0.382	0.407	0.336	0.371	0.403	0.352	**0.420**
Micro AUC ↑	0.814	0.785	0.835	0.840	0.806	**0.844**	0.840
Macro AUC ↑	0.658	0.626	0.664	0.686	0.642	**0.708**	0.697
One-error ↓	0.251	0.338	0.234	**0.232**	0.278	0.240	0.240
Coverage ↓	6.589	7.834	6.414	6.350	7.067	**6.297**	**6.297**
Ranking Loss ↓	0.181	0.233	0.172	0.169	0.218	**0.163**	**0.163**
AvgPrecision ↑	0.749	0.693	0.758	0.760	0.730	**0.766**	**0.766**
# Wins (Ind. Classi.)	0/18	1/18	4/18	6/18	7/18	-	-
# Wins (All)	0/18	0/18	0/18	1/18	1/18	9/18	10/18

5.4 Discussion

The results presented in this paper show the merit of combining multi-label classifiers. In all three datasets, it is hard to pick a multi-label method that can perform consistently well. For example while CLC and RakEL performs quite well on Enron, they do not deliver similar superiority on Scene and Yeast when compared with ECC, MLKNN and IBLR. Furthermore, since multi-label data suffers from class imbalance problem, it is natural to apply ensemble techniques to overcome over-fitting and improve the accuracy of individual classifiers. Both EML_M and EML_T improve consistently when compared with the individual methods and across the majority of evaluation measures. However, this performance gain is at the expense of inherent computational complexity of ensemble techniques since several multi-label classifiers need to be trained separately. The easiest solution is to use parallel computing techniques to improve the efficiency since all base classifiers can be trained independently.

To the best of our knowledge, this is the first study that aims to combine the output of various multi-label classifiers. In this paper, we have investigated nontrainable ensemble techniques based on the MEAN rule and threshold selection. Since, multi-label classifiers inherently are computationally intensive and data is highly imbalanced, it opens new research challenges how to use other combination techniques efficiently such as trainable combiners (Weighted Average, Fuzzy Integral) or class indifferent combiners (Decision Templates and Dempster-Shafer Combination). The other interesting research issue that needs

to be investigated is how to select the base classifiers in EML since different combinations of base classifiers may perform differently for specific problem domains.

6 Conclusion

In this paper, heterogeneous ensemble of multi-label learners is presented to simultaneously tackle both imbalance and correlation problems. For multi-label classification, this idea is especially appealing, as ensembles methods are well-known for overcoming over-fitting problems and improving the performance of individual classifiers. Two nontrainable ensemble techniques based on the MEAN rule and threshold selection are investigated and then applied to three publicly available multi-label data sets using several evaluation criteria. It has been shown that the presented approach provides a very accurate and efficient solution when compared with the state-of-the-art multi-label methods.

Acknowledgements. This work was supported by the EU VIDI-Video Project.

References

1. Li, T., Ogihara, M.: Toward intelligent music information retrieval. IEEE Trans. on Multimedia 8(3), 564–574 (2006)
2. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004)
3. Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans. on Knowledge and Data Engineering 18(10), 1338–1351 (2006)
4. Tahir, M.A., Kittler, J., Yan, F., Mikolajczyk, K.: Kernel discriminant analysis using triangular kernel for semantic scene classification. In: Proc. of the 7th International Workshop on CBMI, Crete, Greece. IEEE, Los Alamitos (2009)
5. Dimou, A., Tsoumakas, G., Mezaris, V., Kompatsiaris, I., Vlahavas, I.: An empirical study of multi-label learning methods for video annotation. In: Proc. of the 7th International Workshop on CBMI, Chania, Greece (2009)
6. Chawla, N.V., Sylvester, J.C.: Exploiting diversity in ensembles: Improving the performance on unbalanced datasets. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 397–406. Springer, Heidelberg (2007)
7. Cheng, W., Hullermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
8. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS (LNAI), vol. 5782, pp. 254–269. Springer, Heidelberg (2009)
9. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: Data Mining and Knowledge Discovery Handbook, 2nd edn. Springer, Heidelberg (2009)
10. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition 37(9), 1757–1771 (2004)

11. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007)
12. Furnkranz, J., Hullermeier, E., Menca, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning 23(2), 133–153 (2008)
13. Zhang, M.L., Zhou, Z.H.: ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition 40(7), 2038–2048 (2007)
14. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in NIPS, vol. 14 (2002)
15. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: Perner, P. (ed.) ICDM 2008. LNCS (LNAI), vol. 5077. Springer, Heidelberg (2008)
16. Kuncheva, L.I.: Combining Pattern Classifiers. Wiley, Chichester (2004)
17. Fan, R.E., Lin, C.J.: A study on threshold selection for multi-label classification. Technical report, National Taiwan University (2007)
18. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco (2005)

New Feature Splitting Criteria for Co-training Using Genetic Algorithm Optimization

Ahmed Salaheldin[1] and Neamat El Gayar[1,2]

[1] Center for Informatics Science, Nile University, Giza, Egypt
[2] Faculty of Computers and Information, Cairo University, 12613 Giza, Egypt

Abstract. Often in real world applications only a small number of labeled data is available while unlabeled data is abundant. Therefore, it is important to make use of unlabeled data. Co-training is a popular semi-supervised learning technique that uses a small set of labeled data and enough unlabeled data to create more accurate classification models. A key feature for successful co-training is to split the features among more than one view. In this paper we propose new splitting criteria based on the confidence of the views, the diversity of the views, and compare them to random and natural splits. We also examine a previously proposed artificial split that maximizes the independence between the views, and propose a mixed criterion for splitting features based on both the confidence and the independence of the views. Genetic algorithms are used to choose the splits which optimize the independence of the views given the class, the confidence of the views in their predictions, and the diversity of the views. We demonstrate that our proposed splitting criteria improve the performance of co-training.

1 Introduction

Supervised learning uses a large number of labeled examples to build classifiers. Such labeled data sets are not always available or need a lot of effort and time to be collected.

Semi-supervised learning tries to overcome this limitation by inducing high accuracy classifiers from small sets of labeled data and enough unlabeled data which may be obtained easily [1].

Co-training is a popular semi-supervised learning technique; that creates classification models using labeled and unlabeled data. It does so by training more than one classifier using different versions of the labeled data called views [2]. The classifiers are then used to predict the labels of the unlabeled data pool. Each classifier then chooses the predictions in which it is most confident and teaches them to the rest of the classifiers. Co-training is suitable for many real life applications because it can utilize unlabeled data given only a small labeled set.

However there are some requirements for the successful use of co-training: namely that the data is redundant i.e. Information useful for classification is redundant in both views so each view is sufficient for classification on its own, and that the features in different views are independent given the class [2]. These

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 22–32, 2010.

requirements are met if the features have a natural split. As an example Blum and Mitchell [2] use a data set of websites where some features are taken from the text in the webpage and the rest from the hyper links that lead to the page.

In many cases a natural split does not exist or is unknown. Therefore, finding good artificial splits has been a research topic for years. Nigam et al. use random splitting of features into two views and demonstrate that co-training could be successful using a random split if the data is redundant enough[3].

Feger et al propose a graph based algorithm that maximizes the independence of the two views based on the concept of mutual independence[4]. Wang et al. show that the co-training process could succeed without splitting the data into two views, given that the two base classifiers have large differences[5].

There are also efforts to split the features among classifiers in supervised multiple classifier systems. Opitz in [6] presents a genetic algorithm for creating ensembles called Genetic Ensemble Feature Selection. In this study a genetic algorithm is used to select the split that maximizes the accuracy of the individual classifier as well as the diversity of the ensemble by measuring the difference between the accuracy of the individual classifiers and that of the ensemble. Global optimization techniques such as genetic algorithms can be used to select feature splits in co-training however in co-training the labels of the data are not available so measures such as the accuracy of the classifier cannot be calculated.

In this paper we investigate different artificial feature splits to be used in conjunction with the co-training algorithm. In particular, we use genetic algorithms to select the split that optimize different splitting criteria. We use the same conditional mutual information equation used by Feger et al. in [4] , as the fitness function of the genetic algorithm to minimize the independence of the views given a class. More importantly in this study we attempt to propose new feature splits based on maximizing the strength of the individual views. Again, we use a genetic algorithm to maximize the strength of the individual views and their diversity.

The contribution of this work can be summarized as follows.

Firstly we propose a new criterion for splitting the features in a co-training setting based on maximizing the strength of the individual views. Since we cannot measure the accuracy of an individual view with unlabeled data, we devise a fitness function that measures the confidence of a view based on the entropy of its output.

Secondly, we aim to satisfy both requirements of co-training by creating a mixed fitness function that takes into consideration both the confidence of the individual views and their mutual independence.

Thirdly we introduce a genetic algorithm that maximizes the strength of the individual views and their diversity. This criterion is similar in concept to that used by Opitz[6] to create feature subset ensembles. However due to the absence of labels necessary to calculate the accuracy of the individual views -and the improvement in accuracy of the combined output- we use the confidence of the output as an indication of accuracy.

We compare the aforementioned splits against random splits and against the natural split if it exists. Experiments are conducted on two real data sets using

Multilayer Perceptron (MLP) and Support Vector Machines (SVM) as the base classifiers for co-training.

The paper is organized as follows: In the next section we explain the co-training algorithm. The different criteria for splitting the features are described in section 3 and 4. Section 5 explains the genetic algorithm and how we use it to select the feature splits. In sections 6 we present our experiments and discuss our findings. Section 7 summarizes the paper and provides suggestions for future work.

2 Co-training

Co-training is a semi-supervised learning method which can induce high accuracy classifiers from a small number of labeled and a large number of unlabeled data. First, the features in the original data set are split into two feature sets V1 and V2, which are called views. Two classifiers C1 and C2 are each trained using one view of the labeled set L. The two classifiers C1 and C2 are different because they are trained with different views of the data. Next, all of the examples in the unlabeled data pool U'- randomly selected from a large unlabeled set U- are classified by the classifiers C1 and C2. The most confidently classified examples by each classifier are added to the labeled dataset. U' is then replenished with examples randomly selected from the unlabeled data set U. By iterating such a process, the number of labeled examples L is increased by the self-labeled examples. As a result, the prediction accuracy of the classifiers is improved. Figure 1 summarizes the steps of the co-training algorithm.

3 Natural, Random and Independent Feature Splits

In this paper we compare splitting the features according to different criteria. Previous research on co-training mainly used natural splits, random splits and splits devised through maximizing the independence between views. We will review these methods as follows.

3.1 Natural Splits

A natural split exists when the data is naturally described by two disjoint feature sets. In this case, each feature set is sufficient to perform classification and the views are truly independent. For example in an email spam classification problem, one view may contain the features describing the subject of the email and the other may contain the features describing the body. Natural splits satisfy the co-training requirements proposed by Blum and Mitchell, they showed that using unlabeled data for co-training improves the performance when a natural split exists[2].

Input:

- **L:** a small set of labeled example
- **U:** a large set of unlabeled example
- **V1, V2:** two sets of describing the example

Algorithm:

- Create a pool **U'** by randomly choosing **u** examples from **U**
- Loop for **k** iterations
- Train Classifier **C1** from **L** based on **V1**
- Train Classifier **C2** from L based on **V2**
- **C1** predicts the class of examples from **U'** based on **V1** and chooses the most confidently predicted **p** positive and n negative examples **E1**
- **C2** predicts the class of examples from **U'** based on **V2** and chooses the most confidently predicted p positive and n negative examples **E2**
- **E1** and **E2** are removed from **U'** and added with their labels to **L**
- Randomly chose **2p+2n** examples from **U** to replenish **U'**
- End

Fig. 1. Co-training algorithm

3.2 Random Splits

Randomly splitting the data is a simple way of creating an artificial split when a natural split does not exist. Nigam et al showed that co-training works with a random split if the data is redundant enough[3].

3.3 Independence of Views

Feger et al [4] proposed an algorithm to measure the independence of views in co-training -which is one of two requirements for successful co-training- based on the conditional mutual information of the features. The Mutual Information (MI) of two features X and Y depends on their entropies H and is defined as follows:

$$MI(X,Y) = H(X) - H(X|Y) = \sum_{xeX} \sum_{yeY} p(x,y) \log_2 \frac{p(x,y)}{f(x)g(y)} \tag{1}$$

$CondMI$ is MI under the condition of an additional constraint:

$$CondMI(X,Y|z=c) = \sum_{xeX} \sum_{yeY} p(x,y|z=c) \log_2 \frac{p(x,y|z=c)}{p(x|z=c)p(y|z=c)} \tag{2}$$

The interdependence of the views is calculated as the sum of the conditional mutual information of all the features in the two views.

The probabilities used in the $CondMI$ are calculated based on the frequency of a feature X having the value x i.e. $P(X = x)$ is the number of records in which $X = x$ over the total number of records. Feger et al. [4] use a graph based method to select the split that minimizes the dependence between the two views. Since the number of features may be very large, exhaustively calculating the dependence between all possible views is computationally infeasible so the split is produced using a graph partition heuristic[4]. Alternatively we use a genetic algorithm to select the split that minimizes the dependence. This is described in more details in the following sections.

4 Proposed Feature Splitting Criteria

Only one of the two requirements for co-training has been addressed in previous work which is the independence of the views. We propose to create a feature split that maximizes the strength of the individual views based on their confidence. We also suggest using a mixed function that considers both requirements. Moreover we introduce a new criterion for splitting the features based on maximizing the strength of the views and their diversity to take advantage of the co-training paradigm. As follows we describe our proposed measures in more details.

4.1 Confidence of the Views

The first requirement for successful co-training is that the features are redundant enough, that is each view is strong enough to perform classification on its own. Based on that hypothesis we propose a genetic algorithm to select the split that maximizes the strength of the views. Since the accuracy of the views cannot be measured because the data is not fully labeled; we use the confidence of a view as an indication of its strength.

The confidence of a view is measured by calculating the average entropy of its output. Entropy is a general measure of the uncertainty. In our case, we use entropy to measure the uncertainty or fuzziness of an output. The entropy $H(X)$ of a variable X is given by:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{3}$$

Where b is the base of the logarithm used.

Each classifier is trained with a view of the labeled dataset; it is then used to predict the class of the unlabeled data set. The entropy of the output produced by each input sample is calculated and the average of this over all input samples indicates the uncertainty of the view. To obtain the split with the most confident views we minimize $E(V_1) + E(V_2)$, where $E(V_i)$is the average entropy of the outputs of view i.

4.2 Mixed Criteria

We also propose to create a split that optimizes both requirements of co-training by simply considering both measures of confidence and independence described above. We aim to minimize $E(V_1) + E(V_2) + InterCMI(V_1, V_2)$, where $E(V_1)$ is the uncertainty of the first view, $E(V_2)$ is the uncertainty of the second view, and $InterCMI(V_1, V_2)$ is the dependence between them.

4.3 Maximum Diversity

The Independence of the views is considered important in the co-training paradigm because each view adds different information to the model like in natural splits. However, in their paper Feger et al. reported that their proposed algorithm for maximizing independence did not improve the accuracy over random splits. Because only a small number of labeled data is available, their maximum independence split was not actually much more independent than the random splits. They used a labeled version of the entire data set to produce a split that was much more independent, however this split could not be used in co-training because the labels for the entire data set would not be available[4]. Therefore we need other methods to ensure that each view adds different information to the model, so we propose to use feature splits that maximize the diversity between the views. Opitz used genetic algorithms to split the features among an ensemble of classifier in a way that would maximize their diversity. He measured diversity as the difference between the accuracies of individual classifier in the ensemble and the combined output of the ensemble. This represents the improvement of the combined output over the individual outputs[6]. We use a similar approach to maximize the diversity between the views in co-training, but since the data is unlabeled we could not measure the accuracy of the output so again we use the confidence in the output to indicate accuracy. We use the equation

$$E(V_1) + E(V_2) + \frac{E(V_1) + E(V_2)}{E(V_1 * V_2)} \tag{4}$$

where $E(V_1 * V_2)$ is the entropy of the combined output. This equation minimizes the entropy of the outputs of individual views and maximizes the entropy of the combined output thus maximizing conflict between confident views.

5 Genetic Algorithm Optimization

Genetic Algorithms are search technique to find approximate solutions to optimization problems. Genetic Algorithms mimic the behavior of natural selection to reach an optimum solution to any multi-dimensional problem[7].An optimal solution is however not guaranteed. Genetic Algorithms attempt to find the best solution to a problem by generating a population of potential solutions to the problem in the form of genomes. The quality of the solution is calculated by a fitness function, whereas the best solution is the minimum of that function. A

new generation of solutions is produced by applying cross over and mutation to the current population genomes. Only the fittest solutions survive and are used to produce the next generation. This process continues until an acceptably good solution is found.

Genetic algorithms are useful in selecting feature splits because finding the optimal split exhaustively is not computationally feasible, moreover the feature vector could be easily represented as a genome for the genetic algorithm. Genetic algorithms have been successfully used to optimize feature subset ensembles [6] in the context of supervised learning[8]. In supervised learning the accuracy of the model is used as the fitness function. However, in semi-supervised learning, creating a fitness function becomes more challenging.

In this paper we use genetic algorithms to select feature splits that optimize fitness functions that measure the confidence, independence and diversity of the views. Solutions are represented as bit strings of the same size as the feature vector. If bit number i equals 1, then feature i in the feature vector belongs to view1; alternatively, if the bit equals 0, the feature belongs to view2.

6 Experiments

6.1 Experimental Setup

As follows we will present the details of our experiments. In particular we describe the data sets, base classifiers and evaluation criteria used.

We use three real data sets to test the feature splitting criteria. The first two data sets used are from the UCI Machine Learning Repository[9]. The Congressional Voting Records data set consisting of categorical attributes and Breast Cancer Wisconsin data set consisting of numerical attributes. In addition we use a data set of fruit images(The Fruits data set) previously collected for a robotic application. The data set consists of 840 objects for 7 classes of fruits (green apple, red apple, tangerine, orange, yellow plum, red plum and lemon). 12 different feature groups are generated for each object. Feature groups are described in different levels of detail; sometimes up to 1024 features. For more details on the collection and description of this data set refer to [10]. In our experiments we use the feature groups of CANNY and color histogram; as our natural splits. As this paper presents only preliminary results for testing several feature splitting criteria, we only experiment with two fruit classes.

As co-training can be dependent on the base classifier[11] in the ensemble, we repeat our experiments using two base classifiers (MLP and SVM).

A MLP is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output. We use a MLP with one hidden layer of 20 nodes. The transfer functions used are *tan sigmoid* for the hidden layer and *pure linear* for the output layer. The training function used is a gradient descend function with learning rate 0.1. On the other hand, SVMs are machine learning techniques that use a hyper plane or a set of hyper planes to divide a high-dimensional space. We use a SVM classifier with a Gaussian RBF kernel for our experiments.

As mentioned before, a genetic algorithm is used to minimize the fitness functions based on the feature splitting criteria. Because genetic algorithms may give many suboptimal solutions, each fitness function is used to generate 10 solutions. The results of the 10 solutions are then averaged. The population size used is 20 samples and the number of generations is 100 generations.

We test the feature splitting criteria described in sections 3 and 4 using the above mentioned datasets and base classifiers. 10 Random splits are generated where each feature belongs to one of two views. Fitness functions are formulated to calculate the independence of the views as described in section 3 and our new splitting criteria as described in section 4. A genetic algorithm is used as described above to generate the solutions based on the splitting criteria. Co-training is run 10 times for each split, and the average of the error is calculated. We also test the natural split in the fruits data set.

6.2 Results

First to test if the confidence is really a good indication of accuracy, we train the classifier using the views with the most confidence (without co-training) and test them against classifiers trained using random subsets of the feature vector.

The following table shows the errors of using most confident views against random views, where $\sum$ views is the sum of the errors of the individual views.

Table 1. Error in confident views against random views

Base Classifier	$\sum$ confident views	$\sum$ random views
MLP Votes dataset	33%	47.1%
SVM Votes dataset	39.4%	36.5%
MLP Fruits dataset	58.1%	72.9%
SVM Fruits dataset	23.5%	13.5%
MLP Breast Cancer dataset	31.7%	37.2%
SVM Breast Cancer dataset	68.7%	67.1%

Table 1 shows that the confidence of a view is a good indication of its strength when using Multilayer perceptron but not when using SVM.

Next we test co-training using the features splits described in sections 3 and 4. Tables 2-5 summarize the errors for co-training after 10 iterations using the

Table 2. Error in co-training on UCI data using MLP

Votes	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity
$E_{1,2}$	14.1%	14.95%	14.38%	14.55%	12.17%
CE	13%	11.34%	12.13%	11.1%	10.93%
Breast Cancer	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity
$E_{1,2}$	16.55%	16.47%	14.52%	12.97%	11.51%
CE	14.18%	12.91%	13.3%	10.4%	9.96%

Table 3. Error in co-training on Fruits data using MLP

	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity	Natural
$E_{1,2}$	25.625%	23.06%	23.45%	22.435%	12.975%	14.765%
CE	26.37%	22.62%	22.84%	22.125%	12.8%	13%

Table 4. Error in co-training on UCI data using SVM

Votes	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity
$E_{1,2}$	14.705%	14.755%	16.4%	14.2%	15.6%
CE	14.24%	13.6%	13.6%	13.32%	13.49%
Breast Cancer	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity
$E_{1,2}$	47.83%	44.24%	42.39%	44.44%	36.86%
CE	52.47%	46.73%	39.68%	39.065%	36.15%

Table 5. Error in co-training on Fruits data using SVM

	Random Split	Independence	Confidence	Mixed(conf+Ind)	Diversity	Natural
$E_{1,2}$	9.75%	9.875%	8.475%	7.62%	10.925%	10.815%
CE	9.5%	10.87%	9.45%	9.19%	11.1%	10.25%

different splitting criteria for the MLP and the SVM base classifiers for the three data sets. $E_{1,2}$ is the average error of the individual views, CE is the error of the combined output.

From the previous tables we observe the following:

For MLP, the split maximizing the diversity gives the best performance for all data sets. The mixed split also performs well on all data sets, and outperforms splits that only optimize one requirement of co-training. We further observe that maximizing the confidence of the views gives better results than randomly splitting the features. This is expected since confidence implies the strength of views when using MLP. This gives credit to the requirement put forth by Blum and Mitchell [2] that the strength of the views is important for co-training. Terabe et al. who used the same votes data set similarly reported that splits with stronger views perform better than splits with weaker views when using RBF Neural networks, however they did not test this by generating optimized splits but rather by randomly creating a number of splits and testing the splits with the strongest views against the ones with the weakest views[12]. They also calculated the accuracy by testing the models with labeled data which could not be used in a real co-training application. Splits that maximize independence also give less error than random splits. Feger et al. reported that maximizing the independence between the views did not improve the performance, however they showed that on their data set, using the maximum independence algorithm on a small labeled data set did not produce a split that was significantly more independent than the random splits[4]. Terabe et al found that Independence of the views when using RBF Neural networks helped co-training on the votes data

set[12] , again they did not generate an optimized split but randomly generated a number of splits and tested the most independent splits against the least independent ones.

For SVM, the diversity and mixed splitting criteria performed well on the votes and breast cancer data sets. Again the splits that take into consideration both confidence and independence of the views perform better than splits that only consider one of the two criteria. The maximum confidence split improves on random splits even though in SVM the confidence of a view does not necessarily imply its strength. On the Fruits data set the different splits generally do not improve on the random split, only the maximum confidence and the mixed split improved slightly. Even the natural split performed worse than the random split. Feger et al[4] similarly reported that a natural split is outperformed by the random split under SVM.

7 Summary and Future Work

In this paper we propose three new criteria for splitting features in co-training and compare them against existing artificial splits and natural split.

We propose a genetic algorithm to generate feature splits with optimum confidence of views. We also propose a feature split that maximizes both confidence of the views and the independence of the views. The independence of the views is calculated using the equation of conditional mutual information used before by [4]. We show that satisfying both requirements of co-training with a mixed fitness function is better than using only one of the two criteria to split the features. Finally we propose a third criterion for splitting the features based on maximizing the views' diversity. The concept of maximizing the diversity of feature subsets was used before by [6] in supervised learning ensembles. We use a fitness function that calculates the confidence of the views and their diversity, and use a genetic algorithm to maximize this function. Our empirical results on two data sets show that our proposed splits are promising alternatives to randomly splitting the data.

We confirm that co-training depends on the base classifier[11] and that Neural networks performs better under co-training than SVM[4].

Currently we are in the process of testing our new splitting criteria on more data sets, using more base classifiers, and different versions of co-training to investigate their effect on the performance. We also propose to extend our experiments to more than two views in the future.

Acknowledgment

We would like to acknowledge the Information Technology Industry Development Agency (ITIDA), Ministry of Communication and Information technology, Egypt for their financial support of the Center for Informatics Science, Nile University. This work is partially funded by the DFG (German Research Society) grants SCHW 623/3-2 and SCHW 623/4-2. We would also like to thank

Dr Friedhelm Schwenker from the Department of Neural Information Processing, Ulm University, Germany for providing us with the fruits data set.

References

1. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational Learning Theory (COLT 1998), pp. 92–100 (1998)
3. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: Proceedings of the Ninth International Conference on Information and Knowledge (CIKM 2000), pp. 86–93 (2000)
4. Feger, F., Koprinska, I.: Co-training using rbf nets and different feature splits. In: Proceedings of 2006 International Joint Conference on Neural Network, pp. 1878–1885 (2006)
5. Wang, W., Zhou, Z.-H.: Analyzing co-training style algorithms. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 454–465. Springer, Heidelberg (2007)
6. Opitz, D.: Feature selection for ensembles. In: Proceedings of the 16th International Conference on Artificial Intelligence, pp. 379–384 (1999)
7. Goldberg, D.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading (1989)
8. Kamp, R.G., Savenije, H.H.G.: Optimising training data for anns with genetic algorithms. In: Hydrol. Earth Syst. Sci., pp. 603–608 (2006)
9. Asuncion, A., Newman, D.J.: Uci machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
10. Rebecca Fay, F.S., Kaufmann, U., Palm, G.: Learning object recognition in a neurobotic system. In: Groß, H.-M., Debes, K., Böhme, H.-J. (eds.) 3rd Workshop on SelfOrganization of AdaptiVE Behavior, SOAVE 2004, pp. 198–209 (2004)
11. Kiritchenko, S., Matwin, S.: Email classification with co-training. In: Proceedings of CASCON 2001, Toronto, Canada, pp. 192–201 (2001)
12. Terabe, M., Hashimoto, K.: Evaluation criteria of feature splits for co-training. In: Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 (2008)

Incremental Learning of New Classes in Unbalanced Datasets: Learn^{++}.UDNC

Gregory Ditzler, Michael D. Muhlbaier, and Robi Polikar*

Signal Processing and Pattern Recognition Laboratory
Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028 USA
ditzle53@students.rowan.edu, rpolikar@rowan.edu

Abstract. We have previously described an incremental learning algorithm, Learn^{++}.NC, for learning from new datasets that may include new concept classes without accessing previously seen data. We now propose an extension, Learn^{++}.UDNC, that allows the algorithm to incrementally learn new concept classes from unbalanced datasets. We describe the algorithm in detail, and provide some experimental results on two separate representative scenarios (on synthetic as well as real world data) along with comparisons to other approaches for incremental and/or unbalanced dataset approaches.

Keywords: Incremental Learning, Ensembles of Classifiers, Learn^{++}, Unbalanced Data.

1 Introduction

Incremental learning requires an algorithm that is capable of learning from new data that may introduce new concept classes, while retaining the previously acquired knowledge without requiring access to old datasets. The ability to learn new information and retaining existing knowledge are often conflicting in nature, which is commonly known as the stability-plasticity dilemma [1]. Ensemble based systems typically provide a good balance between the two by simultaneously increasing the memory (to aid stability) and learning capacity (to aid plasticity) of the learning algorithm. An ensemble based algorithm can add new classifiers to learn the new data and keep the previous classifiers to retain the existing knowledge. However, such an approach has its own shortcomings: there will always be classifiers trained on a subset of the concept classes, which are hence guaranteed to misclassify instances from classes on which they were not trained, potentially out-voting classifiers that were trained on such classes. While there are several ensemble-based incremental learning algorithms, such as those proposed in [2], [3], [4], this issue of out-voting is only explicitly addressed in Learn^{++}.NC (*N*ew *C*lass) [2], through a dynamic consult and vote approach. This approach allows each classifier to predict – based on the votes of others – whether it has been trained on a specific concept class, and withhold its vote on instances of classes that it predicts that it has not seen [2]. Learn^{++}.NC have previously been shown to provide favorable accuracy and parsimony properties compared to its predecessor

* Corresponding Author.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 33–42, 2010.

Learn^{++}, as well as other ensemble based algorithms that are capable of incremental learning, such as dynamic weighted majority, bagging, AdaBoost and arc-x4. Learn^{++}.NC, however, was not designed to address –and hence could not handle – class imbalance, nor can it address the rare but pathological scenario of adding new classes while simultaneously removing existing ones on a subsequent dataset. Learning from unbalanced data was previously attempted in an algorithm called Learn^{++}.UD [5], however, that algorithm was not able to learn new classes. In this paper, we propose a new algorithm that is capable of learning new classes, even in the presence of relatively unbalanced data (including datasets with multiple minority classes).

Unbalanced data is a very real problem that draws growing interest [6] due to its prominence in several applications, such as fraud in financial transactions, spam detection, and weather prediction. Class imbalance occurs when a dataset does not have an equal number of exemplars from each class, which may be quite severe in some applications [7]. Common approaches to address class imbalance typically include oversampling or undersampling. The former involves increasing the number of minority class instances by creating copies of existing ones, which maybe prone to overfitting; whereas the latter involves removing majority class samples at random, which may cause loss of important information about the majority class. Perhaps one of the most unique, popular and unarguably successful approaches for class imbalance is the SMOTE algorithm [8], which modifies the feature space by creating a set of synthetic samples that lie on the line segment connecting two existing minority examples. The SMOTE algorithm was combined with the AdaBoost.M2 algorithm in [9] which allows for an ensemble approach to be combined with SMOTE.

The new member of the Learn^{++} family of algorithms described in this paper is the Learn^{++}.UDNC as it borrows techniques from both Learn^{++}.UD and Learn^{++}.NC. This version of Learn^{++} combines preliminary confidence measures in Learn^{++}.NC, with a transfer function that adjusts the voting weights of classifiers based on the number of instances seen from each class, as well as the class imbalance in each dataset. This approach works well in situations where an incremental learning is required to classify data coming from moderately imbalanced data distributions and new classes are being added and / or removed incrementally. It also works well in situations where classes are being added and removed at the same time from a database.

2 Learn^{++}.UDNC

The Learn^{++}.UDNC algorithm, whose pseudocode is shown in Fig. 1, receives subsequent dataset $\mathfrak{D}^k$ with m^k training examples, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m^k}\}$, and class labels $Y = \{y_1, y_2, \ldots, y_{m^k}\}$, a BaseClassifier to be used in ensemble generation, and T_k the number of classifiers to be generated from the kth dataset $\mathfrak{D}^k$. The algorithm works incrementally, hence never uses instances from previously seen datasets. Similar to that of AdaBoost [10], a distribution D is maintained according to which instances are drawn to train each classifier. Unlike AdaBoost, however, this distribution is updated based on the performance of the current ensemble, and not that of the previous classifier. The ensemble decision itself, on the other hand, uses a preliminary confidence measure and adjusts this confidence measures by the cardinality of each class in the training set.

Algorithm: Learn^{++}.UDNC

Input: Training dataset $\mathfrak{D}^k$, $k = 1,2, \ldots, K$
Sequence of input patterns $\{\mathbf{x}_i \in \mathbf{X}; y_i \in Y\}$ for $i = 1,2, \ldots, m^t$
Supervised learning algorithm: **BaseClassifier**
$N_{k,c}$, the number of class-c instances in $\mathfrak{D}^k$
Integer T_k, specifying the number of **BaseClassifiers** to create at $\mathfrak{D}^k$

Do for $k = 1,2,3 \ldots, K$
Initialize $D_1(i) = 1/m^1 \; \forall i, \; eT_k = \sum_{j=1}^{k-1} T_j$
If $k>1$,
- Update D_t and number of classifiers in the ensemble
- Update $w_{t,c} = w_{t,c} \frac{\sum_{j=1}^{k-1} N_{j,c}}{\sum_{h=1}^{k} N_{h,c}}, \quad t = 1,2, \ldots, eT_k, \; c = 1,2, \ldots, C$
- Start the sub-ensemble generation from Step 5 of the following loop.

Do for $t = 1, 2, \ldots$
1. Set $D_t(i) = D_t(i) / \sum_{j=1}^{m_k} D_t(j)$
2. Call **BaseClassifier**, providing $\mathfrak{D}_t^k \in \mathfrak{D}^k$ drawn from D_t
3. Obtain a hypothesis $h_t: \mathbf{X} \to Y$ and calculate its error
$$\epsilon_t = \sum_{j=1}^{m_k} D_t(j) [\![h_t(\mathbf{x}_j) \neq y_j]\!]$$
If $\epsilon_t > 1/2$, discard h_t and go to step 2. Otherwise compute Normalized performance $p_t = 1 - 2\epsilon_t, 0 \leq p_t \leq 1$
4. Compute class specific weights
$$w_{t,c} = p_t \frac{n_c}{\sum_{j=1}^{k} N_{j,c}}$$
where n_c is the number of class-c instances in $\mathfrak{D}_t^k$
5. Call *EnsembleDecision* to obtain sub-ensemble composite hypothesis H_t
6. Compute Error on composite hypothesis $E_t = \sum_{j=1}^{m_k} D_t(j) [\![H_t(\mathbf{x}_j) \neq y_j]\!]$
7. Set $\beta_t = E_t/(1 - E_t)$ and update the instance weights
$$D_{t+1} = D_t \times \begin{cases} \beta_t, & if \; H_t(\mathbf{x}_j) \neq y_j \\ 1, & otherwise \end{cases}$$

endfor
endfor
Call *EnsembleDecision* to obtain final hypothesis H_{final}

Fig. 1. Learn^{++}.UDNC Algorithm

The algorithm uses two loops for training, one indexed on k for subsequent datasets, and the other on t for individual classifiers to be generated for each training dataset. In step 1 of the inner loop (where we drop the superscript k, when the meaning is unambiguous, to avoid notational clutter), D_t is normalized to ensure a proper distribution, from which a training data subset $\mathfrak{D}_t^k$ is drawn is step 2. A hypothesis h_t is obtained from BaseClassifier in step 3, whose error is computed with respect to the current data distribution D_t. h_t is discarded if this error is greater than ½ and a new subset is drawn from D_t to create a new classifier.

Each classifier receives class specific weights, $w_{t,c}$, computed in step 4, based on the performance of h_t, the number of instances h_t is trained on for each class and the number of instances observed from a specific class. This method of computing the classifier weights limits the values of the classifier weights between 0 and 1, if sampling without replacement from D_t is used. Other weighting methods similar to those

used in the original Learn^{++} (including that of AdaBoost) follow a $\log(1/\beta_t)$ form, which are bound between 0 and infinity. While optimal for learning from a single dataset, $\log(1/\beta_t)$ assigns weights to classifiers based on their performance on the entire dataset, and not on the ability of a classifier to predict a specific class. The proposed weighting approach, described in detail below, gives more weight to classifiers that have more experience with a specific class. This voting scheme also reduces the outvoting problem that occurs when classifiers not trained on a specific class vote – inevitably incorrectly – for instances that come from that class.

The *EnsembleDecision* function is called in step 5 to obtain the composite hypothesis H_t of the current subensemble. The error of the subensemble on $\mathfrak{D}^k$ is computed in step 6 and is used to update the instance weights, D_{t+1} (step 7), emphasizing the instances misclassified by the subensemble. This process is repeated until the subensemble has T_k classifiers, bring the total number of classifiers to eT_k. At any time, the current overall ensemble decision (final hypothesis on all datasets) can also be obtained by calling the *EnsembleDecision*.

When called within the inner loop, *EnsembleDecision* combines all classifiers generated for a subensemble and evaluates the subensemble performance using the dynamically weighted voting mechanism in conjunction with a reweighting transfer function that takes class imbalance into consideration. When called for the final hypothesis, *EnsembleDecision* combines all classifiers generated thus far.

Algorithm: *EnsembleDecision*

Input: Set of instances to be classified $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_n}]$
Hypotheses h_t trained for $\mathfrak{D}^k$ and hypothesis weight matrix $\mathrm{w_{t,c}}$
$N_{k,c}$, the number of class-c instances in $\mathfrak{D}^k$
Integer T_k, specifying the number of **BaseClassifiers** to create at $\mathfrak{D}^k$

Initialization: Calculate $eT_k = \sum_{j=1}^{k-1} T_j$
Compute sum of class specific weights for *k*th subensemble

$$Z_{k,c} = \sum_{j=eT_k}^{eT_k+T_k} w_{j,c}$$

Do for $i = 1, 2, \dots, n$

1. Obtain preliminary confidences

$$P_{k,c} = \sum_{t:h_t(\mathbf{x}_i)=c} w_{t,c} / Z_{k,c}$$

$$t = eT_k, \dots, eT_k + T_k,\ k = 1,2, \dots, K,\ c = 1,2, \dots, C$$

2. Apply transfer function

$$\bar{P}_{k,c} = P_{k,c}^{\lambda_{k,c}},\ \lambda_{k,c} = \frac{N_{k,c}}{\min_c N_{k,c}}$$

3. Update sub-ensemble confidence

$$\hat{P}_{k,c} = \bar{P}_{k,c} \frac{N_{k,c}}{\sum_{j=1}^{K} N_{j,c}}$$

4. Compute ensemble decision

$$H_{final} = \arg\max_c \sum_{j=1}^{K} \hat{P}_{j,c}$$

endfor

Fig. 2. Ensemble Decision

The pseudocode for *EnsembleDecision* is shown Fig. 2. The inputs are a set of instances $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ to be classified, the hypotheses in the current subensemble (h_t) trained on $\mathfrak{D}^k$, the hypothesis weight matrix ($w_{t,c}$), and the number of instances, $N_{k,c}$, from each class in $\mathfrak{D}^k$. Initialization involves computing a normalization constant $Z_{k,c}$ as the total sum of class-specific weights for each class c seen by classifiers trained on $\mathfrak{D}^k$. A preliminary confidence $P_{k,c}$ is computed in step 1 for each class on which the classifiers in the kth subensemble were trained. $P_{k,c}$ represents the confidence of the kth subensemble in classifying class-c instances. This confidence is bound to be biased towards those classes with many instances. A transfer function is then applied in step 2 to reduce this bias on classes where training data is abundantly available. Step 3 updates the subensemble confidences $\bar{P}_{k,c}$ according to the cardinality of each class on which the subensemble was trained, relative to cardinality of the classes that all subensembles have seen. Finally, the ensemble confidence on a specific class is the sum of the subensembles confidences for that class, which form $\hat{P}_{k,c}$. The ensemble decision for $\mathbf{x}_i$ becomes the class corresponding to the largest sum of subensemble confidences.

3 Experimental Results

The Learn^{++}.UDNC algorithm was tested on two different datasets, each with several scenarios. The first is a synthetic Gaussian data, which allows us to compute Bayes performances, and the second one uses a real-world 10-class dataset available from [11]. The class specific performances (recall) as well as overall performances of Learn^{++}.UDNC are compared to those of fuzzy ARTMAP, an incremental learning algorithm capable of learning new classes, and SMOTE, which can learn from unbalanced data.

3.1 Gaussian Data

The first set of experiments involves 2D Gaussian data with four classes. The means for the data sets were $\mu_1 = (0,0)^t$, $\mu_2 = (1,1)^t$, $\mu_3 = (-1,0)^t$ and $\mu_4 = (1,-1)$, where the subscript refers to the class label. All covariance matrices assume that the features are uncorrelated with variances $\sigma_2 = \sigma_3 = \sigma_4 = 0.35$ and $\sigma_1 = 0.15$. Fig. 3 shows the decision boundary of the Bayes classifier on this database. The BaseClassifier used in the Learn^{++}.UDNC was an MLP with 20 hidden nodes on a single layer trained with an error goal of 0.05 and logistic sigmoid activation functions. Fifteen classifiers were created for each database that was introduced to the algorithm. Table 1 contains the class distributions of two Gaussian experiments, indicating the introduction of new classes in subsequent datasets, as well as the class imbalance at a ratio of 1 to 50. Also, note that the TEST data includes instances from all classes. In the first experiment, a new class, ω_3, becomes a second minority class, whereas the second experiment completely removes a class from the current training set. The SMOTE algorithm was applied to both ω_1 and ω_3 for $\mathfrak{D}^4$ in experiment 1 since ω_3 became a minority class along with ω_1.

Table 1. Data Distribution for Experiments with Synthetic Gaussian Data

	Experiment 1				Experiment 2			
Class →	ω_1	ω_2	ω_3	ω_4	ω_1	ω_2	ω_3	ω_4
$\mathfrak{D}^1$	10	0	0	500	10	0	0	500
$\mathfrak{D}^2$	10	500	0	500	10	500	0	500
$\mathfrak{D}^3$	10	500	500	500	10	500	500	500
$\mathfrak{D}^4$	10	500	10	500	10	500	0	500
Test	200	200	200	200	200	200	200	200

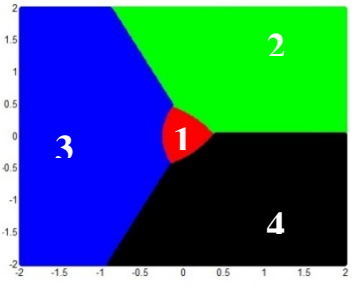


Fig. 3. Bayes Decision Boundary

The results from these two experiments are shown in Table 2 and Table 3. We observe that the Learn^{++}.UDNC outperforms the ARTMAP on all of the minority class examples (recall), as well as overall performance in both experiments. Also, Learn^{++}.UDNC performance on recall is comparable to that of SMOTE (with no significant difference between the two), but since Learn^{++}.UDNC is an incremental learning algorithm it can recall ω_3 better on $\mathfrak{D}^4$ when ω_3 is drastically reduced from the training set. On the other hand, SMOTE does a better job in recalling ω_1 for which the imbalance becomes progressively more severe (1 to 50 in $\mathfrak{D}^1$ and 1 to 100 in $\mathfrak{D}^4$). Even though ω_3 is minority in $\mathfrak{D}_4$, Learn^{++}.UDNC is able to recall its ω_3 knowledge from $\mathfrak{D}^3$, whereas SMOTE does not have this opportunity. Hence Learn^{++}.UDNC will have seen more of the data space to accurately predict on ω_3 while it may take additional data to learn the minority class (ω_1). SMOTE, on the other hand, can handle the more severe imbalance in any single setting.

It is perhaps a bit unfair to compare an ensemble based incremental learning algorithm to a single classifier based SMOTE, not designed for incremental learning. Hence we do not comment on the substantially better overall performance of Learn^{++}.UDNC over SMOTE, which is expected and not surprising; but rather we only comment on recall performance on the minority class, where the performances are more comparable, particularly for moderately unbalanced datasets. Furthermore, we merely point out that Learn^{++}.UDNC is capable of addressing unbalanced data (at least at a ratio of 1 to 50), while adding the capability of incremental learning of new classes, which themselves may be in minority.

Table 2. Gassian Expierment 1 Results on TEST dataset

		ω_1	ω_2	ω_3	ω_4	Overall
ARTMAP	$\mathfrak{D}^1$	41.60±16.53	0	0	99.55±0.20	35.29±4.11%
	$\mathfrak{D}^2$	41.75±12.86	87.60±6.21	0	96.00±2.05	56.45±3.50%
	$\mathfrak{D}^3$	14.85±6.12	90.70±2.07	92.50±1.22	96.30±1.11	73.59±1.53%
	$\mathfrak{D}^4$	18.95±7.04	91.35±0.83	85.90±2.74	95.65±1.32	72.96±1.62%
SMOTE	$\mathfrak{D}^1$	92.10±3.00	0	0	97.90±1.20	47.50±0.71%
	$\mathfrak{D}^2$	78.95±5.68	94.95±0.78	0	96.45±1.09	67.59±1.39%
	$\mathfrak{D}^3$	60.00±5.62	94.44±1.09	91.70±2.71	95.90±1.59	85.50±1.26%
	$\mathfrak{D}^4$	69.35±6.43	94.45±0.76	78.25±7.28	96.15±1.55	84.55±1.87%
Learn^{++}.UDNC	$\mathfrak{D}^1$	91.55±2.25	0	0	98.40±0.55	47.49±0.50%
	$\mathfrak{D}^2$	78.65±3.53	91.55±1.48	0	98.90±0.33	67.28±0.96%
	$\mathfrak{D}^3$	59.15±4.33	93.70±0.93	90.30±1.63	98.60±0.37	85.44±1.00%
	$\mathfrak{D}^4$	56.75±4.56	94.75±0.57	90.35±1.61	98.75±0.35	85.15±1.07%
Bayes						93.1%

Table 3. Gassian Expierment 2 Results on TEST dataset

		ω_1	ω_2	ω_3	ω_4	Overall
ARTMAP	$\mathfrak{D}^1$	39.90±13.65	0	0	99.80±0.25	34.93±3.38%
	$\mathfrak{D}^2$	33.95±9.78	97.55±6.27	0	96.60±1.08	52.52±2.73%
	$\mathfrak{D}^3$	16.05±2.41	86.10±3.29	93.85±1.55	94.70±1.15	72.68±0.93%
	$\mathfrak{D}^4$	19.05±4.69	86.40±3.90	91.05±2.29	94.20±1.12	72.68±1.92%
SMOTE	$\mathfrak{D}^1$	92.30±3.11	0	0	97.10±0.84	47.35±0.60%
	$\mathfrak{D}^2$	78.75±4.81	96.10±0.98	0	95.35±0.79	67.55±1.23%
	$\mathfrak{D}^3$	67.50±5.15	95.60±0.73	90.65±1.83	94.30±0.42	87.01±0.99%
	$\mathfrak{D}^4$	79.15±5.17	95.85±1.39	0	94.05±2.95	67.26±1.37%
Learn^{++}.UDNC	$\mathfrak{D}^1$	91.45±2.90	0	0	95.35±1.12	46.70±0.49%
	$\mathfrak{D}^2$	79.05±3.85	89.70±1.65	0	96.20±0.76	66.24±0.87%
	$\mathfrak{D}^3$	52.80±4.14	92.10±1.14	91.40±1.13	95.65±0.51	82.99±1.08%
	$\mathfrak{D}^4$	50.30±3.86	93.25±1.06	91.15±1.14	95.75±0.51	82.61±1.01%
Bayes						93.1%

3.2 OCR Data

The Optical Character Recognition (OCR) database consists of numerical characters 0~9 in a 8x8 matrix with two features removed due to zero variance of these feature. The training and testing class distributions are shown in Table 4, which has multiple new classes being introduced and removed at the same time in subsequent datasets. In addition to four datasets being introduced incrementally, Table 4 also shows a dataset $\mathfrak{D}^*$ which replaced $\mathfrak{D}^4$ in the OCR experiment. This new dataset, $\mathfrak{D}^*$, makes two of the previously seen classes minority class but includes instances from all classes. The results are shown in Table 5, which compares Learn^{++}.UDNC to ARTMAP.

Table 4. Database Distribution with Removal of Classes

Class →	0	1	2	3	4	5	6	7	8	9
$\mathfrak{D}^1$	0	257	248	0	0	248	248	0	0	252
$\mathfrak{D}^2$	0	0	247	257	0	0	247	252	0	0
$\mathfrak{D}^3$	248	0	0	256	0	0	0	252	248	0
$\mathfrak{D}^4$	247	256	0	0	252	247	0	0	247	252
$\mathfrak{D}^*$	20	20	250	250	250	250	250	250	250	250
Test	50	58	66	62	59	55	62	63	54	58

An MLP was used as the BaseClassifier with a 62x20x10 architecture with sigmoidal activation functions and an error goal of 0.05. Five classifiers were created with each dataset. The single classifier with SMOTE was not used with the OCR data because we wanted to see the effect of the class imbalance within a challenging incremental learning setting, where SMOTE – not designed to handle incremental data – would naturally be unsuccessful, and hence result in an unfair comparison. This is because Learn^{++}.UDNC will always be able to predict on previously seen classes that are not present in the current dataset, whereas SMOTE cannot predict on instances from classes not seen on the current training data.

Table 5. Results on the OCR Database

		0	1	2	3	4	5	6	7	8	9	All
ARTMAP	$\mathfrak{D}^1$	0	85.7	90.2	0	0	85.7	98.9	0	0	87.2	45.0±0.7%
	$\mathfrak{D}^2$	0	85.6	90.0	77.2	0	73.7	99.1	93.2	0	63.8	58.1±1.7%
	$\mathfrak{D}^3$	98.8	60.0	79.3	76.1	0	63.4	91.1	87.1	70.2	52.9	67.9±1.9%
	$\mathfrak{D}^4$	96.1	83.7	72.0	62.8	80.2	77.1	79.3	82.3	74.3	76.4	78.5±1.5%
	$\mathfrak{D}^5$	97.2	80.9	88.7	91.1	94.6	87.7	94.9	94.6	87.5	86.6	90.3±0.9%
Learn^{++}.UDNC	$\mathfrak{D}^1$	0	93.5	98.0	0	0	97.6	99.3	0	0	95.2	48.6±0.2%
	$\mathfrak{D}^2$	0	84.7	97.6	92.1	0	85.7	99.8	96.3	0	48.5	60.3±1.4%
	$\mathfrak{D}^3$	99.8	70.4	96.9	92.6	0	88.1	97.5	97.1	80.7	63.1	78.7±1.1%
	$\mathfrak{D}^4$	99.5	88.4	96.9	91.9	78.7	89.4	98.9	98.4	85.7	68.2	89.6±0.5%
	$\mathfrak{D}^5$	99.8	96.1	88.0	93.9	76.9	88.9	99.8	99.8	87.5	72.4	90.3±0.6%

Table 5 shows the results for the Learn^{++}.UDNC and ARTMAP applied to the incremental learning problem described in Table 4. A large performance boost is observed for Learn^{++}.UDNC as $\mathfrak{D}^4$ is introduced, resulting in fuzzy ARTMAP being outperformed by a large margin. The Learn^{++}.UDNC maintains the best recall of the minority classes when $\mathfrak{D}^*$ is introduced, however the overall performances here are no longer significantly different. Of the two minority classes, while the recall of character 0 for both algorithms are very close, the recall for character 1 with the Learn^{++}.UDNC is significantly better than that of ARTMAP.

An additional experiment was also created with the OCR data that has a single minority class (character 0), with 40 instances of this class presented with each dataset, otherwise using the same class introduction and removal shown in Table 4. Therefore,

the minority class is always present in the data and the incremental learning algorithm needs to continue to learn new parts of the minority and majority classes. ($\mathfrak{D}^*$ is not present in this new experiment).

Table 6. OCR Database Results with Single Minority Class

		0	1	2	3	4	5	6	7	8	9	All
ARTMAP	$\mathfrak{D}^1$	68.6	85.9	88.3	0	0	92.3	96.9	0	0	83.9	51.8±0.7%
	$\mathfrak{D}^2$	82.9	83.8	88.0	78.4	0	67.3	96.2	96.4	0	54.6	64.5±1.6%
	$\mathfrak{D}^3$	87.7	58.4	77.8	78.1	0	63.7	89.6	93.4	72.9	44.8	66.2±3.0%
	$\mathfrak{D}^4$	85.1	81.7	70.2	66.1	86.8	79.5	81.1	88.6	73.5	71.4	78.3±1.9%
Learn^{++}.UDNC	$\mathfrak{D}^1$	97.9	97.2	94.4	0	0	90.7	97.1	0	0	95.17	57.1±0.5%
	$\mathfrak{D}^2$	99.6	88.8	94.0	93.9	0	71.6	97.8	97.9	0	57.9	69.8±1.4%
	$\mathfrak{D}^3$	99.6	70.0	89.1	95.6	0	71.3	92.0	97.9	86.1	62.7	75.5±1.5%
	$\mathfrak{D}^4$	99.4	83.2	89.3	94.9	88.9	81.5	97.3	98.2	89.6	69.8	88.9±0.9%

Table 6 shows that the Learn^{++}.UDNC is able to consistently outperform fuzzy ARTMAP, not only on the minority class (0), but also on classes 2,3,6,7,8, with generally higher (but not statistically significant) performance on others.

4 Conclusions and Future Work

We described an incremental learning algorithm, Learn^{++}.UDNC, that combines several novelties of different algorithms within the Learn^{++} family, including a class specific weighting method, normalized preliminary confidence measures and a new transfer function that is used to reduce the confidence bias of a sub-ensemble trained on a majority class. Preliminary results indicate that Learn^{++}.UDNC is able to consistently outperform fuzzy ARTMAP under a variety of incremental learning scenarios and with a wide margin on unbalanced data problems. This was observed with both synthetic and real-world incremental learning problems. While not quite as effective as SMOTE on severely unbalanced data, we have shown that the Learn^{++}.UDNC performs comparably to SMOTE on minority class recall on moderately unbalanced datasets, but with the added advantage of learning incrementally, without applying any oversampling (or undersampling). This algorithm has shown the ability to perform well on a broad spectrum of incremental learning problems where the previous members of the Learn^{++} algorithms are not able to be reliable predictors on all classes. Current and future work include evaluating the algorithm on more severe unbalanced data on synthetic and real-world datasets, as well as integrating SMOTE and Learn^{++}.

Acknowledgements

This material is based on work supported by the National Science Foundation, under Grant No: ECCS-0926159.

References

1. Grossberg, S.: Nonlinear neural networks: principles, mechanisms, and architectures. Neural Networks 1, 17–61 (1988)
2. Muhlbaier, M., Topalis, A., Polikar, R.: Learn++.NC: Combining Ensembles of Classifiers with Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes. IEEE Transactions on Neural Networks 20(1), 152–168 (2009)
3. Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.: Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3, 698–713 (1992)
4. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: an ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
5. Muhlbaier, M., Topalis, A., Polikar, R.: Incremental learning from unbalanced data. In: Proc. of Int. Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary, July 2004, pp. 1057–1062 (2004)
6. Chawla, N., Japkowicz, N., Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter 6(1), 1–6 (2004)
7. Kubat, M., Holte, R.C., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30, 195–215 (1998)
8. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Oversampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
9. Chawla, N., Lazarevic, A., Hall, L., Bowyer, K.: SMOTEBoost: Improving Prediction of the Minority Class in Boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)
10. Freund, Y., Schapire, R.E.: Decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
11. Asuncion, A., Newman, D.J.: UCI Repository of Machine Learning (November 2009), http://www.ics.uci.edu/~mlearn/MLReporitory.html

Tomographic Considerations in Ensemble Bias/Variance Decomposition

David Windridge

CVSSP, University of Surrey, Guildford, UK

Abstract. Classifier decision fusion has been shown to act in a manner analogous to the back-projection of Radon transformations when individual classifier feature sets are non or partially overlapping. It is possible, via this analogy, to demonstrate that standard linear classifier fusion introduces a morphological bias into the decision space due to the implicit angular undersampling of the feature selection process. In standard image-based (eg medical) tomography, removal of this bias involves a filtration process, and an analogous n-dimensional processes can be shown to exist for decision fusion using Högbom deconvolution.

Countering the biasing process implicit in linear fusion, however, is the fact that back projection of Radon transformation (being additive) should act to reduce variance within the composite decision space. In principle, this additive variance-reduction should still apply to tomographically-filtered back-projection, unless the filtration process contravenes.

We therefore argue that when feature selection is carried-out independently for each classifier (as in e.g. multi-modal problems) unfiltered decision fusion, while in general being variance-decreasing, is typically also bias-increasing. By employing a shot noise model, we seek to quantify how far filtration acts to rectify this problem, such that feature selection can be made *both* bias and variance reducing within an ensemble fusion context.

1 Introduction

A central result of both the MCS and regression ensemble fields is that of the bias-variance-covariance decomposition of the mean squared error (MSE) [8,1]. Whereas in individual classifiers we are concerned only with a bias-variance trade-off (i.e. assessing flexibility verses structural risk), Ueda and Nakano [4] demonstrated that ensembles must also consider correlations between estimators either implicitly or explicitly. This can be related to the Tumer and Gosh [7] framework for describing fused classifier error in terms of the effect on the margin. Thus (adopting the nomenclature of Brown et al. [1]), we have that $f(X, y, params)$ defines an estimator of some true underlying function $t(X, y)$ defined over the feature space X w.r.t. to the class y. We denote the combination of M estimators as:

$$\bar{f} = \frac{1}{M} \Sigma_{i=1}^{M} f_i(X, y, params) \tag{1}$$

(Henceforth, we drop the parameter-denotation from f and t)

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 43–53, 2010.

Thus, we have that for an ensemble of M estimators, the estimated mean square error can be decomposed via eqn. 1 as follows:

$$E\{MSE\} = E\{(\bar{f} - t)^2\} = (E\{\bar{f}\} - t)^2 + E\{(\bar{f} - E\{\bar{f}\})^2\} \quad (2)$$

$$= bias(\bar{f})^2 + variance(\bar{f}). \quad (3)$$

$$= \bar{bias}^2 + \frac{1}{M}\bar{var} + \left(1 - \frac{1}{M}\right)\bar{covar} \quad (4)$$

where the bar dictates an ensemble average quantity, i.e.:

$$\bar{bias} = \frac{1}{M}\Sigma_i(E\{(f_i - t)\}) \ , \ \bar{var} = \frac{1}{M}\Sigma_i E\{(f_i - E\{f_i\})^2\} \quad (5)$$

$$\bar{covar} = \frac{1}{M}\Sigma_i\Sigma_{j\neq i}E\{(f_i - E\{f_i\})(f_j - E\{f_j\})\} \quad (6)$$

$E\{.\}$ is the expectation over all samples and all X. Error minimization hence requires that we seek to reduce the bias, variance and covariance of the constituent classifiers as far as possible. It is thus clear that various advantages accrue from using ensembles: we allow for the *possibility* for uncorrelated biases to cancel each other out and for the relative suppression of absolute deviations via the additivity of variance. It is also possible that, unlike the other terms, covariance can be negative, permitting a further avenue for error minimization in the ensemble.

In the current scenario, we consider only estimators f_i that include an (explicit or implicit[1]) feature selection stage for each classifier, such that the rejected features are treated by the omission of ordinates from the data vectors, i.e.:

$$\forall X^n, y, params \ f_i(X^n, y, params) \propto f_i(x^{n_i}, y, params)$$

with $n_i \subset n$ (specifically, x^{n_i} is a projective subset of X^n, a convention we shall adopt throughout). Classifiers are hence taken to model marginal distributions of $t(X^n, y)$ (though we will still consider the classifier to be defined over all X^n, as this will become important later). This consideration potentially complicates all 3 aspects (bias, variance and covariance) of the standard analysis, and reveals other strategies for reducing the overall MSE.

For example, ensemble covariance should tend towards zero when it is legitimate to make a naive Bayes assumption about the data irrespective of the underlying classifiers; that is, classifier diversity may be brought about by the feature selection process rather than the intrinsic nature of the classifiers in an ensemble. (Arguably the strongest motivation for feature selection in an MCS context is projection of largely independent data into independently-classified marginal distributions in order to maximize sampling (and thereby minimize structural risk) at no cost to the feature-space coverage; however, we here consider the more general case in which features may be associated to classifiers for

[1] In multi-modal decision fusion, we can consider the individual modalities as being a feature-selected subset of some composite space.

purely instrumental reasons). In rejecting the naive Bayes assumption as generally unrepresentative, it is evident that classifier variance can only be reduced by a factor related to the increase in bias within a feature-selection context (classification within marginal projections implies fewer parameters to represent the data). Feature-selected classier ensembles can thus not take advantage of any intrinsic decorrelation in bias in the same way as non-feature-selected ensembles.

Feature selection thus, in general, acts to reduce variance via the reduction in dimensionality, reduce *co*-variance via the introduction of the possibility of classifiers relating to potentially independent subspaces, but increases bias by eliminating information-bearing feature compositions (cf [3]).

The argument of this paper is that this issue is not necessarily as clear-cut as is usually considered: that we can, in fact, exploit feature selection to reduce variance and yet offset some the increase in bias by appropriate treatment (eg deconvolution) of the morphological sampling artifacts induced by the feature selection process. To do this we need to consider the tomographic nature of the feature-selection/combination process. Section 2 will therefore outline this analogy and its practical application. Section 3 will extend this work by quantifying a theoretical application of this approach in bias/variance terms and section 4 will apply an experimental test of the idea.

2 The Tomographic Analogy for Classifier Fusion

Full details of the tomographic approach to removing the morphological bias from decision fusion are given in [9]. Put briefly, the tomographic analogy assumes that the projective nature of feature selection process with respect to the original feature space, followed by the subsequent representation of marginals within generative (and to a lesser extent discriminative) classifiers can be modelled as n-dimensional Radon transformation (Radon transformation being the 'line-of-sight' integration carried out by eg X-ray detectors in medical imaging). Thus we assume:

$$f(\boldsymbol{x_i}, y)dx_i \approx \int_{x}^{\boldsymbol{x_i}+dx_i} t_i(x_i, y)dx_i \equiv \int_{x}^{\boldsymbol{x_i}+dx_i} \int_{all\ x'} t(X^n, y)dx'\ dx_i \qquad (7)$$

where $\boldsymbol{x_i} \in \boldsymbol{X}^{n_i}$ & $\boldsymbol{x'_i} \in \boldsymbol{X'}^{n_i}$ with $X^{n_i} \subset X$ and $X'^{n_i} = X^{n\perp}$ (i.e. X'^{n_i} is the orthogonal complement of X_i^n; $t_i(x_i, y, params)$ is thus the true marginal distribution.

If multiple classifiers are derived in this fashion (i.e. so that it is not necessarily the case that feature sets within the individual classifiers are fully coincident) then it can be shown that linear classifier combination (eg Sum Rule, Product Rule) is either equivalent to, or bounded by, back-projection, the inverse operation to Radon projection; $p_b(\boldsymbol{X^n}) = \frac{1}{M}\Sigma_{i=1}^{M} f_i(\boldsymbol{x_i}, y)$. However, this introduces an axially aligned artefact, $A(\boldsymbol{X^n}) = \Sigma_i dx_i . \int_{all} dX'^{n_i}$, that is a consequence of the fact that the Radon projections induced by feature selection represent only

a small fraction of the total angular sample-space required for lossless reconstruction of the function $t(\boldsymbol{X^n}, y)$. What *is* recovered by back-projection (i.e. linear classifier fusion) is an estimation of the function $t(\boldsymbol{X^n}, y) \star A(\boldsymbol{X^n})$ (ie the true function t convolved with the artefact A). What we actually require is an estimate of $t(\boldsymbol{X^n}, y)$. While, in general, it is impossible to recover all of the 'lost' information brought about by feature selection by deconvolving A from the back-projected (ie fused classifier) output, performing this deconvolution does give rise to a *morphologically unbiased* estimate of t.

Rather than explicitly perform this deconvolution, pre-filtration of the Radon integrals is generally used prior to back-projection in medical imaging. However, since this approach can gives rise to negative values unrepresentative of stochastic estimates, the present paper considers a post-filtration approach, via iterative Högbom deconvolution of the biasing artefact[2].

Högbom deconvolution consists in iterative removal of the biasing artefact and replacing it by a Dirac delta function (or a coarse approximation to it) in the composite decision space. This process can be shown [10] to be equivalent to seeking correlated morphology in the classifiers and progressively reconstructing the morphology giving rise to this correlation in the composite decision space. It thus outperforms linear combination methods by using the correlated morphologies of classifiers in the ensemble to give more information about the sampled point than would otherwise be available. (Note that this approach works even for discriminative classifiers, though is optimal for generative classifiers). As a pseudo-code, the Högbom methodology is as set out in Appendix 1.

3 Theoretical Study: Morphologically Induced Bias Following Variance-Motivated Feature-Selection

For the present study, we assume a generative model of classification, in which classifiers (even if feature-selected) estimate the overall class distribution. If we were further to assume a unimodal model in which classes are represented by an arbitrary single-peaked distribution then the Högbom algorithm is provably optimal (ie can potentially recover the entire composite distribution $t(\boldsymbol{X^n}, y)$) from the marginal classifiers, provided the unimodality is of known cross-sectional form. This covers a wide range of possibilities included Gaussians with arbitrary covariance matrices. Under more realistic conditions (ie with an unknown cross-section), the Högbom algorithm is generally sub-optimal, but well-behaved, making only conservative (ie non-biasing) assumptions about the ambiguities arising from deconvolution, such that unimodal distributions of $t(\boldsymbol{X^n}, y)$ will give rise to identically unimodal estimates $t_{est}(\boldsymbol{X^n}, y)$ for all possible Radon projections and

[2] Note that Högbom deconvolution is not necessarily an approach that would be economic in practise; we here consider it because of its guaranteed positivity preserving characteristics. Note that efficient implementations of post-filtration *are* possible by appropriate kernelisation of the method (though beyond the scope of the current paper to set-out).

back-projections of X^n. The same is not generally true of linear fusion methods; an arbitrary change of basis of $t_{est}(\boldsymbol{X^n}, y)$ can potentially introduce differing numbers of modes within transformed marginal distributions.

In order to estimate the effect that this axial bias has on a standard feature-selection/classifier-fusion approach, we consider instead a simplified shot noise distribution model within X^n, such that n individual classifiers consist of non-intersecting unidimensional marginal distribution estimates (i.e. one feature is allocated for each of the n classifiers). The shot noise model considered consists of a random placement of K_{tot} distribution centroids such that, within each feature i, the marginal projection of each one of the K individual distributions has a well-defined width of ω_i (such that the marginal density projection has a value of exactly zero elsewhere). This occurs within a bounded width Δ_i attributable to the feature as a whole.

The marginal distribution estimate is the integral over the remaining $n-1$ components (assuming 1 selected feature per classifier) of the K distributions, which thus have density distributions D in X^n and marginal distributions D_i in X_i^n, i.e.:

$$P_i(x_i) = \int_{\forall \boldsymbol{x_j : j \neq i}} \Sigma_{K=1}^{K_{tot}} D(\boldsymbol{x} - \boldsymbol{c^K})\, d\boldsymbol{x} \equiv \Sigma_{K_i=1}^{K_{tot}} D_i(x_i - c_i^K) \tag{8}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots x_n)$ and $\boldsymbol{c^K} = (c_1^K, c_2^K, \ldots c_n^K)$ is the Kth cluster center.

For the sake of the current analysis, we initially assume that marginal distributions are sufficiently densely-sampled for there to be no significant variance issues when classifiers C_i are assigned to each feature, with undersampling only evident in the composite space $\boldsymbol{x}$ (perhaps motivating the feature selection in the first place). That is, we wish to isolate the tomographic influence on bias at this stage.

Since $D_i(x_i - c_i^K)$ is only non-zero for $x_i = c_i^K \pm \omega_i/2$, we can write:

$$P_i(x_i) = \Sigma_{k(x_i)} D_i(x_i - c_i^k) \tag{9}$$

where $k(x_i) \leq K_{tot}$ indexes the set of cluster centers for which $c_i^K = x_i \pm \omega_i/2$ (ie the clusters that become 'merged' under marginal projection).

The Högbom algorithm iteratively identifies and removes either whole or partial $D(\boldsymbol{x} - \boldsymbol{c^K})$ components from the back-projected (sum rule) composite space by recursively selecting the peaks in the density functions of classifiers defined over each marginal distribution. In general, the peaks of each marginal distribution estimate will be defined by the peaks of $k(x_i)$ which is, in turn, determined solely by the distribution of shot noise in the model (this is always true if $D_i(x_i - c_i^k) = const$ for $x_i = (c_i^K \pm \omega_i/2)$. This means that $P_i(x_i) \approx P(k(x_i)) = {}^{K_{tot}}C_k p^k (1-p)^{K_{tot}-k}$ (ie k is Binomially distributed, with $p = \omega_i/\Delta_i$.)

In the recursive Högbom deconvolution, all marginal distribution components of D at 'density level' $k/(K_{tot}\omega_i) < h < (k+1)/(K_{tot}\omega_i)$ are removed at the k-th iteration (these components are the level sets parameterized by h, i.e. the closed topological sets created by the truncation of the marginal density at value h).

The original shot noise components to which these level sets refer cannot be disambiguated by the procedure if the cardinality of closed topologies is greater than 1 for more than one of the features, and the reconstituted space must consist in all possible compositions of components ie $\{\{c_1^{k_h}\} \times \{c_2^{k_h}\} \times \ldots\}$, with k_h the number of marginal components for which $h \leq k.const$. The cluster centers generated in the composite space are thus the set of *all* ordered n-tuples:

$\{(a_1, a_2, \ldots a_I) | a_1 \in \{c_1^{k_h}\}, a_2 \in \{c_2^{k_h}\} \ldots a_I \in \{c_2^{k_h}\}\}$) .

From the Binomial distribution of marginal components, we have that the mean density level of the marginal distributions is $K_{tot} \cdot \frac{\omega_i}{\Delta_i} \cdot \frac{1}{K_{tot}}$, meaning that there will be $\approx \left(\frac{\Delta_i}{\omega_i}\right)^n$ cluster centers in the reconstituted space following Högbom deconvolution if K_{tot} is large. Under sparse conditions, however, this figure will be $(K_{tot})^n$. All but K_{tot} of these reconstituted cluster centers are excess with respect to the true distribution of t in X^n; however we are guaranteed that these K_{tot} cluster centers are accurately represented if the marginal density estimate is accurate. These excess cluster centers (of cardinality $\approx ((K_{tot})^n - K_{tot})$) constitute the main source of remaining bias in the tomographic fusion model when applied to the shot noise model, representing the irrecoverable information loss implicit in feature selection.

Without Högbom deconvolution (ie using standard linear decision fusion), cluster centers are not explicitly identified in the composite space X^n. However, each point of the back-projected composite space *does* contain a contribution from the correct center. In fact, the unfiltered back-projected space consists of Dirac delta functions located at the correct centers (ie $\delta(c_1^1, c_2^1), \delta(c_1^2, c_2^2) \ldots \delta(c_1^K, c_2^K)$) that are convolved with the axially-aligned-artefact $A(X^n)$, such that the reconstituted classifier density generated by standard linear classifier combination is $\Sigma_K D(\boldsymbol{x} - \boldsymbol{c^K})) \star A(X^n)$. However, the interstices of the convolved artefacts themselves produce further Dirac delta functions (eg $\delta(c_1^1, c_2^2), \delta(c_1^2, c_2^1), etc$), that are equivalent to the novel cluster centers produced by the Högbom algorithm. Thus, filtered and unfiltered decision fusion are identical in terms of the generation of spurious cluster centers within the decision space under a shot noise model. This represents the unavoidable bias in decision fusion. However, in the absence of Högbom filtration, there is also the additional ambiguity created by the convolution artefacts. This represents the excess bias created by standard linear combiners.

Thus, to quantify these biases, we have that the excess bias generated by linear combination followed by Högbom filtration is:

$$\text{Bias}_{Tom} \approx \int_{\forall \boldsymbol{x_j}: \boldsymbol{j} \neq \boldsymbol{i}} \Sigma_{K=1}^{K_{tot}} D(\boldsymbol{x} - \boldsymbol{c^K}) \, d\boldsymbol{x} \, . \, \frac{1}{K_{tot}} \left((K_{tot})^n - K_{tot}\right)^2 \quad (10)$$

$$\approx \left[\frac{1}{K_{tot}} \left((K_{tot})^n - K_{tot}\right)^2\right] \quad (11)$$

The corresponding quantity for the sum rule decision scheme (representing linear fusion), which includes the axially-aligned artefacts generated by the convolution of reconstructed cluster centers with A, is the following:

$$\text{Bias}_{Sum}=\sum_{i=1}^{n}\frac{1}{n}\int_{\forall \boldsymbol{x_{j:j\neq i}}} d\boldsymbol{x'_i}.\int_{\boldsymbol{x_i}}\sum_{K=1}^{K_{tot}} D_i(x_i-c_i^K)^2\,d\boldsymbol{x_i}-\int_{\forall X^n} D(\boldsymbol{x}-\boldsymbol{c^K})^2\,d\boldsymbol{x} \quad (12)$$

$$\approx \Sigma_{i=1}^{n}\left(\frac{1}{nK_{tot}}\left[K_{tot}\left(\Pi_{i=1}^{n}\frac{\Delta_i}{\omega_i}\right)-K_{tot}\right]\right)^2 \quad (13)$$

(weighted sums, while not directly considered, should behave similarly)

Since the condition of sparsity is that $\frac{\Delta_i}{\omega_i} >> K_{tot}$ this implies that linear decision schemes will *always* have higher bias than the tomographically-filtered equivalent for sparse distributions for the case of single features per classifier with distribution centers as per the shot noise model. The problem worsens with both increasing dimensionality and increasing sparsity.

3.1 Variance Estimation

Variance is imported into this simplified model by considering sampling variation with respect to the marginal histograms standing in for classifiers. Variance within single-feature classifiers is hence reduced by a factor relating to the marginal integration; we denote this post-feature-selected marginal variance: v_i. However, in addition to this variance reduction mechanism, the back projection implicit in feature selection has the potential to reduce this variance further via summation. Specifically, in the composite decision space, we expect this further reduction at the points of convergence of the non-zero marginal histograms due to the combination. For the (normalized) sum rule, as for the Högbom-filtered decision space, this implies a variance of $\frac{1}{n^2}\Sigma_{i=1}^{n} v_i$ at the cluster centers (on the assumption of decorrelation). However, elsewhere the non-zero component caused by the Radon artefacts in the sum rule will not experience this reduction; variance for the axial component will be as for the marginal distributions (i.e. v_i). In general, these will dominate for a sparse distribution, giving a total variance of:

$$\text{Variance}_{Sum} \approx \Sigma_i(\tfrac{\Delta_i}{\omega_i} - K_{tot})\omega_i.v_i + K_{tot}^n\tfrac{1}{n^2}\Sigma^n v_i\omega_i$$

The corresponding variance for the filtered combination is simply:

$$\text{Variance}_{Tom} \approx K_{tot}^n\tfrac{1}{n^2}\Sigma^n v_i\omega_i$$

However, this analysis does not consider the effect of Högbom deconvolution *within* cluster centers, where the recursive identification of morphology may introduce other sources of variance.

4 Experimental Investigation at the Sparse/Dense Boundary

In order to quantify these effects further, we perform an experimental implementation using the shot noise model. In particular, we wish to evaluate more typically borderline cases, in which the sparseness of distribution centroids is reduced to the point of dense overlap.

To do this, we distribute randomly-parameterized Gaussian distributions within the composite (i.e. non-feature selected) pattern-space, and form uni-dimensional marginal histograms to act as classifiers of the overall distribution. We hence consider a two class case, in which each class consists of a density function so specified. The random distributions are obtained according to the standard multivariate Gaussian distribution,

$$f(X) = \sum_e \frac{|covmat_e|^{-\frac{1}{2}}}{2\pi^{d/2}} . A_e . e^{-\frac{1}{2}(X-M_e)^T covmat_e^{-1}(X-M_e)} \tag{14}$$

d is the dimensionality of the problem (in this case 2); $P(A) = const$ for $A \in [0, 1]$

The covariance matrix *covmat* is derived via its Eigendecomposition; i.e. $covmat = U \Lambda U^T$, such that U is considered a rotation matrix over arbitrarily chosen θ thus:

$P(\theta) = const$ for $0 < \theta < 360$, $P(\theta) = 0$ otherwise.

$$U = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \Lambda = \begin{pmatrix} R_x & 0 \\ 0 & R_y \end{pmatrix}$$

$P(R_x), P(R_y) = const$ for $0 < R_x, R_y < 1$, $P(R_x), P(R_y) = 0$ otherwise.

Sampling of this distribution is achieved via Cholesky factorization of *covmat* and multiplication with randomly sampled vector uniform distribution over the domain bounded by the Δ_i's. As a proxy for sparseness variation, we keep the number of marginal histogram bins fixed, and range over Gaussian number, width and sampling parameters (we choose $\sigma^{-1} = [1 : 10] * const$, Gauss $n^o = [1 : 5]$ and max sample $n^o = 9375$ so as to nominally straddle the sparse/dense border at $\sigma^{-1} \approx 6$, when σ is of the order of the histogram bin width). Bias and variance are then evaluated with respect to the two fusion methodologies; the density-normalised Sum Rule and Högbom filtered Sum Rule (since we evaluate bias and variance with respect to the final fused classifiers in the composite space, it is not appropriate to consider intra-ensemble covariance). We also consider a histogram binning classifier in the original space with identical bin-width characteristics to the marginal histograms. Two separate distributions are created for each sampling of the parameters and designated as class 1 and 2. A misclassification rate is also calculated. Results are as depicted in figures 1-3.

5 Discussion and Conclusions

We find that, under the test conditions of borderline sparse/dense shot-noise distribution, the Högbom method retains its low bias but develops a significantly higher variance than the Sum rule despite backprojection. However, this does not appear to affect Bayes error rate adversely. Hence the "boundary bias" [2] (i.e. $bias(f, E(\bar{f})) = \text{sign}(1/2 - f)(E(\bar{f}) - 1/2)$) that typically favors *generalized* low variance over low bias in terms of the Bayes error rate does not apply in this case. This would suggest that the Högbom method experiences low bias and low variance at the most classification-critical regions.

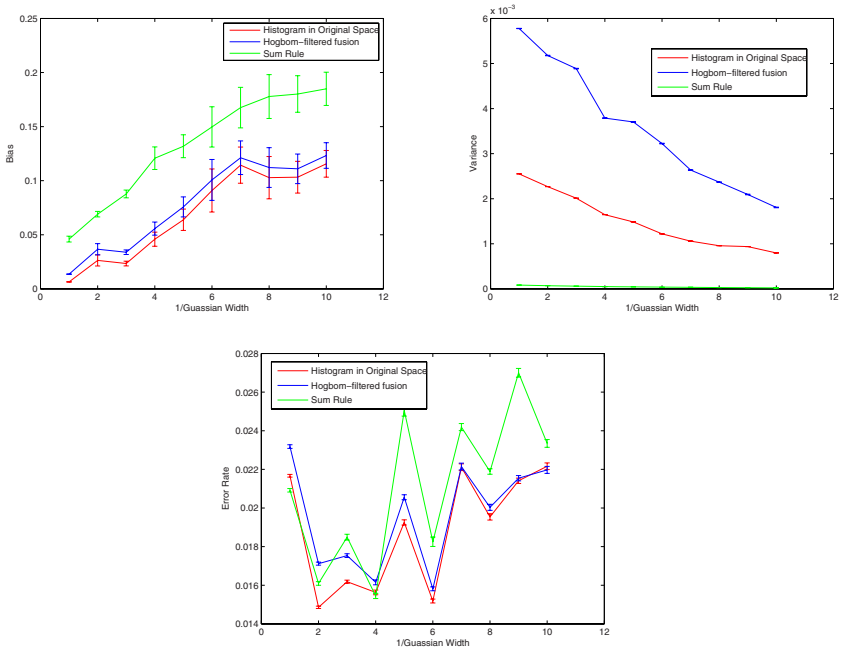


Fig. 1. Bias, Variance and Error Rate Per Histogram Bin vs Sparsity

In conclusion, we have shown theoretically that under sparse conditions, filtered classifier fusion can decrease both bias and variance. In experimental conditions with more marginal distributions, this decreased bias is retained only at the expense of increased variance with respect to the linear decision rules. This would appear to be a side effect of seeking morphological correlation within the classifiers. However this does not appear to adversely effect misclassification rate.

Should this increased variance prove to be problematic in more general scenarios, bootstrap re-sampling (i.e. bagging) should mitigate the effect. In this way we can simultaneously reduce ensemble bias and variance (cf eg [6], [5]).

Acknowledgement. The research leading to these results has received funding from the EC 7^{th} Framework Programme FP7/2007-2013 under grant agreement n^o 215078.

References

1. Brown, G., Wyatt, J.L., Tiňo, P.: Managing diversity in regression ensembles. J. Mach. Learn. Res. 6, 1621–1650 (2005)
2. Friedman, J.H.: On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Min. Knowl. Discov. 1(1), 55–77 (1997)
3. Munson, M.A., Caruana, R.: On feature selection, bias-variance, and bagging. In: ECML/PKDD, vol. (2), pp. 144–159 (2009)

4. Uedar, N., Nakano, R.: Generalization error of ensemble estimators. In: Proceedings of International Conference on Neural Networks, pp. 90–95 (1996)
5. Smith, R.S., Windeatt, T.: The bias variance trade-off in bootstrapped error correcting output code ensembles. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 1–10. Springer, Heidelberg (2009)
6. Suen, Y.L., Melville, P., Mooney, R.J.: Combining bias and variance reduction techniques for regression trees. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 741–749. Springer, Heidelberg (2005)
7. Tumer, K., Ghosh, J.: Theoretical foundations of linear and order statistics combiners for neuralpattern classifiers. Technical Report TR-95-02-98, Computer and Vision Research Center, University of Texas, Austin (1995)
8. Valentini, G., Dietterich, T.G.: Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. J. Mach. Learn. Res. 5, 725–775 (2004)
9. Windridge, D., Kittler, J.: A morphologically optimal strategy for classifier combination: Multiple expert fusion as a tomographic process. IEEE Trans. Pattern Anal. Mach. Intell. 25(3), 343–353 (2003)
10. Windridge, D., Kittler, J.: Performance measures of the tomographic classifier fusion methodology. Intern. Jrnl. of Pattern Recognition and Artificial Intelligence 19(6) (2005)

Appendix 1: Procedural Implementing of Post-filtered Tomographic Classifier Combination

1. Assemble the combiners as a series of estimators ranging over n discrete feature spaces of respective dimensionality; $a_1, a_2 \ldots a_n$ for the class set; $\omega_1, \omega_2, \ldots \omega_m$; label these $P_n(\boldsymbol{X}_n)$, where $\boldsymbol{X}_n$ ranges over the vector space of dimensionality a_n.

2. Select the first class of the series, ω_1, and establish peak probability density value(s), $P_n^{\max}$, for of each expert's individual representation of that class.

3. Specify a pair of accuracy parameters, Δz and Δx, that respectively denote the probability density and feature-space resolutions.

4. Establish the 'hyper-area' between the probability density ordinates representing the peak value and (peak value$-\Delta z$) for each of the classifier PDFs: ie, the *scalar* number of $(\Delta x)^{a_i} \times \Delta z$ units between the two probability density values for each of the classifiers in the fusion. Vectors within these bounds are designated $\boldsymbol{X}'_n$.

5. Specify a matrix of dimension; $a_1 + a_2 + \ldots + a_n$ with each element designating an (initially zero) probability density value attributable to every $(\Delta x)^{a_1+a_2+\ldots+a_n}$ unit of the composite feature-space. Add a value, N, to those points representing all combinations of n concatenations of the respective (co-)ordinates established in **4**: That is, the Cartesian product $\{\boldsymbol{X}'_1\} \times \{\boldsymbol{X}'_2\} \times \{\boldsymbol{X}'_3\} \times \ldots \times \{\boldsymbol{X}'_n\}$. ($N$ must be $> \sum_{i=1}^{n} P_n^{\max}$).

6. Subtract the resolution parameter Δz from each peak value $P_n^{\max}$; $\forall i$, and set an iteration parameter (say, t) to zero.

7. Subtract a quantity $|X'_1| \times |X'_2| ... \times |X'_{i-1}| \times |X'_{i+1}| \times ... \times |X'_n| \times dz$ from the *current* peak value of each classifier, $P_n^{\max}$; $|X'_j|$ being the scalar values derived in **5**, ie: the number of coordinate vectors $\{\boldsymbol{X}'_i\}$ of dimensionality a_i counted by the PDF hyper-area establishing procedure above. Note, especially, the absence of $|X'_i|$ in the product entity.

8. Establish the *new* hyper-area value associated with subtraction **7**, ie: the hyper-area between the probability density ordinates representing the previous and current peak-values (as per **4**).

9. Allocate a value $N - t.\Delta z$ to those points in the deconvolution matrix representing *novel* coordinates established after the manner of **4.** That is, the Cartesian product *difference*:
$[(\{\boldsymbol{X}'_1\}_{old} \cup \{\boldsymbol{X}'_1\}_{new}) \times \overline{(\{\boldsymbol{X}'_2\}_{old} \cup \{\boldsymbol{X}'_2\}_{new})} \times \ldots \times (\{\boldsymbol{X}'_n\}_{old} \cup \{\boldsymbol{X}'_n\}_{new})] -$
$[\{\boldsymbol{X}'_1\}_{old} \times \{\boldsymbol{X}'_2\}_{old} \ldots \{\boldsymbol{X}'_n\}_{old}]$
(t the cycle count number, N as above).

10. Increment the cycle counter, t, by 1 and go to **7** while $P_n^{\max} > 0, \ \forall i$.

11. After termination of the major cycle **7-11**, subtract a value $t.\Delta z$ from each point of the deconvolution matrices to establish true PDFs, if required (see footnote 5).

12. Repeat from **2** for the remaining classes in the sequence $\omega_1, \omega_2 \ldots \omega_m$.

Choosing Parameters for Random Subspace Ensembles for fMRI Classification

Ludmila I. Kuncheva and Catrin O. Plumpton

School of Computer Science, University of Bangor, UK
{l.i.kuncheva,c.o.plumpton}@bangor.ac.uk

Abstract. Functional magnetic resonance imaging (fMRI) is a non-invasive and powerful method for analysis of the operational mechanisms of the brain. fMRI classification poses a severe challenge because of the extremely large feature-to-instance ratio. Random Subspace ensembles (RS) have been found to work well for such data. To enable a theoretical analysis of RS ensembles, we assume that only a small (known) proportion of the features are important to the classification, and the remaining features are noise. Three properties of RS ensembles are defined: usability, coverage and feature-set diversity. Their expected values are derived for a range of RS ensemble sizes (L) and cardinalities of the sampled feature subsets (M). Our hypothesis that larger values of the three properties are beneficial for RS ensembles was supported by a simulation study and an experiment with a real fMRI data set. The analyses suggested that RS ensembles benefit from medium M and relatively small L.

1 Introduction

Functional magnetic resonance imaging (fMRI) measures blood oxygenation level-dependent (BOLD) signal in a quest to discover how mental states are mapped onto patterns of neural activity. Advanced as they are, pattern recognition and machine learning are yet to contribute powerful bespoke techniques to fMRI data analysis [1, 2]. The formidable challenges come from: (i) the extremely large feature-to-instance ratio, in the order of 5000:1; (ii) the spatial relationship between the features (voxels in the 3-D image of the brain); (iii) the low contrast-to-noise ratio; and (iv) the great redundancy in the feature set. Preferences tend to be for linear classifiers because they are simple, fast, reasonably accurate and interpretable. The favourite, however, has been the support vector machine classifier (SVM) [3–7]. A recent comparison of classification methods for an fMRI data set placed the Random Subspace Ensemble (RS) with SVM base classifiers as the most accurate classification method across a variety of voxel pre-selection methods [8]. To construct a random subspace ensemble with L classifiers, L samples of size M are drawn without replacement from a uniform distribution over the set of voxels. A classifier is trained on each feature subset using either the whole training set or a bootstrap sample thereof [9].

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 54–63, 2010.

Random Subspace ensembles have been considered for problems with large dimensionality and excessive feature-to-instance ratio [10], e.g., problems arising from microarray data analysis [11] and face recognition [12]. The overwhelming computational demand in applying RS to the raw fMRI data led to the idea of pre-selection of voxels. Univariate statistical methods have been employed for that [13, 14]. The importance of a voxel is measured by the p-value of a t-test (or ANOVA for multiple classes) for equivalence of the class means. The initial set of voxels is subsequently reduced to a subset of 1000 or 2000 voxels, and RS ensembles are created on that subset. Admittedly, univariate approaches may destroy important relationships between features. Such features would not be indicative individually but may form a highly indicative group. In balance, pre-selection eliminates the vast majority of irrelevant voxels which justifies some (hypothetical) loss of discriminative information.

Relevance and redundancy are two different aspects in large-scale feature selection [15], and both are present in fMRI data. As the data is a "snapshot" of the whole brain, the vast majority of the voxels are irrelevant for each particular task. The relevant voxels, on the other hand, are likely to be spatially grouped into clusters exhibiting large correlations (redundancy). Most voxel selection methods do not guard against redundancy because the position, size and shape of the clusters of relevant voxels is of interest to the investigator. Thus we examine the effect of the number of relevant voxels on the parameter choices of RS ensembles. An advantage of RS ensembles compared to many other ensemble methods and single classifiers is that they need only two parameters, L, the ensemble size and M, the size of the feature sample. Given the specifics of fMRI data, this paper offers a theoretical perspective on choosing values of these parameters. Section 2 introduces the theoretical framework. Simulation experiments are reported in Section 3, and discussed in Section 4.

2 Random Subspace Ensembles

Let $X = \{x_1, \ldots, x_n\}$ be the set of n features (voxels). L samples, each of size M, are drawn without replacement from a uniform distribution over X and a classifier is trained on each sample. The ensemble decision is made by majority vote among the L classifiers.

In many fMRI studies, the relevant information is typically a sparse irregular pattern of responsive voxels in the 3-D image of the brain. It is likely that a small number of voxels contain most of the discriminative information. We assume that there are Q "important" voxels, set $\mathcal{I} = \{q_1, \ldots, q_Q\}$, $\mathcal{I} \subset X$, where $|\mathcal{I}| = Q << n$, and the remaining $n - Q$ voxels are random noise. We also assume that the cardinality of the subspaces, M, is much smaller than n. The question is whether we can recommend L and M for a given n and Q. We base this study on the following postulate [16–18].

Postulate. Accurate and diverse individual classifiers are a prerequisite for better ensembles. ∎

The subset of features, on which the individual classifiers are built, can serve as an indication of the expected accuracy and diversity of these classifiers. If a classifier uses only 'noise' features, its accuracy will be no better than random chance. Also, classifiers that use the same 'important' features will be similar or identical, therefore redundant in the ensemble. Finally, we would like the whole of $\mathcal{I}$ to be covered, so that important information is not lost. In other words, we would like each $q \in \mathcal{I}$ to be selected at least once in the L samples of M features.

2.1 Usability

Definition 1. We call a classifier *usable* if its feature subset contains at least one 'important' voxel $q \in \mathcal{I}$. ∎

To calculate the probability of drawing a feature subset of a usable classifier, take Y to be the number of 'important' features in a subset of size M, drawn without replacement from X. Y is a random variable with hypergeometric distribution with probability mass function

$$P(Y = i) = \frac{\binom{Q}{i}\binom{n-Q}{M-i}}{\binom{n}{M}}, \quad i = 0, 1, \ldots, Q.$$

The probability of drawing a usable classifier is

$$P(\text{usable classifier}) = 1 - P(Y = 0) = 1 - \frac{\binom{n-Q}{M}}{\binom{n}{M}}$$

Definition 2. The *degree of usability of the ensemble*, U, is defined as the proportion of usable classifiers out of L. ∎

Since the subsets are sampled independently, the probability of having a completely usable ensemble is

$$P(U = 1) = P(\text{usable classifier})^L = \left(1 - \frac{\binom{n-Q}{M}}{\binom{n}{M}}\right)^L. \tag{1}$$

The ratio of the two binomial coefficients can be simplified for computational purposes to give

$$P(U = 1) = \left(1 - \prod_{i=0}^{M-1}\left(1 - \frac{Q}{n-i}\right)\right)^L. \tag{2}$$

Since we assumed $M << n$, the equation can be simplified further to

$$P(U = 1) \approx \left(1 - \left(1 - \frac{Q}{n}\right)^M\right)^L. \tag{3}$$

This approximation is equivalent to replacing the hypergeometric distribution with a binomial distribution. Given the size of n for fMRI data, we can say that sampling *with* replacement is approximately equivalent to sampling *without* replacement. Y can therefore be approximated with a binomial distribution with parameters M and $p = \frac{Q}{n}$. The probability of a usable classifier in this case would be $1 - \left(1 - \frac{Q}{n}\right)^M$.

To calculate the expected value of the degree of usability of the ensemble, $E[U]$, let Z be a random variable expressing the number of usable classifiers in the ensemble. Z has a hypergeometric distribution with the following parameters. The *total* is the number of all possible samples (without replacement) of size M from X, i.e., $\binom{n}{M}$. The number of *usable* classifiers is calculated by taking the number of non-usable classifiers, $\binom{n-Q}{M}$, from the total. The number of *selected* classifiers at a time is L. The expected value of Z is $\frac{\text{Selected} \times \text{Usable}}{\text{Total}}$, therefore the expected usability of the ensemble is $E[U] = \frac{1}{L}E[Z]$

$$E[U] = \frac{1}{L} \times L \times \left(1 - \frac{\binom{n-Q}{M}}{\binom{n}{M}}\right) = 1 - \frac{\binom{n-Q}{M}}{\binom{n}{M}}. \tag{4}$$

The expected usability of the ensemble is equivalent to the probability of selecting a usable classifier, and does not depend on the ensemble size L. Our hypothesis is that the higher the degree of usability, the more accurate the ensemble.

2.2 Coverage

Definition 3. The *degree of coverage of the ensemble*, C, is the proportion of the features $q \in \mathcal{I}$ (out of Q) selected for one of more of the base classifiers. ■

For calculating coverage, we can again use the binomial approximation to the hypergeometric distribution. This implies that the feature subsets are sampled independently from the X. For a given important feature $q \in \mathcal{I}$ the probability of selecting that feature in a sample of size M is $\frac{M}{n}$. The probability of not selecting q in a sample of size M is therefore $1 - \frac{M}{n}$. The probability of not selecting q in at least one of L samples of size M is $P(\bar{q}) = \left(1 - \frac{M}{n}\right)^L$. The probability of q being selected in at least one of the L samples is $1 - P(\bar{q})$. The probability of all features being covered is

$$P(\text{Complete coverage}) = P(C = 1) = \left(1 - \left(1 - \frac{M}{n}\right)^L\right)^Q. \tag{5}$$

Denote by Z the number of covered features out of Q. Z has binomial distribution with parameters Q and $p = 1 - \left(1 - \frac{M}{n}\right)^L$. The expected coverage is

$$E[C] = \frac{1}{Q}\left(1 - \left(1 - \frac{M}{n}\right)^L\right) Q = 1 - \left(1 - \frac{M}{n}\right)^L. \tag{6}$$

The expected coverage depends on the ensemble size L and the subset size M but not on Q. The hypothesis here is that the higher the degree of coverage, the more accurate the ensemble.

2.3 Feature Set Diversity

Note that for fixed n and Q, $E[U]$ is monotonically increasing on M, and $E[C]$ increases with both L and M. This suggests that larger ensembles with larger feature sample size M should be preferred. In the extreme case where $M = n$, the ensemble will contain identical copies of the base classifier trained on all features. This defeats the point of having an ensemble altogether. Besides, with the extremely large feature-to-instances ratio, the individual classifier may easily overfit the data. Therefore we introduce a third property.

Definition 4. Denote by $S_1, S_2, \ldots, S_L$ the L feature subsets sampled from X. Consider $S_1, S_2 \subset X$ such that $|S_1| = |S_2| = M$. Denote by $I_1 \subseteq \mathcal{I}$ and $I_2 \subseteq \mathcal{I}$ the respective subsets of 'important' features in S_1 and S_2 respectively. We define *Feature Set Diversity* (D) as

$$D(S_1, S_2) = |I_1 \cup I_2| - |I_1 \cap I_2|. \qquad \blacksquare$$

Two classifiers are *non-identical* if their feature subsets differ by at least one 'important' voxel. Each feature $q \in \mathcal{I}$ may or may not contribute to D. A value of 1 will be added if q is in either set but not in both. Then the expected diversity for any pair of subsets S_1 and S_2 is

$$E[D] = \sum_{i=1}^{Q} P(q_i \in I_1)P(q_i \notin I_2) + P(q_i \notin I_1)P(q_i \in I_2).$$

Since all features in $\mathcal{I}$ have equal chance of being selected in a subset of size M, and the subsets are drawn independently,

$$E[D] = 2Q\frac{M}{n}\left(1 - \frac{M}{n}\right). \tag{7}$$

This calculation disregards non-usable classifiers. So an ensemble can be diverse even if it contains non-usable classifiers for which $I_1 = I_2 = \emptyset$.

Figure 1 shows the theoretical and simulated curves for $E[U]$ (4), $E[C]$ (6) and $E[D]$ (7) for $n = 1000$, $Q = 100$ and $L = 10$. Changing the value of L to 50 and 100, and Q to 10 and 50 did not lead to large differences in the shapes and positions of the curves. The results suggest that values of M close to $\frac{n}{2}$ are optimal as all three criteria reach their maxima, also observed across different ensemble sizes.

3 A Simulation Experiment

The important question here is to what extent the three characteristics are related to the classification accuracy of the RS ensemble.

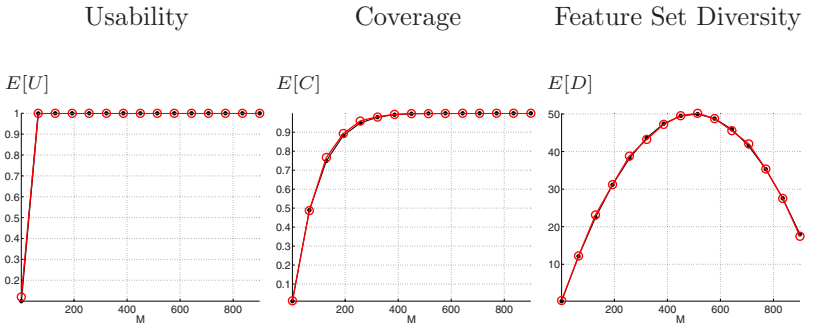


Fig. 1. Theoretical and simulation curves (coinciding) for the expected values of U, C and D for $n = 1000$, $Q = 100$ and $L = 10$. The empirical curve is calculated as an average of 10 ensembles with randomly sampled $L = 10$ sets of M features.

3.1 Data

We decided to use simulated data that exhibit properties similar to real data while keeping control on the parameters n and Q. We used an fMRI data set collected at the School of Psychology, University of Bangor. The data consisted of the single-subject BOLD responses to 3 types of stimuli: faces, places and objects. Each presentation of a stimulus defined a point in the data set. The total number of voxels (features) was 106720 and the number of objects was 36, 12 in each class. The classification task was to predict which type of stimuli the subject is looking at, judging by the fMRI response.

For a 2-class problem, Contrast-to-Noise-Ratio (CNR) is defined using the means and the standard deviations for the classes, separately for each voxel. For voxel v, CNR is $\frac{\mu_1(v)-\mu_2(v)}{\frac{1}{2}(\sigma_1(v)+\sigma_2(v))}$, where $\mu_i(v)$ is the mean and $\sigma_i(v)$ is the standard deviation of v for class i. The higher the CNR, the more separable the two classes are using only voxel v. We took only classes 1 and 2 (faces and places) and calculated CNR for each voxel. The voxels were then sorted by their CNR, in descending order. The means and the covariance matrices for the two classes of the top Q voxels were stored and subsequently used to simulate the first (important) Q features in the data. We simulated multivariate Gaussian distributions for each class, using the Statistics toolbox of Matlab. The remaining $n - Q$ features were simulated as independent random noise with mean zero and standard deviation equal to the mean CNR for the Q important features. Running a separate simulation study even in addition to the experiments with the real data was necessary in order to have *control over* Q in a surrogate pseudo-real environment.

3.2 Experimental Protocol

The parameters were varied in the following ranges: the total number of features, n, took values 200, 500 and 1000, and the number of 'important' features, Q,

was chosen accordingly to model ratios $\frac{Q}{n}$ of 0.02, 0.05, 0.1, 0.25, 0.5 and 1. The feature set cardinality, M, took 20 equally spaced values from 1 to n, and the ensemble size L took values at regular intervals from 1 to 200.

For each combination (M, L, Q, n), we generated 10 data sets with 20 training examples (10 per class) and 200 testing examples (100 per class). The small size of the training data was chosen to mirror that of real data sets. SVM was used as the base classifier. For each (M, L, Q, n) we calculated the RS ensemble error, and also estimated the observed degree of usability U, the degree of coverage C, and the feature set diversity D of the ensemble.

3.3 Results

Figure 2 gives an example of the type of surfaces over the (L, M) grid, obtained through the simulations. Each point in the space is calculated as the average across 10 simulations with data drawn independently from the chosen 'realistic'

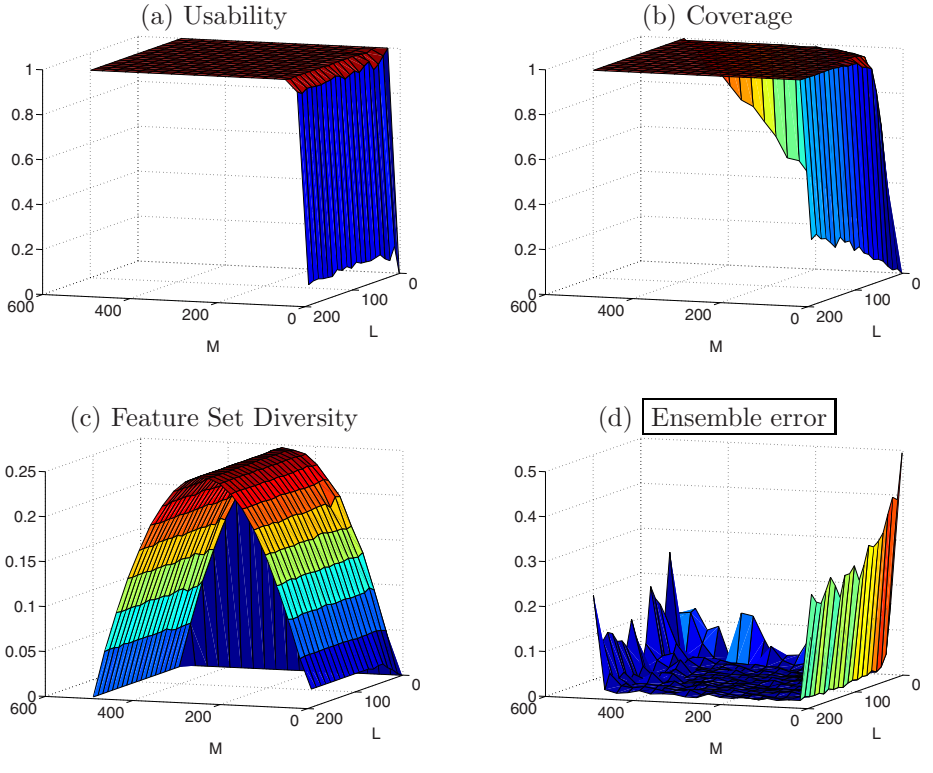


Fig. 2. The three RS characteristics and the ensemble error as functions of the ensemble size L and the feature set size M. Each of the 2 classes in the data set was sampled from a Gaussian distributions with $Q = 50$ relevant and $n - Q = 450$ noise features.

Table 1. Summary of the simulation results. $\bar{E}$ is the average RS ensemble error across the (L, M) grid. $\bar{E}^*$ is the value of the ensemble error for the recommended parameter values, $M = \frac{n}{2}$ and $L = \frac{n}{10}$.

			Correlation with E		
$\frac{Q}{n}$ ratio	$\bar{E}$	$\bar{E}^*$	Usability	Coverage	Diversity
0.02	4.09	1.25	−0.940	−0.863	−0.410
0.05	3.81	1.75	−0.972	−0.903	−0.438
0.10	3.83	1.40	−0.855	−0.802	−0.581
0.25	3.19	0.90	−0.593	−0.640	−0.608
0.50	3.65	2.15	−0.123	−0.233	−0.608
1.00	6.88	1.05	N/A	0.014	−0.448

distribution. The surfaces in the plots were obtained with $n = 500$ and $Q = 50$. They have typical shapes observed for the six $\frac{Q}{n}$ ratios. The surfaces confirm visually the hypothesis that larger usability, coverage and feature set diversity lead to better ensembles (lower error).

As expected, the degree of usability (subplot (a)) does not depend on L, and quickly raises to 1 with M. The 'tent'-shaped surface of D in subplot (c) also depends on M but not on L. Largest values of D are achieved for $M \approx \frac{n}{2}$. Like U, the degree of coverage, C, maintains its maximum value of 1 for the largest part of the (L, M) grid. The figure also suggests that small to medium values of the ensemble size L are sufficient. Hence, as a rule of thumb, we recommend $M = \frac{n}{2}$ and $L = \frac{n}{10}$ for fMRI type of data.

Table 1 shows a summary of the simulation results. We show the $\frac{Q}{n}$ ratio, the average error rate of the RS ensemble over the whole (L, M) grid, $\bar{E}$, as well as the error using the *recommended* values, $\bar{E}^*$. For all $\frac{Q}{n}$ ratios, $\bar{E} > \bar{E}^*$. We also give the correlation coefficients between the RS ensemble error E, on the one hand, and U, C, and D, on the other hand. Even though calculated on an artificial grid, these coefficients support the hypothesis that large values of usability, coverage and feature-set diversity are beneficial for the ensemble.

3.4 Experiment with the Real fMRI Data

The RS ensemble with SVM base classifiers was run on classes 1 and 2 (faces and places) of the real fMRI data set. First, $n = 1000$ voxels were pre-selected by the SVM method [14]. An SVM classifier was trained on all voxels, the voxels were sorted by descending absolute value of the SVM weights, and the top 1000 voxels were retained. Three-fold cross-validation was applied to test the RS ensemble for a 10×10 grid of values for M and L. M was varied from 1 to n at equal intervals, and L was varied from 1 to $n/5$. Figure 3 plots the surface of the ensemble error over the (L, M) grid. The recommended values of $M = 500$

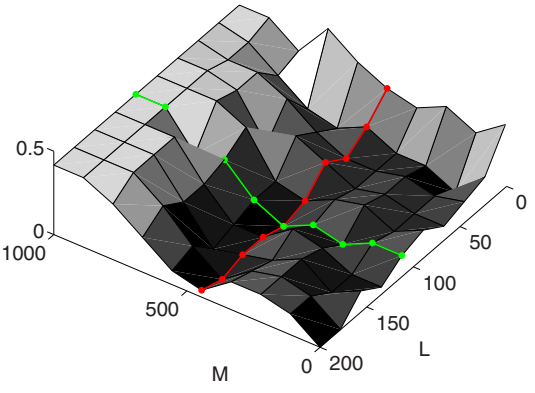


Fig. 3. RS error on the real fMRI data set as a function of the ensemble size L and the feature size M. The recommended values for L and M are overlayed on the surface.

and $L = 100$ are marked as lines on the 3-D plot. The lines intersect near the minimum of the error surface, which confirms empirically the recommendation for L and M. While the average error over the whole grid was 0.2138, the error at $M = 500$ and $L = 100$ was 0.0521.

4 Conclusions

We examine the Random Subspace ensemble (RS) for fMRI type of data where the feature-to-instance ratio is in the order of 50000:1, and the number of truly relevant features (voxels in the 3-D image of the brain) is much smaller than the total number of features. Following previous fMRI studies we consider n pre-selected voxels, where n is in the region of 1000. Assuming that there are Q 'important' features among the pre-selected n features, three characteristics of the RS ensemble are defined: usability U, coverage C and feature-set diversity D. Expected values of these characteristics are derived theoretically as functions of n, Q, L and M. Our hypothesis was that higher values of U, C and D are beneficial for the RS ensemble. A simulation study was carried out, with two heteroscedastic Gaussian classes whose covariance matrices were estimated from a real fMRI data set and augmented with Gaussian noise. The results support the research hypothesis. As a rule of thumb, we propose to use feature set size $M = \frac{n}{2}$ and ensemble size $L = \frac{n}{10}$. These values were found to work well for the real fMRI data.

Acknowledgements

We are grateful to David Linden and Stephen Johnston, School of Psychology, Bangor University, UK, for providing the fMRI data.

References

1. Norman, K.A., Polyn, A.M., Detre, G.J., Haxby, J.V.: Beyond mind-reading: multi-voxel pattern analysis of fMRI data. Trends in Cognitive Sciences 10, 424–430 (2006)
2. Pereira, F., Mitchell, T., Botvinick, M.: Machine learning classifiers and fMRI: a tutorial overview. NeuroImage 45(1, suppl. 1), 199–209 (2009)
3. Cox, D.D., Savoy, R.L.: Functional magnetic resonance imaging (fMRI): detecting and classifying distributed patterns of fMRI activity in human visual cortex. NeuroImage 19(2), 261–270 (2003)
4. LaConte, S., Strother, S., Cherkassky, V., Anderson, J., Hu, X.: Support vector machines for temporal classification of block design fMRI data. NeuroImage 26(2), 317–329 (2005)
5. Mourao-Miranda, J., Bokde, A.L., Born, C., Hampel, H., Stetter, M.: Classifying brain states and determining the discriminating activation patterns: Support Vector Machine on functional MRI data. NeuroImage 28(4), 980–995 (2005)
6. Wang, Z., Childress, A.R., Wang, J., Detre, J.A.: Support vector machine learning-based fMRI data group analysis. NeuroImage 36(4), 1139–1151 (2007)
7. Ku, S.-p., Gretton, A., Macke, J., Logothetis, N.K.: Comparison of pattern recognition methods in classifying high-resolution BOLD signals obtained at high magnetic field in monkeys. Magnetic Resonance Imaging 26(7), 1007–1014 (2008)
8. Kuncheva, L.I., Rodríguez, J.J.: Classifier ensembles for fMRI data analysis: An experiment. Magnetic Resonance Imaging (to appear)
9. Ho, T.K.: The random space method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
10. Skurichina, M., Duin, R.P.W.: Bagging, boosting and the random subspace method for linear classifiers. Pattern Analysis & Applications 5, 121–135 (2002)
11. Lai, C., Reinders, M.J., Wessels, L.: Random subspace method for multivariate feature selection. Pattern Recognition Letters 27(10), 1067–1076 (2006)
12. Zhu, Y., Liu, J., Chen, S.: Semi-random subspace method for face recognition. Image and Vision Computing 27(9), 1358–1370 (2009)
13. Friston, K., Ashburner, J., Kiebel, S., Nichols, T., Penny, W. (eds.): Statistical Parametric Mapping: The Analysis of Functional Brain Images. Academic Press, London (2007)
14. De Martino, F., Valente, G., Staeren, N., Ashburner, J., Goebel, R., Formisano, E.: Combining multivariate voxel selection and support vector machines for mapping and classification of fMRI spatial patterns. NeuroImage 43(1), 44–58 (2008)
15. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research 5, 1205–1224 (2004)
16. Kuncheva, L.I.: Combining Pattern Classifiers. Methods and Algorithms. John Wiley and Sons, New York (2004)
17. Brown, G.: Ensemble learning. In: Sammut, C., Webb, G. (eds.) Encyclopedia of Machine Learning. Springer, Heidelberg (2009)
18. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Information Fusion 6(1), 5–20 (2005)

An Experimental Study on Ensembles of Functional Trees*

Juan J. Rodríguez, César García-Osorio, Jesús Maudes,
and José Francisco Díez-Pastor

University of Burgos, Spain
{jjrodriguez,cgosorio,jmaudes}@ubu.es, jdp0014@alu.ubu.es

Abstract. Functional Trees are one type of multivariate trees. This work studies the performance of different ensemble methods (Bagging, Random Subspaces, AdaBoost, Rotation Forest) using three variants (multivariate internal nodes, multivariate leaves or both) of these trees as base classifiers. The best results, for all the ensemble methods, are obtained using Functional Trees with multivariate leaves and univariate internal nodes. The best overall configuration is obtained with Rotation Forest. Ensembles of Functional Trees are compared to ensembles of univariate Decision Trees, being the results favourable for the variant of Functional Trees with univariate internal nodes and multivariate leaves. Kappa-error diagrams are used to study the diversity and accuracy of the base classifiers.

Keywords: Functional Trees, Decision Trees, Bagging, Random Subspaces, Boosting, Rotation Forest.

1 Introduction

Decision trees [1] are one of the most commonly used base classifiers for ensemble methods [2]. They are fast and rather unstable, making them ideal as base classifiers.

Usually, decision trees are assumed to be univariate, that is, each internal node only considers one attribute. Nevertheless, there are several approaches for constructing multivariate trees [3,4,5,6]. Although these methods can improve the results of univariate decision trees, they can be less adequate as base classifiers: they are slower and more stable than univariate decision trees.

This work studies the application of different ensemble methods to Functional Trees [5] (a type of multivariate trees). In that work [5], Bagging was used as a variance reduction method. The present work also considers different ensemble methods (Random Subspaces [7], AdaBoost [8], Rotation Forest [9]) and compares them with the corresponding ensembles of univariate Decision Trees.

* This work was supported by the "Caja de Burgos" and University of Burgos Project 2009/00204/001.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 64–73, 2010.

There are some works that considered ensembles of multivariate decision trees. In [7], the Random Subspace method was used with trees using different splitting functions, some of them multivariate. In [10], Bagging was used with Mean Margins Decision Trees, a type of multivariate tree.

One of the current approaches for ensemble construction is training the base classifiers using different projections of the data set [11,9,12,13]. When using univariate decision trees as base classifiers of these methods, the resulting trees are multivariate on the original space. Hence, a sensible question is if it is better to use non projection-based ensemble methods with multivariate trees or projection-based ensemble methods with univariate trees. This work considers Rotation Forest as the projection-based method.

The combination of projection-based ensembles with multivariate trees can seem pointless at first glance (at least for linear projections), but it can have sense if we take into account that in Functional Trees we can restrict multivariate nodes to the internal nodes or the leaves. If the internal nodes are univariate, the classifiers constructed in different subspaces can be rather diverse.

The rest of the paper is organized as follows. Section 2 gives an introduction to Functional Trees. The experiments and results using ensembles of Functional Trees are described in Section 3. Ensembles of Functional Trees are compared to Ensembles of Decision Trees in Section 4. Section 5 shows kappa-error diagrams. The conclusions are presented in Section 6.

2 Functional Trees

Functional trees [5] are multivariate trees that can use combinations of attributes in the leaves and/or in the internal nodes. If the combinations of attributes are allowed in the two types of nodes, they are denoted as FT. Otherwise, they are denoted FT-Inner or FT-Leaves, depending if the combinations are allowed in one type of nodes or in the other. They can be used for classification and regression, although this work will only consider classification.

The method follows the standard top-down approach. For each internal node, a model is constructed using the training examples available for that node. In the used implementation [14] the models are constructed using logistic regression [6]. As many attributes as classes are created, their values are the probabilities predicted by the model for the corresponding class. For the selection of the attribute for the internal node, both the original and the newly constructed attributes are considered. If one of the new attributes is selected, the internal node will be multivariate.

The construction of the logistic models is done according to the method described in [6]. LogitBoost [15] is used, the base regressors that it combines are obtained using linear functions of one of the attributes.

When pruning the tree, the method considers the replacement of an internal node with a leaf that predicts a class, but also with a leaf that predicts according to the logistic model. Hence, it is also possible to have models in the leaves. In this type of leaves, the prediction is not always the same class, but the prediction obtained using the logistic model (a classifier) that is associated to that leaf.

Fig. 1 shows the three types of Functional Trees for the "dna" data set. Logistic models can appear, depending on the type, in internal nodes and leaves. They are denoted, respectively, LMI and LML, followed by a sequential number.

In the internal nodes, the logistic models are used to split the data, comparing the probability predicted by the logistic model for one of the classes with a threshold. For instance, "LMI1 C3" indicates that the value compared is the probability for the third class according to the logistic model LMI1.

Instead of the logistic models, in the internal nodes can appear the original attributes (e.g., a090, a150), while in the leaves can appear the predicted class (e.g., Class=3).

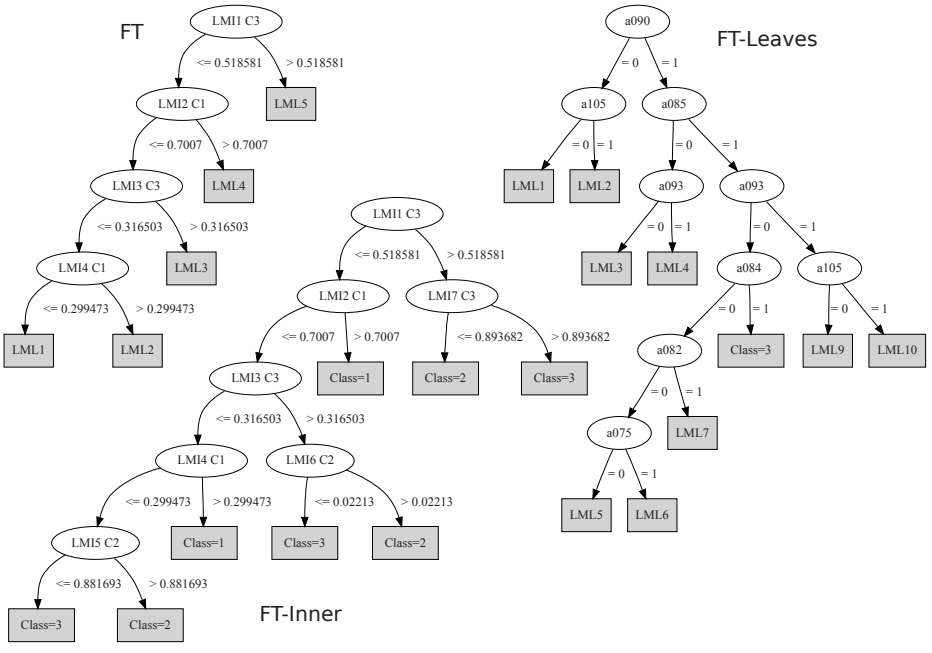


Fig. 1. Examples of Functional Trees

3 Experiments: Ensembles of Functional Trees

3.1 Settings

Table 1 shows the characteristics of the data sets used in the experiments. They are from the UCI repository [16].

Weka [14] was used for the experiments. The experiments were done using 5×2 folds cross-validation [17]. The ensemble size was 10. The reason for this small number is the complexity of functional trees, in memory size and training and testing times.

Table 1. Summary of the data sets used in the experiments

Dataset	#N	#D	#E	#C
abalone	7	1	4177	28
anneal	6	32	898	6
audiology	0	69	226	24
autos	15	10	205	6
balance-scale	4	0	625	3
breast-w	9	0	699	2
breast-y	0	9	286	2
bupa	6	0	345	2
car	0	6	1728	4
credit-a	6	9	690	2
credit-g	7	13	1000	2
crx	6	9	690	2
dna	0	180	3186	3
ecoli	7	0	336	8
glass	9	0	214	6
heart-c	6	7	303	2
heart-h	6	7	294	2
heart-s	5	8	123	2
heart-statlog	13	0	270	2
heart-v	5	8	200	2
hepatitis	6	13	155	2
horse-colic	7	15	368	2
hypo	7	18	3163	2
ionosphere	34	0	351	2
iris	4	0	150	3
krk	6	0	28056	18
kr-vs-kp	0	36	3196	2
labor	8	8	57	2
led-24	0	24	5000	10
letter	16	0	20000	26
lrs	93	0	531	10
lymphography	3	15	148	4
mushroom	0	22	8124	2
nursery	0	8	12960	5
optdigits	64	0	5620	10
page	10	0	5473	5
pendigits	16	0	10992	10
phoneme	5	0	5404	2
pima	8	0	768	2
primary	0	17	339	22
promoters	0	57	106	2
ringnorm	20	0	300	2
sat	36	0	6435	6
segment	19	0	2310	7
shuttle	9	0	58000	7
sick	7	22	3772	2
sonar	60	0	208	2
soybean	0	35	683	19
soybean-small	0	35	47	4
splice	0	60	3190	3
threenorm	20	0	300	2
tic-tac-toe	0	9	958	2
twonorm	20	0	300	2
vehicle	18	0	846	4
vote1	0	15	435	2
voting	0	16	435	2
vowel-context	10	2	990	11
vowel-nocontext	10	0	990	11
waveform	40	0	5000	3
yeast	8	0	1484	10
zip	256	0	9298	10
zoo	1	15	101	7

#N: numeric features, #D: discrete features, #E: examples, #C: classes.

As base classifiers, the three variants of Functional Trees were considered: using models in the internal nodes and leaves (denoted as FT), using models only in the leaves (denoted ad FT-Leaves) and using models only in the internal nodes (denoted as FT-Inner).

As ensemble methods, five configurations were considered: Bagging, Random Subspaces (using 50% or 75% of the features), AdaBoost and Rotation Forest. The combinations of base classifiers and ensemble methods produces 15 different configurations. Moreover, the three base classifiers are also considered as standalone methods.

For each data set and configuration, the accuracy is calculated. For comparing the methods, average ranks [18] are used. The value of the average ranks are

Table 2. Average ranks of ensemble methods with functional trees

	Average Rank	Ensemble Method	Base Classifier
1	4.23	Rotation Forest	FT-Leaves
2	5.98	Bagging	FT-Leaves
3	7.01	Subspaces-75%	FT-Leaves
4	7.23	Subspaces-75%	FT
5	7.42	Rotation Forest	FT
6	8.19	Bagging	FT
7	8.41	Subspaces-50%	FT-Leaves
8	8.90	AdaBoost	FT-Leaves
9	9.61	AdaBoost	FT
10	9.81	Subspaces-50%	FT
11	9.90	Rotation Forest	FT-Inner
12	10.60	Subspaces-75%	FT-Inner
13	10.68	Bagging	FT-Inner
14	11.01	Single	FT-Leaves
15	11.50	AdaBoost	FT-Inner
16	12.94	Subspaces-50%	FT-Inner
17	13.02	Single	FT
18	14.54	Single	FT-Inner

	FT-Leaves	FT	FT-Inner
Rotation Forest	4.23	7.42	9.90
Bagging	5.98	8.19	10.68
Subspaces-75%	7.01	7.23	10.60
Subspaces-50%	8.41	9.81	12.94
AdaBoost	8.90	9.61	11.50
Single	11.01	13.02	14.54

between one and the number of considered configurations. Smaller values of the average rank indicate better results.

3.2 Results

Table 2 shows the average ranks of the considered methods. The best configuration is Rotation Forest using FT-Leaves as base classifier. The second and third configurations also use FT-Leaves, with Bagging and Random Subspaces (75%), respectively.

For all the considered ensemble methods, FT-Leaves as base classifier has a better average rank than using FT. Moreover, FT as a base classifier also has a better average rank than using FT-Inner. The same order (FT-Leaves, FT and FT-Inner) is preserved when using these methods without ensembles.

The results for AdaBoost are worse than the results for the other method (with the possible exception of Subspaces-50%). A possible cause could be that Boosting is used in the construction of Functional Trees, using Boosting twice could be redundant.

4 Comparison with Ensembles of Decision Trees

In this section, ensembles of functional trees are compared with ensembles of decision trees.

Table 3 shows the average ranks for each ensemble method. Five base classifiers are considered, the three types of Functional Trees and Decision Trees, pruned and unpruned. There is a set of average ranks for each ensemble method, the values are between one and five, because five base classifiers are considered. For all the considered cases, FT-Leaves has the better average rank.

Table 3. Average ranks of decision and functional trees as base classifiers for different ensemble methods

	FT-Leaves	FT	DT-Pruned	DT-Unpruned	FT-Inner
Single	2.01	2.60	3.23	3.90	3.27
Bagging	2.19	2.68	3.29	3.39	3.45
Subspaces-50%	2.09	2.64	3.33	2.98	3.96
Subspaces-75%	2.23	2.40	3.48	3.48	3.42
AdaBoost	2.35	2.71	3.23	3.48	3.24
Rotation Forest	2.12	3.15	3.00	2.86	3.87
Average	2.19	2.71	3.26	3.35	3.49

Table 4 shows the average ranks obtained from all the considered configurations. The best average rank is again for Rotation Forest of FT-Leaves. The second position is for Bagging of FT-Leaves, while the third and fourth positions are for Rotation Forest with Decision Trees. The first configuration that does not uses Functional Trees nor Rotation Forests is Bagging of pruned Decision Trees, in position 15.

Table 4. Average ranks of ensemble methods with Functional and Decision Trees

	Average Rank	Ensemble Method	Base Classifier
1	6.83	Rotation Forest	FT-Leaves
2	9.19	Bagging	FT-Leaves
3	9.24	Rotation Forest	DT-Unpruned
4	9.46	Rotation Forest	DT-Pruned
5	10.54	Subspaces-75%	FT-Leaves
6	11.12	Subspaces-75%	FT
7	11.18	Rotation Forest	FT
8	12.27	Bagging	FT
9	13.07	Subspaces-50%	FT-Leaves
10	13.11	AdaBoost	FT-Leaves
11	14.69	AdaBoost	FT
12	14.93	Subspaces-50%	FT
13	15.23	Rotation Forest	FT-Inner
14	15.82	Single	FT-Leaves
15	15.85	Bagging	DT-Pruned
16	16.32	Bagging	FT-Inner
17	16.35	Subspaces-75%	FT-Inner
18	16.60	Bagging	DT-Unpruned
19	16.65	AdaBoost	DT-Pruned
20	17.15	AdaBoost	DT-Unpruned
21	17.32	AdaBoost	FT-Inner
22	18.20	Subspaces-50%	DT-Unpruned
23	18.21	Subspaces-75%	DT-Pruned
24	18.73	Subspaces-75%	DT-Unpruned
25	18.90	Subspaces-50%	DT-Pruned
26	19.18	Single	FT
27	20.33	Subspaces-50%	FT-Inner
28	21.98	Single	FT-Inner
29	22.10	Single	DT-Pruned
30	24.41	Single	DT-Unpruned

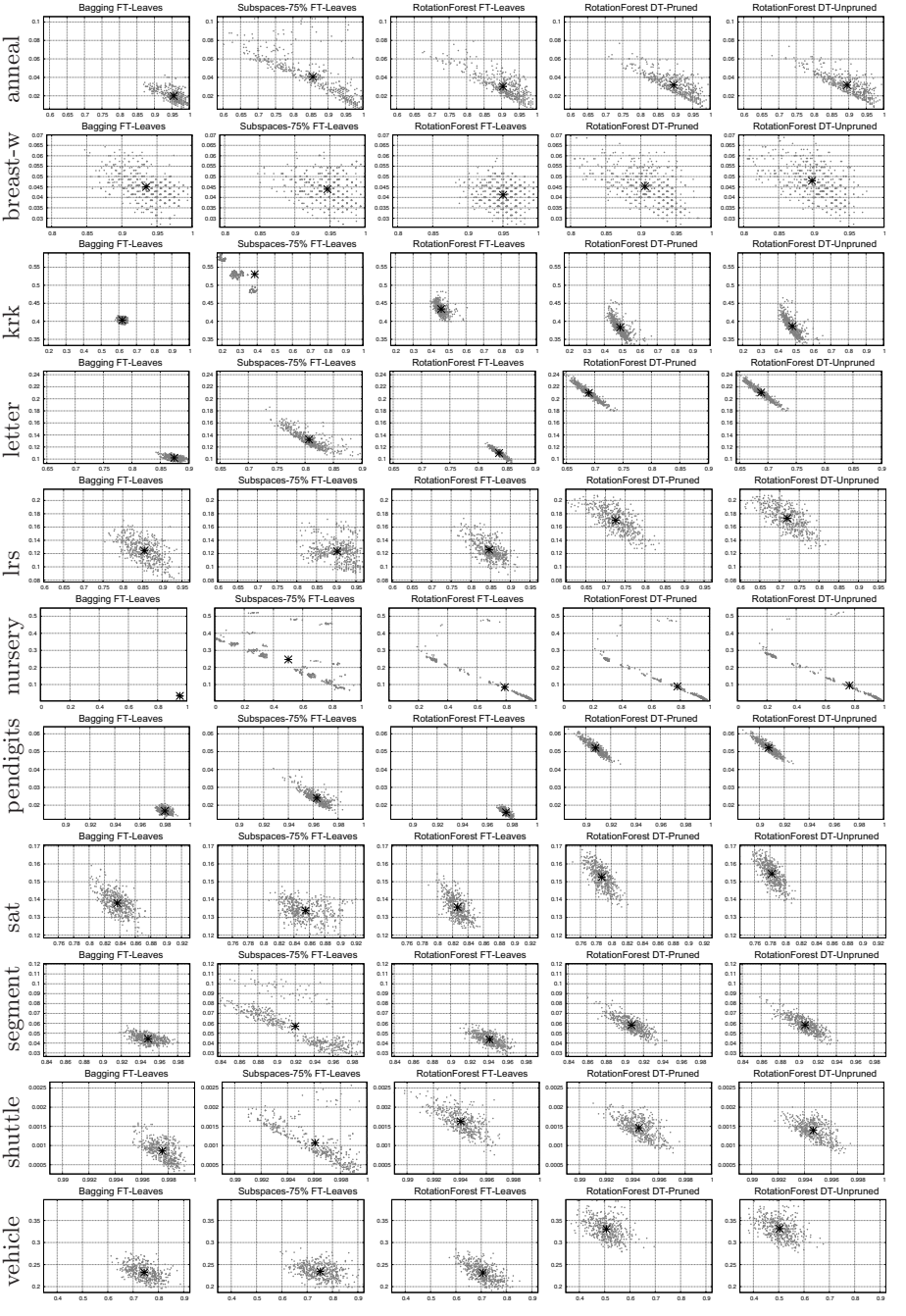


Fig. 2. Kappa-error diagrams

Regarding the question if it is better to use a projection-based ensemble method with univariate trees or a non projection-based ensemble method with multivariate trees, the first answer is that neither is the best. Projection-based ensemble methods are not incompatible with multivariate trees, the best method is Rotation Forests of FT-Leaves. A more direct answer is that Rotation Forest of Decision Trees are better (according to the average ranks) than all the non projection-based ensembles of Functional Trees, with only one exception, Bagging of FT-Leaves. Nevertheless, the average ranks are very similar (9.19, 9.24 and 9.46).

5 Kappa-error Diagrams

Figure 2 shows kappa-error diagrams [19], for some data sets and the best five configurations according to the average ranks. In each diagram, the average of the points is shown with a star.

In order to summarize the kappa-error diagrams for all the data sets, kappa-error relative movement diagrams [20] are used. Figure 3 shows these diagrams. They show the relationship, for all the data sets, between the kappa error diagrams of two methods. Given the average values of the points in the kappa error diagrams of the two methods, an arrow can be drawn from one of the centers to the other. In relative movement diagrams, the base of the arrow is taken to

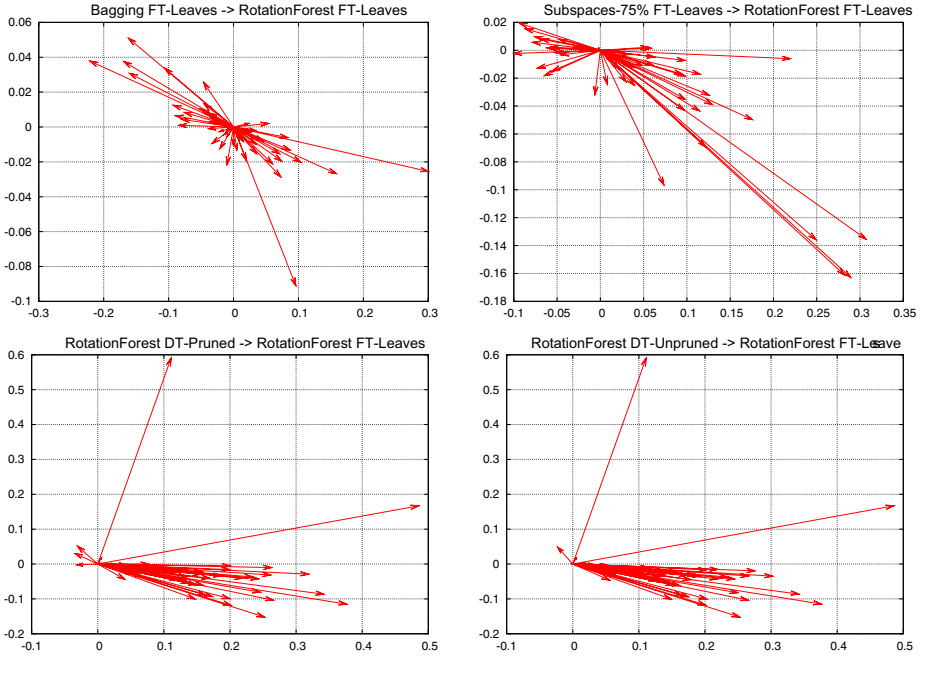


Fig. 3. Kappa-error relative movement diagrams

the origin of coordinates and there are as many arrows as data sets. In these diagrams, the horizontal axis represents the kappa difference while the vertical axis represents the error difference.

Comparing Bagging and Rotation Forests of FT-Leaves, there is not a clear pattern. There are very few arrows that go top right (the undesired direction, less diversity and more error). In the comparison of Random Subspaces and Rotation Forest of FT-Leaves, again very few arrows go top right. There are more arrows that go down than up. The arrows that go down right can be much longer than the others.

Comparing Rotation Forest of Decision Trees with FT-Leaves, there are only two arrows that go up right, but they are rather long. The great majority of the arrows go down right. This means that for FT-Leaves the base classifiers are less diverse but more accurate. Hence, the improvements obtained on Rotation Forests by using FT-Leaves as base classifiers instead of Decision Trees are due to the better accuracies of the base classifiers. The diagrams for pruned and unpruned decision trees are very similar.

6 Conclusions

This work presents an experimental study on ensembles of Functional Trees. The three types of Functional Trees (models in the leaves, models in the internal nodes, models in both) have been considered as base classifiers for Bagging, Random Subspaces, AdaBoost and Rotation Forest.

According to the average ranks, FT-Leaves is better than FT and FT is better than FT-Inner, regardless of the ensemble method used.

Ensembles of Functional Trees have been compared with ensembles of Decision Trees (both pruned and unpruned). Ensembles of FT-Leaves have better average ranks than ensembles of Decision Trees.

From all the considered configurations, including ensembles of decision trees, the best overall configuration is Rotation Forest of FT-Leaves. Comparing Rotation Forests of Decision Trees with other ensembles of Functional Trees, the average ranks are favourable for Rotation Forest with only one exception, Bagging of FT-Leaves.

Kappa-error diagrams have been used to study the diversity and accuracy of the base classifiers. Generally, the base classifiers in Rotation Forest of FT-Leaves are less diverse but more accurate than the base classifiers of Rotation Forest of Decision Trees.

Acknowledgements

We wish to thank the developers of Weka. We also express our gratitude to the donors of the different datasets and the maintainers of the UCI Repository.

References

1. Quinlan, J.R.: C4.5: Programs for Machine Learning. Machine Learning (1993)
2. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
3. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 2, 1–32 (1994)
4. Brodley, C.E., Utgoff, P.E.: Multivariate decision trees. Machine Learning 19, 45–77 (1995)
5. Gama, J.: Functional trees. Machine Learning 55, 219–250 (2004)
6. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Machine Learning 59, 161–205 (2005)
7. Ho, T.K.: The random space method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 832–844 (1998)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55, 119–139 (1997)
9. Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1619–1630 (2006)
10. Gashler, M., Giraud-Carrier, C., Martinez, T.: Decision tree ensemble: Small heterogeneous is better than large homogeneous. In: Seventh International Conference on Machine Learning and Applications, ICMLA 2008, pp. 900–905 (2008)
11. Tumer, K., Oza, N.C.: Input decimated ensembles. Pattern Anal. Applic. 6, 65–77 (2003)
12. García-Pedrajas, N., García-Osorio, C., Fyfe, C.: Nonlinear boosting projections for ensemble construction. Journal of Machine Learning Research 8, 1–33 (2007)
13. Schclar, A., Rokach, L.: Random projection ensemble classifiers. In: Enterprise Information Systems 11th International Conference Proceedings. LNBIP, vol. 24, pp. 309–316. Springer, Heidelberg (2009)
14. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005), http://www.cs.waikato.ac.nz/ml/weka
15. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics 38, 337–374 (2000)
16. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
17. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1923 (1998)
18. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
19. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning, pp. 211–218. Morgan Kaufmann, San Francisco (1997)
20. Rodríguez, J.J., García-Osorio, C., Maudes, J.: Forests of nested dichotomies. Pattern Recognition Letters 31, 125–132 (2010)

Multiple Classifier Systems under Attack

Battista Biggio, Giorgio Fumera, and Fabio Roli

Dept. of Electrical and Electronic Eng., Univ. of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
{battista.biggio,fumera,roli}@diee.unica.it
http://prag.diee.unica.it

Abstract. In adversarial classification tasks like spam filtering, intrusion detection in computer networks and biometric authentication, a pattern recognition system must not only be accurate, but also *robust* to manipulations of input samples made by an adversary to mislead the system itself. It has been recently argued that the robustness of a classifier could be improved by avoiding to overemphasize or underemphasize input features on the basis of training data, since at operation phase the feature importance may change due to modifications introduced by the adversary. In this paper we empirically investigate whether the well known bagging and random subspace methods allow to improve the robustness of linear base classifiers by producing more uniform weight values. To this aim we use a method for performance evaluation of a classifier under attack that we are currently developing, and carry out experiments on a spam filtering task with several linear base classifiers.

1 Introduction

In *adversarial classification* tasks like spam filtering, intrusion detection in computer networks and biometrics [1,2,3,4], the goal of a pattern recognition system is to discriminate between two classes, which can be named "legitimate" and "malicious", while an intelligent adversary manipulates samples to mislead the system itself. Adversarial classification problems are therefore non-stationary, which implies that a pattern recognition system should be designed by taking into account not only its accuracy (usually evaluated from a set of training samples) but also its robustness, namely the capability of undergoing an accuracy degradation as low as possible when it is under attack. Very few works addressed so far the problem of devising practical methods to improve robustness. Recently, in [7] it was suggested that a more robust classifier could be obtained by avoiding to give features too much or too little emphasis during classifier training, and a similar approach was suggested in [6]. This allows to design robust classifiers against attacks in which the adversary exploits some knowledge on the classification function (e.g., the most discriminant features), as we discuss in the next section.

It is well known that one of the main motivations for the use of multiple classifier systems (MCSs) is the improvement of classification accuracy with respect

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 74–83, 2010.

to a single classifier. Recently, MCSs have also been applied to adversarial classification tasks based only on intuitive motivations, although there is no clear evidence so far that they can be also useful to improve robustness. Our aim is to investigate whether and under which conditions MCSs allow to improve the robustness of a pattern recognition system in adversarial classification tasks, with respect to a single classifier architecture. In this work we will focus on two of the most known methods for constructing MCSs, namely bagging [8] and the random subspace method (RSM) [9]. The reason is that we argue that these methods could result in avoiding to give features too much or too little emphasis with respect to a single classifier during classifier training, which is the strategy suggested in [7] to improve robustness. Accordingly, we first experimentally investigate whether this assumption holds in practice, and then, whether this allows bagging and RSM to produce more robust classifiers. Our experimental analysis was carried out on a spam filtering task, using the well known and widespread open source anti-spam filter *SpamAssassin* [10].

In Sect. 2 we survey related works. The method to assess the robustness is explained in Sect. 4.1. The experiments are reported in Sects. 4 and 5.

2 Related Works

MCSs are currently being used in several adversarial classification tasks, like multimodal biometric systems [1,4], intrusion detection in computer systems [2], and spam filtering [11,12,13]. Two practical reasons are that in many of such tasks several heterogeneous feature subsets are available, and they can be easily exploited in a MCS architecture where each individual classifier is trained on a different feature subset [1]; moreover, in tasks like intrusion detection it is often necessary to face never-seen-before attacks, which can be easily done with a MCS architecture by adding new classifiers. For instance, MCSs are used in commercial spam filters which use heterogeneous information sources to label incoming e-mails (like the e-mail's text, its header, the attached images if any, etc.). Another motivation was proposed by several authors: using many classifiers would improve robustness because it would require the adversary to evade all the individual classifiers, or at least more than one of them, to evade the whole system [4,2,11]. However, we point out that this motivation is only based on intuition, and its validity has never been evaluated. Accordingly, understanding whether and how MCSs actually allow to improve robustness is still an open question.

In our past works we addressed this problem under several viewpoints. Since adding new detection rules to a system in response to new attacks is a common practice in spam filtering and in intrusion detection in computer systems, in [14] we investigated whether adding classifiers to a given ensemble can improve its robustness. In [15] we analysed a randomization strategy based on MCSs to improve robustness by preventing an adversary from gaining too much knowledge on a classifier. However in these works we used an analytical model for adversarial classification problems proposed in [5], which is based on unrealistic assumptions, and thus our results could not be exploited to devise practical

methods to design robust classifiers. In [16] we provided an empirical evidence[1] that a MCS architecture can be more robust than a single classifier architecture. This work was however limited to a non-standard MCS architecture (a logic OR of Boolean outputs of individual classifiers).

Among the few practical methods proposed so far in the literature for improving classifier robustness, the ones in [6,7] are particularly interesting since they are not limited to a specific task. In [7] it was pointed out that assessing the performance of a classifier during its design phase is likely to provide an *optimistic* estimate of the performance at operation phase, since training samples cannot contain any attack targeted to the classifier under design. In particular, if some features have a high discriminant capability on training samples and the adversary is aware of that, he could manipulate his samples at operation phase to modify the values of such features. The consequence is that their discriminant capability becomes lower at operation phase. For instance, text classifiers based on the bag of words feature model (in which terms occurrence or frequency are used as features) are widely used in spam filters. However spammers can guess the most discriminant terms that characterize spam and legitimate emails, and indeed two real and widely used attack strategies consist in camouflaging "spammy" terms (e.g., by misspelling them) to avoid their detection, and in adding "legitimate" terms not related to the spam message. The strategy proposed in [7] to improve robustness is hence to avoid to give features too much or too little emphasis during classifier training, to protect the classifier against such kinds of attacks. Several versions of this strategy were devised for linear classifiers, resulting in learning algorithms whose goal is a trade-off between attaining a high training accuracy and forcing the feature weights to be as much uniform as possible. A seemingly different strategy was proposed in [6], where a SVM-like learning algorithm was developed to make a linear classifier based on Boolean features robust against the modification of the most important features. Interestingly, it turns out that the effect of the proposed algorithm is similar to the strategy of [7], namely it results in producing more uniform weight values.

3 Motivations of This Work

The strategies in [6,7] are based on the intuition that, to improve robustness against attacks based on some knowledge on the relevance of each feature in the classifier's decision function, it can be useful to prevent the learning algorithm to overemphasize or underemphasize features. In the case of linear classifiers this strategy can be implemented by forcing the feature weights to be as uniform as possible. In this paper we investigate whether this effect can be obtained by some known MCS construction methods. We focus in particular on the well known bagging and RSM. In RSM, base classifiers are trained on randomly chosen feature subsets. Thus each feature may not be used by some base classifiers. In the particular case of linear classifiers, we can say that the most discriminant

[1] Reported analytical results turned out to be wrong (see the Erratum at the end of these proceedings).

features (on the training set) have a zero weight when they are not used, so their average weight across base classifiers could be lower than in a classifier trained on the whole feature set. Analogously, the average weight of less discriminant features could be higher in average, since their importance can be higher if they are used in base classifiers where the most discriminant features do not appear. When bagging is used, the training set of each base classifier is a bootstrap replicate of the original training set. Therefore, each training sample could not appear in some bootstrap replicates. One of the possible effects could be a reduction of the average weight of most discriminant features and an increase of the average weights of less discriminant ones. As mentioned above, our goal is to experimentally investigate whether and how this happens, and whether this results in a higher robustness with respect to an individual classifier, against attacks based on some knowledge on the feature discriminant capability. We focus in particular on a linear combination of linear classifiers, which can be easily analysed. Indeed the discriminant function of such a linearly combined set of classifiers can be written as:

$$g(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K}\left(\sum_{i=1}^{n} w_i^k x_i + w_0^k\right) = \sum_{i=1}^{n} w_i^{\text{avg}} x_i + w_0^{\text{avg}} \tag{1}$$

which is still a linear discriminant function in feature space, where K is the number of base classifiers, n the number of features of each base classifier and $w_0^{\text{avg}} \dots w_n^{\text{avg}}$ are the weights assigned by the MCS to the input features, each one computed by averaging the correspondent K weights of the base classifiers. Therefore, our goal is to understand whether the weights $w_0^{\text{avg}} \dots w_n^{\text{avg}}$ obtained by bagging and by RSM are more uniform than the weights of a single linear classifier trained on the whole feature set and on all the available training samples, and whether this results in a higher robustness under attack with respect to an individual classifier.

4 Experimental Setup

We first describe the method used in this paper to evaluate the robustness of a classifier, and then the data set and the classifiers used in the experiments.

4.1 A Method to Assess the Robustness of Classifiers under Attack

Ideally, the robustness of a classifier against a given attack should be evaluated on samples corresponding to such attack. However training samples cannot contain attacks devised against the classifier which is being designed, as pointed out in [7], and it could be very difficult to construct real attacks. Accordingly, we are currently developing a methodology to evaluate classifier's robustness, whose basic idea is to evaluate the accuracy of a classifier on *artificial* samples obtained by modifying the feature vectors of the available malicious testing samples with the aim of *simulating* the effect of a given attack of interest. Our methodology

is an extension of the method used in [7] to evaluate the proposed strategies to improve robustness (see Sect. 2).

The modifications to feature vectors of testing samples can be made with different criteria, depending on the specific application, the classifier and the attack to simulate. Nevertheless, in general it is useful to evaluate robustness under attacks of different strength [17,6,7]. The attack strength on a given sample can be measured as the distance in the feature space between the original sample and the one camouflaged by the attacker. We point out that the distance in feature space was used in [5,18] to measure the adversary's effort in modifying a given sample. The underlying rationale is that the more modifications an adversary makes to a sample, the more is the required effort. In particular, in the case of binary features a straightforward distance measure is the Hamming distance, which amounts to the number of features modified to "camouflage" a malicious sample.

One of the conditions under which it can be interesting to evaluate robustness is a "worst case" attack, in which the adversary is assumed to exactly know the classifier's decision function. Let us denote with $A(\mathbf{x})$ the modification of a feature vector $\mathbf{x}$, with m the maximum distance in the feature space between the original and modified feature vector $\mathbf{x}'$ according to a given metric $d(\cdot,\cdot)$ (namely, the maximum attack strength), and with $g(\mathbf{x})$ the discriminant function of the considered classifier, with the convention that the decision function is $f(\mathbf{x}) = \operatorname{sign} g(\mathbf{x}) \in \{-1,+1\}$, and that $+1$ and -1 are the labels respectively of the malicious and legitimate class. For any given malicious pattern $\mathbf{x}$, the worst case attack is the one which modifies $\mathbf{x}$ to the pattern $\mathbf{x}'$ which decreases most the value of the discriminant function $g(\mathbf{x}')$, under the constraint $d(\mathbf{x},\mathbf{x}') \leq m$. This can be written as the solution of a constrained optimization problem:

$$A(\mathbf{x}) = \operatorname{argmin}_{\mathbf{x}'} g(\mathbf{x}'), \text{ s.t. } d(\mathbf{x},\mathbf{x}') \leq m \ . \tag{2}$$

In the case when the features are Boolean (and thus m corresponds to the maximum number of features which can be modified), $d(\cdot,\cdot)$ is the Hamming distance and the discriminant function is linear ($g(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i + w_0$), it is easy to see that the solution can be found as follows. First the absolute values of the weights $|w_1|,\ldots,|w_n|$ must be sorted in decreasing order. Then the features must be considered in that order, and the values of up to m of them must be switched either from 1 to 0, if the corresponding weight is positive, or from 0 to 1, if the weight is negative.

4.2 Data Set and Base Classifiers

Our experiments were carried out on a spam filtering task. We used the TREC 2007 e-mail corpus, publicly available at `http://plg.uwaterloo.ca/~gvcormac/treccorpus07` and made up of 75,419 real e-mails (25,220 legitimate and 50,199 spam messages). We considered the first, second and third sequence of 10,000 e-mails (in chronological order) denoted in the following as D_1, D_2 and D_3, to build the training and testing sets as described below. The performance measure adopted to evaluate accuracy and robustness is the portion of the area under the ROC

curve corresponding to false positive (FP) error rates between 0 and 10%, denoted as $AUC_{10\%}$. Since the area under the whole ROC curve takes on values in $[0, 1]$, $AUC_{10\%}$ takes on values in $[0, 0.1]$. This measure was suggested in [7], to take into account the fact that in adversarial classification tasks (and especially in security applications) FP errors are typically much more harmful than false negative ones, and thus the classifier's operating point is required to have a low FP rate. We carried out two sets of experiments, with two different linear classifiers used in spam filtering tasks.

Experiment 1. In the first set of experiments we used two text classification algorithms proposed in the spam filtering literature, namely, support vector machine (SVM) [19] and logistic regression (LR) [7]. The e-mails in D_1 and D_2 were used to build respectively the training and testing set. We used the *bag of words* feature model, which is a common choice in text classification and spam filtering tasks. We first constructed a *vocabulary*, namely the set of distinct terms appearing in training e-mails (to this aim we used the anti-spam filter SpamAssassin [10], see below), which turned out to be 366,709. Each training and testing e-mail was then represented as a feature vector of the same size, in which each component was associated to a vocabulary term, and was set to 1 if that term occurred in the e-mail, to zero otherwise. The LR classifier was trained by maximising the classification accuracy through a gradient descent algorithm. Since the original feature set was too large for SVMs, we selected 20,000 features from training e-mails using the information gain criteria. The C parameter of the SVM learning algorithm was chosen among the values $\{0.001, 0.01, 0.1, 1, 10, 100\}$ by maximising the $AUC_{10\%}$ through a 5-fold cross validation on the training set. In the experiments we compared a single classifier built using LR and SVM, and MCSs built using bagging and RSM, for different ensemble sizes (3, 5 and 10), different fractions of randomly selected features from the original feature set for RSM (20%, 50%, 80%), and different training set sizes for bagging (20% and 100% of the original one). The robustness was then evaluated using the method described in Sect. 4.1 on the testing set.

Experiment 2. The second set of experiments was carried out using the popular and widespread open source anti-spam filter *SpamAssassin* [10] (version 3.2.5). It is made up of some hundred Boolean "tests" on the e-mail content, each one aimed at detecting a particular characteristic of spam or legitimate e-mails (for instance the presence of a typical spam word, or a known e-mail's header malformation indicating that it has been forged by an automatic software, as typically done by spammers [20]). Denoting with $\mathbf{x} = (x_1, \ldots, x_n) \in \{0, 1\}^n$ the tests' outputs, the decision function of the whole filter can be written as $f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} w_i x_i + w_0\right)$ where $\{w_0, \ldots, w_n\} \in \mathbb{R}^{n+1}$ is a set of weight values, and $\text{sign}(t) = +1$ if $t \geq 0$, -1 elsewhere. An e-mail is classified as spam (legitimate) if $f(\mathbf{x}) = +1$ ($f(\mathbf{x}) = -1$). In our experiments, we used only the tests whose value was not zero for at least one e-mail of the data set, which turned out to be 549. Default weight values $w_0, \ldots, w_n$ are provided by SpamAssassin developers[2]. These values were derived by manually adjusting the

[2] `http://spamassassin.apache.org/tests_3_2_x.html`

values obtained by a perceptron trained over a huge amount of data, with the aim of increasing the robustness of the spam filter. Note that default weights of tests associated to characteristics of spam (legitimate) e-mails are positive (negative). We used SpamAssassin as a linear classifier, and carried out the same experiments described above for text classifiers. Namely, we computed the weight values using an individual LR and SVM classifier, and a MCS obtained using bagging and RSM, on the same base classifiers. These classifiers were also compared to the individual classifier with the default weight values, namely the standard SpamAssassin. We point out that some SpamAssassin tests (features) are based on the outputs of a text classifier, which must be previously trained. We trained it using the D_1 e-mail subset. To compute the weights with the LR and SVM classifiers we used the e-mails of the D_2 subset as training set. The LR an SVM classifiers were trained as in the previous experiments. Robustness was evaluated on the e-mails in D_3. Results are reported in Sect. 5.

5 Experimental Results

The results of our experiments are reported in terms of $AUC_{10\%}$ as a function of the attack strength m. Since the features are Boolean, we measured the attack strength m as the maximum number of features which can be modified. According to this measure, a classifier is more robust than another if it exhibits a higher $AUC_{10\%}$ value for the same value of m. Note that the true positive (TP) classification rate of a classifier decreases as the attack strength m increases, and so does the $AUC_{10\%}$ value. As a limit case, beyond some m value all the modified malicious testing samples are labelled as legitimate. Consequently, the TP rate equals zero for any FP value and the corresponding $AUC_{10\%}$ equals zero as well. We also report a measure of the evenness of the feature weights assigned by the classifiers, which was proposed in [7], defined as the ratio of the sum of the top K absolute weight values to the sum of all the n absolute weight values, for $K = 1, \ldots, n$:

$$F(K) = \left(\sum_{i=1}^{K} |w_{(i)}| \right) / \left(\sum_{i=1}^{n} |w_{(i)}| \right) \tag{3}$$

where $|w_{(1)}|, \ldots, |w_{(n)}|$ are the absolute weight values sorted in decreasing order. The weight w_0 is disregarded since it does not affect the $AUC_{10\%}$ value. When all weights are equal, $F(K) = K$, while as the weight distribution become uneven, $F(K)$ approaches 1 for any K value.

Experiment 1. The results obtained with text classifiers are shown in Fig. 1. We found that the robustness of MCSs significantly increased for increasing ensemble size with RSM (especially for a low feature subset size), while no significant changes were observed with bagging. This result provides some support the one reported by the authors in [14]. Due to lack of space, only the results for the largest considered ensemble size (10) are reported. Similarly, the robustness of the RSM method significantly increased as the size of feature subsets increased, while no significant difference was observed with bagging for different training set

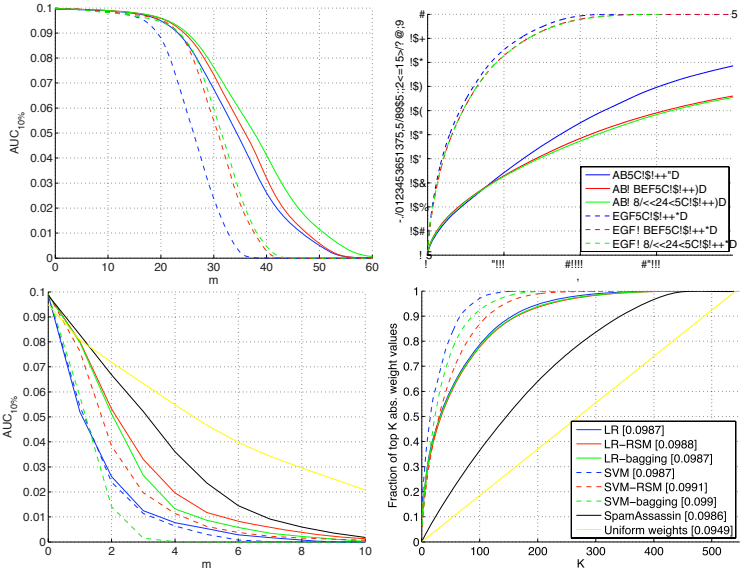


Fig. 1. $AUC_{10\%}(m)$ (left) and $F(K)$ (right) for single LR and SVM classifiers, and for bagging and RSM with LR and SVM as base classifiers. Top: text classifiers (exp. 1); bottom: SpamAssassin (exp. 2). For the latter, results obtained with default and uniform weights are also reported. The $AUC_{10\%}$ value at $m = 0$ is reported also in the legend. Results for the MCSs are averaged over 5 runs (standard deviation is not reported since it is negligible). Note that the $AUC_{10\%}$ drops to zero when the attack strength becomes so high that all the modified malicious testing samples are labelled as legitimate, and thus the TP rate equals zero for any FP value.

sizes. Due to lack of space, only results corresponding to the highest considered feature subset size in RSM (80%) and training set size in bagging (100%) are reported. Let us compare bagging and RSM with a single classifier. First, Fig. 1 shows for $m = 0$ (namely, on the original testing set) the $AUC_{10\%}$ value of bagging and RSM was nearly the same as the one of the corresponding individual classifiers trained on the original training set. This means that bagging and RSM did not improve the performance of the individual classifiers. However, for $m > 0$ (namely, when the classifiers are under attack) both bagging and RSM allowed to obtain more even weight distributions with respect to the individual classifiers, and also exhibited a higher robustness. This result supports our main hypothesis, namely that bagging and RSM result in more uniform weight values than a single classifier (Sect. 3). This result is interesting also because it shows that although MCSs did not improve classification accuracy with respect to individual classifiers, they nevertheless allowed to improve robustness. It can also be noted that in this case using a MCS did not increase the computational cost at classification phase, since the discriminant function of the MCS was linear as the one of the single classifier.

Experiment 2. The results obtained with the SpamAssassin filter as a linear classifier are reported in Fig. 1. On the basis of the results of the previous experiments, we considered only ensembles of 10 base classifiers, and feature subsets of 80% of the whole feature set for RSM. We also report the results obtained by SpamAssassin as a single classifier, both using its default weight values and using uniform weight values, equal to either $+1$ (for features associated to characteristics of spam e-mails, see above) or to -1. As expected, SpamAssassin with uniform weights attains the highest robustness (see Fig. 1, bottom). However, it does not exploit any information about the discriminant capability of the features, and therefore it exhibits the worst performance on the original testing set without attacks ($AUC_{10\%}$ for $m = 0$). An interesting result is that weights obtained by the LR and SVM classifiers slightly outperformed the default SpamAssassin weights for $m = 0$, while the default values exhibited a higher robustness for $m > 0$, namely when the classifiers were under attack. This provides evidence that the default weights, which were manually set by SpamAssassin developers, are actually capable to improve its robustness under attack. In particular, we observed that the LR and SVM learning algorithms assigned higher weights to the most discriminant features (on training samples), which turned out to be the ones related to the text classifier included in SpamAssassin. This guaranteed to achieve higher performances when the filter was not under attack, but also undermined its robustness under attack. The corresponding default weight values were indeed lower than the ones assigned by the LR and SVM classifiers. As in the previous experiments, Fig. 1 also shows that the use of MCSs did not improve the classification accuracy when the classifiers were not under attack, but it allowed to improve the robustness when the classifiers were under attack, with the only exception of bagging with the SVM base classifier. It is interesting to note that the default SpamAssassin weights exhibited a higher robustness than bagging and RSM, besides than single classifiers.

6 Conclusions

While MCSs are mainly used to improve classification accuracy, inspired by [6,7] we argued that methods like bagging and RSM could also result in improving robustness in adversarial classification tasks against attacks based on some knowledge of the classifier's discriminant function, due to a potential side effect consisting in giving more uniform weights to the features with respect to a single classifier. Reported experiments performed on a real spam filtering task, using text classifiers and a real spam filter, provided evidence that this intuition is correct. In particular, our experiments showed that, even in cases when bagging and RSM did not improve the performance of a single classifier when they are not under attack, they turned out to be significantly more robust under attack. These results provide a first sound motivation to the application of MCSs in adversarial classification tasks, other than the intuitive and qualitative considerations that have motivated their use so far, and thus open a new and relevant area of research for MCSs.

References

1. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Trans. Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
2. Perdisci, R., Gu, G., Lee, W.: Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In: Int. Conf. Data Mining (ICDM), pp. 488–498. IEEE Computer Society, Los Alamitos (2006)
3. Giacinto, G., Roli, F., Didaci, L.: Fusion of multiple classifiers for intrusion detection in computer networks. Patt. Rec. Lett. 24, 1795–1803 (2003)
4. Ross, A.A., Nandakumar, K., Jain, A.K.: Handbook of Multibiometrics. Springer, Heidelberg (2006)
5. Dalvi, N., Domingos, P., Mausam, Sanghai, S., Verma, D.: Adversarial classification. In: ACM Int. Conf. Knowledge Discovery and Data Mining, pp. 99–108 (2004)
6. Globerson, A., Roweis, S.T.: Nightmare at test time: robust learning by feature deletion. In: Int. Conf. Machine Learning, pp. 353–360 (2006)
7. Kolcz, A., Teo, C.H.: Feature weighting for improved classifier robustness. In: Sixth Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA (2009)
8. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
9. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
10. The Apache SpamAssassin Project, `http://spamassassin.apache.org/`
11. Hershkop, S., Stolfo, S.J.: Combining email models for false positive reduction. In: Proc. ACM Int. Conf. Knowledge Discovery in Data Mining, pp. 98–107. ACM, New York (2005)
12. Lynam, T.R., Cormack, G.V., Cheriton, D.R.: On-line spam filter fusion. In: Proc. Int. ACM SIGIR Conf. on Res. and Dev. Inf. Retr., pp. 123–130. ACM, New York (2006)
13. Tran, T., Tsai, P., Jan, T.: An adjustable combination of linear regression and modified probabilistic neural network for anti-spam filtering. In: Int. Conf. Patt. Rec., pp. 1–4 (2008)
14. Biggio, B., Fumera, G., Roli, F.: Evade hard multiple classifier systems. In: Okun, O., Valentini, G. (eds.) Supervised and Unsupervised Ensemble Methods and their Applications. Studies in Comp. Int., vol. 245, pp. 15–38. Springer, Heidelberg (2009)
15. Biggio, B., Fumera, G., Roli, F.: Adversarial pattern classification using multiple classifiers and randomisation. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 500–509. Springer, Heidelberg (2008)
16. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems for adversarial classification tasks. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 132–141. Springer, Heidelberg (2009)
17. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Proc. 2006 ACM Symp. Information, computer and communications security (ASIACCS 2006), pp. 16–25. ACM, New York (2006)
18. Lowd, D., Meek, C.: Adversarial learning. In: Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, pp. 641–647 (2005)
19. Drucker, H., Wu, D., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Trans. Neural Networks 10(5), 1048–1054 (1999)
20. Stern, H.: A survey of modern spam tools. In: 5th Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA (2008)

SOCIAL: Self-Organizing ClassIfier ensemble for Adversarial Learning

Francesco Gargiulo and Carlo Sansone

Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II,
Via Claudio, 21 I-80125 Napoli, Italy
{francesco.grg,carlosan}@unina.it

Abstract. Pattern recognition techniques are often used in environments (called *adversarial environments*) where adversaries can consciously act to limit or prevent accurate recognition performance. This can be obtained, for example, by changing labels of training data in a malicious way.

While Multiple Classifier Systems (MCS) are currently used in several security applications, like intrusion detection in computer networks and spam filtering, there are very few MCS proposals that explicitly address the problem of learning in adversarial environments. In this paper we propose a general algorithm based on a multiple classifier approach to find out and clean mislabeled training samples. We will report several experiments to verify the robustness of the proposed approach to the presence of possible mislabeled samples. In particular, we will show that the performance obtained with a simple classifier trained on the training set "cleaned" by our algorithm is comparable and even better than those obtained by some state-of-the-art MCS trained on the original datasets.

1 Introduction

Pattern recognition techniques are often used in environments where adversaries can consciously act to limit or prevent accurate performance from a classification point of view (*adversarial environments*). A classical example is spam filtering where spammers tailor messages to avoid the most recent spam detection techniques. Further examples of adversarial environments arise in the field of computer security where there is an escalating competition between detection and evasion techniques for various types of attacks. In general, in fact, one can expect that whenever pattern recognition technique is used to provide protection from illegal activities, adversaries will deliberately attempt to circumvent these defences.

An interesting paper in the context of adversarial learning is the one by Barreno et al. [2]. One of the open questions posed by the authors is about what defenses exist against adversaries manipulating (attacking) learning systems. In this paper we try to give an answer by means of a multiple classifier approach. Multiple Classifier Systems (MCS) are currently used in several security applications like intrusion detection in computer networks [6] and spam filtering [3,9,11]. However, there are very few MCS proposals that explicitly address the problem of learning in adversarial environments. One of this

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 84–93, 2010.

can be found in the papers by Biggio et al. [4,5]. Although they afford the adversarial classification in a quite general way, the proposed solutions seem tailored to the spam filtering problem.

In adversarial environments mislabeled samples can be forced by malicious users that are able to perform a corruption of the training data labels (this kind of attack is also known as *poisoning attack* [1]). An example arises in the problem of identification of HTTP traffic flowing through port TCP 80. In this case the ground truth of the training data is typically only based on the port number [10]. So, it is possible that some training labels are incorrect since a malicious user can have performed a sort of *port-spoofing* attack, changing the actual port number of a flow.

In this paper we propose a general MCS approach to find out and clean mislabeled training samples. The proposed algorithm, by suitably combining the output of an ensemble of base classifiers whose weights are dynamically updated, iteratively changes the labels assigned to training samples and considers the training set *cleaned* when the amount of changes becomes stable across successive iterations.

We will describe several experiments to verify the robustness of the proposed approach to the label corruption problem. In particular, we will show that the performance obtained with a simple classifier trained with the "cleaned" training set is comparable and even better than those obtained by some state-of-the-art MCS trained on the original (i.e., corrupted) datasets.

The rest of the paper is organized as follows: in Section 2 some related work are reviewed. In Section 3 the proposed approach is presented, while experimental results are described in Section 4. Finally, some conclusions are drawn in Section 5.

2 Related Work

As stated in the Introduction, there are very few proposals in the open literature about the use of an MCS approach to deal with adversarial learning problems. On the other hand, the more general problem of learning when some samples were randomly assigned the wrong class label (also known as *label noise* problem) has been afforded in the past in the MCS context.

AdaBoost [8] has shown to often improve the base learner accuracy. Since its introduction, it has been successfully applied to many problems. Although its wide-spread success, it is susceptible to overfitting when noisy samples are present, as pointed out by Dietterich [7]. In fact, when training samples are noisy and therefore difficult to fit, AdaBoost increases the weights of those samples and overfit them because many consecutive base models may not learn them properly. Oza [14] proposed an approach, called AveBoost2, to overcome this problem that can be seen as a relaxed version of AdaBoost. The AveBoost2 averaging mechanism does not allow the weights of noisy samples to increase rapidly, thereby mitigating the overfitting problem. An experimental comparison of AveBoost2 with AdaBoost on some UCI datasets with a 10% of label noise added to the original data demonstrated the better robustness of the former algorithm to label noise.

Melville and Mooney [13] introduced a new kind of MCS, called DECORATE, to take into account the label noise problem. DECORATE (Diverse Ensemble Creation by Oppositional Relabelling of Artificial Training samples) uses an existing "strong" learner to build an effective diverse committee in a fairly simple, straightforward manner. This is accomplished by adding different randomly constructed samples to the training set when building new committee members. These artificially constructed samples are given labels that disagree with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Thiel [17] made a comparison between single classifiers and an ensemble when noisy labels are present in the training data. Another point addressed in this paper is if classifiers trained on *soft* labels are more resilient to label noise than those trained on hard labels. Several MCS schemes were considered: Minimum, Median, Average, Dempster-Shafer, orthogonal sum rule, Decision Templates, simple probabilistic product, and an optimal least squares solution calculated using the pseudoinverse. The main finding of this study is that the ensemble typically improves the overall accuracy in presence of label noise, but the right MCS architecture has to be properly selected. Moreover, by comparing classifiers trained on soft versus hard labeled data, it turned out that the soft approach is more resistant to label noise.

3 The Proposed Approach

SOCIAL is the acronym of ***S**elf-**O**rganizing **C**lass**I**fier ensemble for **A**dversarial **L**earning*. It is specifically designed to deal with training sets having corrupted labels as the ones that occur in *adversarial leaning* problems. The principle behind it is that a community through a democratic approach can remove most of its own initial mistakes and then can improve its behavior.

SOCIAL is based on a multiple classifier approach with a parallel topology. In particular, by looking at the results provided by the ensemble, SOCIAL evaluates and dynamically updates a reliability for each base classifier. By suitably combining the outputs of the base classifiers using these reliability values, it is possible to detect and clean mislabeled data in the training set. The algorithm, in fact, after an iterative evolution, returns a *cleaned* training set. This result is obtained by iteratively changing the labels assigned to the samples and considering the training set *cleaned* when the amount of changes becomes stable according to a suitably defined criterion.

3.1 The SOCIAL Algorithm

The main parameter used by SOCIAL is the *Degree of Truth*, hereinafter denoted with **DoT**[1]. A DoT value, ranging in $[0, 1]$, is assigned to each sample; it represents the probability that the label assigned to the sample is correct.

[1] The concept of *DoT* is often used in the context of *fuzzy theory* [15]. In this case statements are described in terms of membership functions, that are continuous and have values belonging to the range $[0, 1]$. For example, given the measured value of a parameter, the membership function gives the *degree of truth* that the parameter is "high" or "low".

The algorithm is composed by two phases: a *bootstrap* step ($t = 1$) and an iterative evolution ($t > 1$). In the bootstrap step the DoT distribution requires to be initialized. Making the assumption that the noise distribution is unknown, a possible criteria is to assign a value equal to 1 to the DoT of each sample. This means that we initially trust the labels assigned to the training set samples.

Since the second phase of the algorithm provides for an iterative evolution, another parameter to be considered is δ_{dB}, that is the value used for the terminal condition of the algorithm. This value is calculated as the ratio (in decibel) between the number of the samples that change their label across two consecutive steps of the algorithm.

Fig. 1 shows the pseudocode of the algorithm. SOCIAL has as input the training set $T = (x_1, y_1(1), \ldots, (x_N, y_N(1))$, the base model learning algorithms $L_1, \ldots, L_B$ and a threshold value τ for the terminal condition. So, $y_i(1)$ represents the initial label (possibly corrupted) of the sample x_i.

The algorithm, for each step t, maintains a distribution $DoT(t)$, where each element $DoT_i(t)$ is associated to the sample x_i. This distribution gives the probability that the sample x_i really belongs to the class y_i. As stated earlier, in the bootstrap step $DoT_i(1) = 1 \quad \forall i \in 1, 2, \ldots, N$.

For each step t, as first operation SOCIAL builds, through a k-fold cross validation approach, a function $h^b(t)$ that associates a predicted class $\hat{y}_i^b(t)$ to each sample x_i, for each base classifier b ($b \in 1, \ldots, B$). More in details, the training set T is randomly partitioned into F disjoint subsets $V_1, \ldots, V_F$ (called folds) of equal size. The samples of each V_j are then classified using $T - V_j$ as the training set, so obtaining the predicted class labels $\hat{y}_i^b(t)$.

Starting from $\hat{y}_i^b(t)$, SOCIAL evaluates the $\mathbf{WCM}$ (see Tab. 1), where each entry is calculated as shown in Eq. 1. Here N_i represents the number of samples belonging to class C_i. Note that WCM differs from a standard *confusion matrix* since each classified (correctly or not) sample x_k is weighted by its DoT_k.

Table 1. Weighted Confusion Matrix ($\mathbf{WCM}$) for M-classes classification

True Class	Predicted Class $\hat{C}_1$	$\hat{C}_2$	...	$\hat{C}_M$
C_1	e_{11}	e_{12}	...	e_{1M}
C_2	e_{21}	e_{22}	...	e_{2M}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_M	e_{M1}	e_{M2}	...	e_{MM}

$$e_{ij}^b(t) = \frac{\sum_{k=1:y_k(t)=C_i \, and \, \hat{y}_k^b(t)=C_j}^{N} DoT_k(t)}{N_i} \tag{1}$$

Successively, the algorithm evaluates the Classifier Reliability $\mathbf{R}^b(t)$ for each base classifier b. This is a vector whose values $R^b(C_i, t)$ provide, for each predicted class C_i, a degree of belief on the correctness of the classifier's guess. These values are obtained

by comparing the response of the classifier to the output of the ensemble. In particular, they are calculated starting from the **WCM** in this way:

$$R^b(C_i, t) = e^b_{ii}(t) \tag{2}$$

After the Classifier Reliability evaluation, the algorithm combines the output of the base classifiers by means of a *Weighted Majority Voting* approach. In this way it is possible to update the label $y_i(t)$ of each training set sample as well as the $DoT(t)$ distribution:

$$y_i(t+1) = argmax_j \frac{\sum_{k=1:\hat{y}^k_i(t)=C_j}^{B} R^k(\hat{y}^k_i(t))}{B}$$

$$DoT_i(t+1) = \max_j \frac{\sum_{k=1:\hat{y}^k_i(t)=C_j}^{B} R^k(\hat{y}^k_i(t))}{B}$$

This provides a label for each sample (possibly different from the one obtained in the previous step) with the associated DoT. At this point SOCIAL evaluates the number of samples that change their label:

$$changes(t) = \sum_{i=1}^{N} \Delta_i, \qquad \Delta_i = \begin{cases} 1, & if\ y_i(t-1) \neq y_i(t) \\ 0, & otherwise \end{cases} \tag{3}$$

Finally, the terminal condition is evaluated, i.e. if

$$10 * \log_{10} \frac{changes(t-1)}{change(t)} < \tau.$$

When such a condition is reached, SOCIAL ends and returns the *cleaned* training set $(x_1, y_1(t)), \ldots, (x_N, y_N(t))$.

4 Experimental Results

In order to assess the algorithm performance, we will show the results obtained on data produced by some synthetic distributions in which label noise is added according to a model described in the following.

In order to set-up the SOCIAL architecture we have considered three base classifiers: a *Decision Tree* (DT), a *Probabilistic Neural Network* [16] (PNN) and a *K-Nearest Neighbors* (KNN) with $K = 3$. We chose these classifiers since they are very different each other and this can favour the *diversity* among them. It was in fact experimentally demonstrated that the diversity of the base classifiers is very important to increase the overall performance of a Multiple Classifier System [12].

In all the tests we will compare the accuracy obtained by the worst base classifier trained with the training set *cleaned* by SOCIAL and the accuracy obtained by all the base classifiers and by some state-of-the-art Multiple Classifier Systems on the *original* (i.e. corrupted) training set. As comparing systems, we chose four well-known MCS architectures, i.e. *Bagging*, *AdaBoost*, *MultiBoost*, and *DECORATE*.

$SOCIAL((x_1, y_1(1)), \ldots, (x_N, y_N(1)), L_1, \ldots, L_B, \tau)$

$\triangleright$ N is the number of samples, B is the number of base classifiers,
$\triangleright$ τ is terminal condition threshold, M is the number of the classes.

Initialize $DoT_i(1) = 1 \quad \forall i \in 1, 2, \ldots N.$
Initialize $t = 0$
Initialize $\delta = +\infty$
do
$t = t + 1$
for $b = 1, 2, \ldots, B,$
$\triangleright$ Classifier performance evaluation through a K-fold Cross Validation Approach:
$h^b(t) = L_b((x_1, y_1(t)), \ldots, (x_M, y_M(t)))$
for $i = 1, 2, \ldots, M,$
for $j = 1, 2, \ldots, M,$
$\triangleright$ **WCM** entries calculation

$$e_{ij}^b(t) = \frac{\sum_{k=1:y_k(t)=C_i \ and \ \hat{y}_k^b(t)=C_j}^{N} DoT_k(t)}{N_i}$$

$\triangleright$ Classifier Reliability calculation:
$R^b(C_i, t) = e_{ii}^b(t)$
for $i = 1, 2, \ldots, N,$
$\triangleright$ Label updating for each sample:

$$y_i(t+1) = argmax_j \frac{\sum_{k=1:\hat{y}_i^k(t)=C_j}^{B} R^k(\hat{y}_i^k(t))}{B}$$

$\triangleright$ DoT updating for each sample:

$$DoT_i(t+1) = \max_j \frac{\sum_{k=1:\hat{y}_i^k(t)=C_j}^{B} R^k(\hat{y}_i^k(t))}{B}$$

$\triangleright$ Label changes evaluation:

$$changes(t) = \sum_{i=1}^{N} \Delta_i, \qquad \Delta_i = \begin{cases} 1, & if \ y_i(t-1) = y_i(t) \\ 0, & otherwise \end{cases}$$

$\triangleright$ Terminal condition evaluation:
if $t > 1,$
$\delta = 10 * \log_{10} \frac{changes(t-1)}{changes(t)}$
while $\delta > \tau$
return The *cleaned* training set $(x_1, y_1(t)), \ldots, (x_N, y_N(t))$.

Fig. 1. The SOCIAL Algorithm

To experimentally determine the impact of label noise on classification accuracy, we need to artificially add noise according to a certain model. In our case, when only two classes are present, a given portion of the training is randomly selected and the associated labels are changed. This method can be easily extended to the multi-class case, with new labels chosen once again in a random manner among the other classes.

Since we are considering synthetically generated data, all the results reported in the following have been obtained by averaging ten different runs on data generated according to the distributions reported in table 2. Training labels were instead corrupted according to the above described noise model.

Table 2. The three considered datasets

Distribution	Training Set Samples	Test Set Samples	Classes
Gaussian	4000	1000	2
Mixture of Three Gaussians	4000	1000	2
Rotated Check Board (45°)	4000	1000	2

The first test was made by using the three synthetic distributions described in table 2 by considering a 30% of label noise.

Table 3. Results obtained with a 30% of label noise on the training set

	Gaussian	Mixture of Gaussians	Rotated Check Board
Multiple Classifiers Systems			
AdaBoost	95.70%	80.60%	74.10%
MultiBoost	95.70%	78.60%	72.30%
DECORATE	96.60%	82.00%	84.20%
Bagging	96.40%	79.60%	89.40%
Base Classifiers			
DT	80.40%	68.40%	70.00%
KNN	86.30%	71.00%	71.00%
PNN	92.60%	75.80%	75.80%
DT trained with the *cleaned* training set			
DT	97, 10%	82, 40%	91, 00%

In table 3 the first four rows report the accuracy obtained with Multiple Classifier Systems trained with the original training set corrupted by a 30% of label noise. The following three rows report the accuracy obtained with the three base classifiers of the SOCIAL algorithm trained on the original training set and, finally, the last row reports the accuracy obtained by the worst base classifier (DT, in this case), trained with the training set cleaned by SOCIAL. It is worth noting that SOCIAL makes the classification problem simpler than the original one, and even the worst classifier trained on the *cleaned* training set becomes better, in terms of accuracy, with respect to all the base classifiers and all the other MCSs approaches.

In figure 2 it is shown how SOCIAL modifies the training set, and how the δ parameter changes, in case of the *Rotated Check Board* dataset. In particular, the graph on the left of the first row represents the behavior of the accuracy across the different steps of the algorithm. It can be seen that after the first steps in which the accuracy improves (this corresponds to an *effective* cleaning of the training set), there is a second phase in which the accuracy decreases. This corresponds to an excessive *smoothing* of the original distribution. It is possible to find similar information in the graph on the right of the first row, in which the different values of δ_{dB} across the steps are plotted. In this case we are monitoring the variation of the number of label changes between

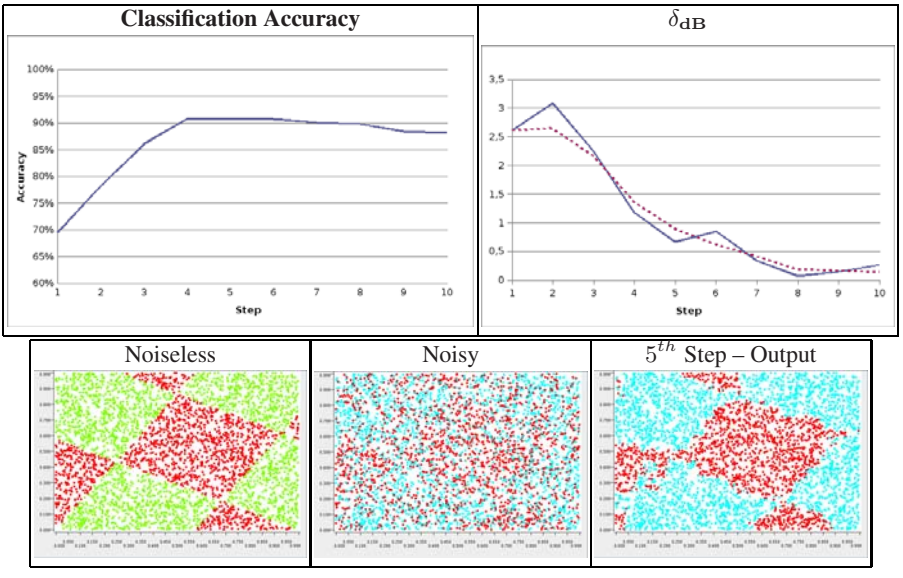


Fig. 2. Classification accuracies and δ_{dB} as a function of the step t of the algorithm, on the Rotated Check Board dataset with a 30% of label noise (first row). Noiseless data, the noisy training set and the output of the algorithm (second row).

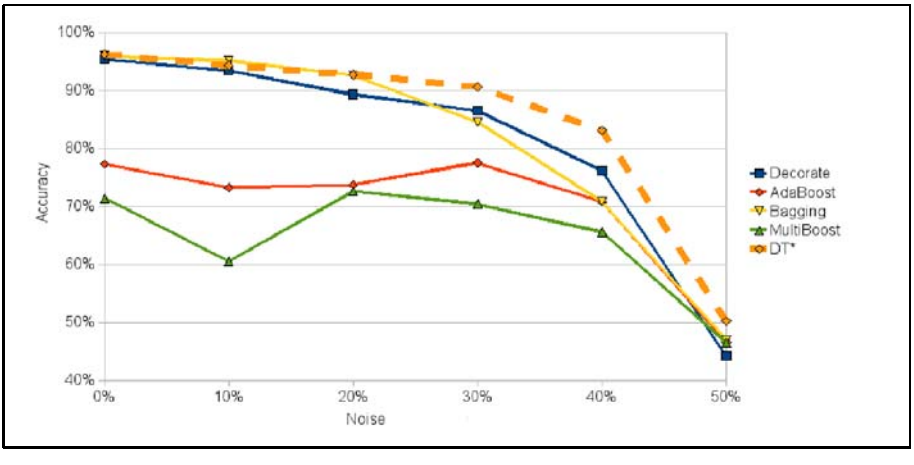


Fig. 3. Comparison of different MCSs trained on corrupted data and a Decision Tree (DT*) trained on cleaned data, as the % of noise varies.

two consecutive steps. The value is in dB, so if there are no variations between two consecutive steps we have a δ value equal to 0. We experimentally noticed that if we do not want to alter the initial distribution, a good value for the threshold τ is $1dB$. In this graph the dotted line represents an interpolation along three consecutive points of the δ_{dB} line (i.e the continuous one). The threshold τ is indeed applied to the dotted line in order to better absorb the variations of the original values of δ between two successive steps. This allows us to choose the right iteration for stopping the SOCIAL algorithm.

In order to evaluate the robustness of the SOCIAL approach to the noise level added to training set labels, we have considered different percentages of label noise and have evaluated the accuracy of the DT and the other MCS approaches under test, as reported in Figure 3. For the sake of brevity, we present results on the *Rotated Check Board* dataset only. Anyway, similar results have been obtained on the other datasets. The DT trained with the dataset cleaned by SOCIAL performs always significantly better than the DT trained with the original data. Moreover, as it can be noted from Figure 3, starting from a noise level of 30%, the DT trained with the dataset cleaned by SOCIAL exhibits the best performance among all the considered systems, giving rise to significant improvements in classification accuracy.

5 Conclusion

In this paper we proposed an algorithm named SOCIAL that tries to *clean* a training set from label noise by using a MCS approach. The algorithm is designed to work in an adversarial learning context, in which a malicious user tries to camouflage training samples to limit the performance of the classification system.

The proposed approach demonstrated its effectiveness on several synthetic datasets. The performance obtained with a simple classifier trained by using data "cleaned" by SOCIAL is comparable and even better than some state-of-the-art MCS trained on the original (i.e., corrupted) datasets.

As regards future works, we want to study the convergence of the algorithm as well as the influence of different base classifiers on its performance. Finally, we are interested in applying SOCIAL in real adversarial learning contexts.

References

1. Abad, C., Bonilla, R.: An analysis on the schemes for detecting and preventing ARP cache poisoning attacks. In: ICDCS Workshops, p. 60. IEEE Computer Society, Los Alamitos (2007)
2. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Lin, F.-C., Lee, D.-T., Lin, B.-S., Shieh, S., Jajodia, S. (eds.) ASIACCS, pp. 16–25. ACM, New York (2006)
3. Biggio, B., Fumera, G., Pillai, I., Roli, F.: Image spam filtering using visual information. In: Proc. of the 14th International Conf. on Image Analysis and Processing (ICIAP), pp. 105–110 (2007)
4. Biggio, B., Fumera, G., Roli, F.: Adversarial pattern classification using multiple classifiers and randomisation. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T.-Y., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSSPR 2008. LNCS, vol. 5342, pp. 500–509. Springer, Heidelberg (2008)

5. Biggio, B., Fumera, G., Roli, F.: Multiple classifier systems for adversarial classification tasks. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 132–141. Springer, Heidelberg (2009)
6. Corona, I., Giacinto, G., Mazzariello, C., Roli, F., Sansone, C.: Information fusion for computer security: State of the art and open issues. Information Fusion 10(4), 274–284 (2009)
7. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
8. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proc. 13th International Conference on Machine Learning, pp. 146–148. Morgan Kaufmann, San Francisco (1996)
9. Fumera, G., Pillai, I., Roli, F.: Spam filtering based on the analysis of text information embedded into images. Journal of Machine Learning Research 6, 2699–2720 (2006)
10. Gargiulo, F., Kuncheva, L.I., Sansone, C.: Network protocol verification by a classifier selection ensemble. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 314–323. Springer, Heidelberg (2009)
11. Gargiulo, F., Penta, A., Picariello, A., Sansone, C.: A personal antispam system based on a behaviour-knowledge space approach. In: Okun, O., Valentini, G. (eds.) Applications of Supervised and Unsupervised Ensemble Methods. Studies in Computational Intelligence, vol. 245, pp. 39–57. Springer, Heidelberg (2009)
12. Kuncheva, L.I.: Classifier ensembles for changing environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)
13. Melville, P., Mooney, R.J.: Diverse ensembles for active learning. In: Brodley, C.E. (ed.) ICML. ACM International Conference Proceeding Series, vol. 69. ACM, New York (2004)
14. Oza, N.C.: Aveboost2: Boosting for noisy data. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 31–40. Springer, Heidelberg (2004)
15. Smets, P., Magrez, P.: The measure of the degree of truth and the grade of membership. Fuzzy Sets Syst. 25(1), 67–72 (1988)
16. Specht, D.F.: Probabilistic neural networks. Neural Networks 3, 109–118 (1990)
17. Thiel, C.: Classification on soft labels is robust against label noise. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part I. LNCS (LNAI), vol. 5177, pp. 65–73. Springer, Heidelberg (2008)

Unsupervised Change-Detection in Retinal Images by a Multiple-Classifier Approach

Giulia Troglio[1,2], Marina Alberti[1], Jón Atli Benediksson[2], Gabriele Moser[1], Sebastiano Bruno Serpico[1], and Einar Stefánsson[2]

[1] University of Genoa, Dept. of Biophysical and Electronic Eng. (DIBE), Via Opera Pia 11a, 16145, Genoa, Italy
[2] University of Iceland, Faculty of Electrical and Computer Eng. and Dept. of Ophthalmology, P.O. Box Hjardarhagi 6, 107 Reykjavik, Iceland

Abstract. The aim of this work is the development of an unsupervised method for the detection of the changes that occurred in multitemporal digital images of the fundus of the human retina, in terms of white and red spots. The images are acquired from the same patient at different times by a fundus camera. The proposed method is an unsupervised multiple classifier approach, based on a minimum-error thresholding technique. This technique is applied to separate the "change" and the "no-change" areas in a suitably defined difference image. In particular, the thresholding approach is applied to selected sub-images: the outputs of the different windows are combined with a majority vote approach, in order to cope with local illumination differences. A quantitative assessment of the change detection performances suggests that the proposed method is able to provide accurate change maps, although possibly affected by misregistration errors or calibration/acquisition artifacts. The comparison between the results obtained using the implemented multiple classifier approach and a standard one points out that the proposed algorithm provides an accurate detection of the temporal changes.

Keywords: Multiple Classifiers, Change Detection, Bioimaging.

1 Introduction

The analysis of images of the human retina is an important diagnostic tool in ophthalmology [1]. Fundus images may be used to diagnose many diseases that affect the vascular system by revealing the changes that have occurred during the period between two consecutive visits. In the last years, an intensified effort has been undertaken in developing tools to assist in the diagnosis of diabetic retinopathy, which is the most common cause of blindness in the working-age population of developed countries [2,3,4,5,6].

Hipwell *et al.* adapted a technique, which was originally developed for fluorescein angiograms, and applied it to microaneurysm detection in digitally acquired red-free retinal images [7]. The system was tested on a large sample of images

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 94–103, 2010.

designed to mimic a screening scenario. Usher *et al.* developed a system to detect microaneurysm, haemorrhages and exudates in color retinal images [6]. The primary aim of their system was to detect any diabetic retinopathy in images from a diabetic screening population with the secondary aim at detecting all patients with sight threatening disease. The system was evaluated on a large sample of unselected digitally acquired low-resolution images from an existing Diabetic Retinopathy screening programme. Walter *et al.* proposed an algorithm focusing only on the detection of exudates with an approach based on morphological techniques [8]. Exudates are found by detecting high grey level variations, and the removal of the optic disc is indispensable to this approach. Their algorithm can be divided into two parts: First, they find candidate regions that possibly contain exudates; then, they apply morphological techniques in order to find the exact contours of these exudates.

The purpose of this work is the use of a multiple classifier approach for the automatic detection of changes in pathologies such as microaneurysm, hard exudates, cotton wool spots, hemorrhages, and edema in terms of white and red spots in color fundus images. The white spots represent hard and soft exudates, whereas the red spots represent hemorrhages or aneurysmata.

An automatic change detection technique is proposed for color fundus images, which is based on the unsupervised thresholding method proposed by Kittler and Illingworth (K&I) [9] and originally developed for computer vision purposes. The key idea of the method is to model the "change" and "no-change" pixels of a pair of multitemporal images (i.e., images taken of the same patient at different times, which capture temporal changes) by two Gaussian-distributed classes and to discriminate between such classes by applying a thresholding technique to a suitably defined "difference image." In particular, the K&I method allows the threshold-selection task to be formalized in an unsupervised framework as the minimization of a criterion function defined according to the Bayes decision theory. However, a direct application of K&I to the whole image may be severely affected by the spatial behavior of the overall illumination field at the two observation dates. In order to compensate for non-uniform illumination across each acquisition and variation of illumination between the two acquisitions to be compared, a multiple classifier voting approach is used, performing a fusion of label outputs computed on different windows. The K&I method is applied to subimages, which are centered in randomly generated and uniformly distributed pixels. Each window corresponds to a classifier and the generated output is weighted. By combining multiple classifiers we are aiming at a more accurate classification decision even though that comes at the expense of increased complexity and computational cost [10].

2 Methodology

Given two color fundus images I_1 and I_2 of the human retina, acquired at times t_1 and t_2 respectively, $(t_1 < t_2)$, the purpose of a change detection algorithm is to identify the meaningful differences (i.e., the "changes") between I_1 and I_2. The

proposed algorithm also aims at drawing conclusions about what kind of change has occurred at a change pixel by distinguishing the changes due to white spots from the ones given by red spots. Hence, an output image expressing the change occurred at each pixel and its typology is generated. The impact of "perturbating" factors, such as sensor noise, differences in field angle or in patient position, might be partially reduced through the selection of appropriate data. For instance, problems arising from field angle differences can be dealt with by selecting data acquired at the same angle level set by the ophthalmologist.

Here, an unsupervised approach is chosen due to the lack of *a priori* information about the shapes and the statistics of the change areas. After a preprocessing step, which includes the correction for non-uniform illumination, the two co-registered and radiometrically corrected images to be analyzed are converted in a gray level image by computing the ratios of their green and red channels (see Section 2.3). These new gray-level images are compared, in order to generate two further images ("difference images") obtained by a pixel-by-pixel subtraction of I_1 from I_2, and viceversa. The difference image is computed in such a way that pixels associated with retinal changes present gray-level values that are significantly different from those of pixels associated with unchanged areas.

The K&I algorithm is applied in order to automatically detect the change pixels by applying a decision threshold to the histogram of the difference image [11]. The selection of the decision threshold is of major importance, as the accuracy of the final change map strongly depends on this choice. This last step is highly critical in the development of completely automatic and unsupervised techniques for the detection of retinal changes.

In the proposed method the K&I algorithm is applied not to the whole image but to a set of sub-images, which are situated in random positions. For each window a change map is obtained. Subsequently, for each pixel of the image, a fusion of label outputs is performed, where each window map corresponds to a different classifier. This approach compensates for non-uniform illumination across the image.

Subsequently, the change map obtained from the previous step is further classified into different categories (we denote this second stage by *subclassification*), corresponding to the different typologies of change that occurred (red vs white spots). To this end, each pixel is described by some features. Our feature space consists of the green/red ratio and the green channel for both images. We assign the "white spot" and the "red spot" labels based on the intensities of these features, which are compared to corresponding thresholds. The thresholds are selected by using an interactive approach, starting from the average values of the corresponding features, which are calculated on the entire image excluding the dark background. We use a trial-and-error approach by varying the threshold, starting from the average value, with an excursion of 30% of the average value, in order to optimize the results from a visual point of view.

In order to evaluate the multiple classifier approach, the resultant change maps are compared with the results obtained by applying the K&I algorithm to the entire image, without correcting for non-uniform illumination.

2.1 Preprocessing of Retinal Images

Before applying an unsupervised approach to detect changes in two different retinal images, a preprocessing step is usually necessary to make the two images comparable in both the spatial and spectral domains. Concerning the former, the registration has been manually performed by selecting control points, which are located between blood vessel bifurcations, and by applying a 2nd-order polynomial image transformation [12]. With regard to the spectral domain, changes in light, in field angle and in the absorption of the mydriatic drop between the two acquisition times may be potential sources of errors. This problem is mitigated by performing first a radiometric calibration of the images.

Then the optic disc, which appears in color fundus images as a bright yellowish or white region, has to be identified and removed from the two acquisitions. It is important to remove the optic disc for accurate change detection because it has similar attributes to the exudates in terms of brightness, color and contrast. Moreover, its detection is essential for the subsequent illumination correction algorithm. Here, the location of the optic disc is estimated by identifying the area with the highest variation in intensity of adjacent pixels [13]. In fact, the appearance of the optic disc region is characterized by a relatively rapid variation in intensity, because the "dark" blood vessels are beside the "bright" nerve fibres. The variance of intensity of adjacent pixels is used for the localization of the optic disc.

Consequently, the non-uniform illumination is corrected by using a homomorphic filtering technique [14]. For Lambertian surfaces, an observed image I can be modeled as a composition of a luminance component, Ł, and a reflectance component, R (i.e., $I = L \cdot R$). This imaging model holds for retinal images due to the diffusive characteristics of the fundus. An exception for this model is the optic disc, which has to be excluded from the computation. The luminance component can be assumed to vary slowly over space, whereas the reflectance component contains also medium and high frequency details. By first applying the logarithm, we transform the multiplicative relation between I, L and R in an additive one, i.e.:

$$log(I) = log(L) + log(R). \tag{1}$$

After applying the logarithm, the image is low-pass filtered, by using a Gaussian filter, and, then, subtracted from the logarithmic original, yielding a high-pass component (i.e., $log(R)$). Exponentiation of both high-pass and low-pass components approximately separates the image into luminance and reflectance components. Next processing steps are applied to the latter component.

2.2 Multiple Classifiers Approach Using Random Windows

In order to compensate for the problems due to different angles of illumination in the two acquisitions, which causes local illumination variation not compensated by the homomorphic filtering, an approach based on multiple classifiers is used. A thresholding approach for the detection of temporal changes is not applied to the whole image but to a set of randomly selected sub-images, which can

be considered as single classifiers. The windows are generated in a random way: They are centered in randomly generated pixels, which are uniformly distributed in all the image but the dark background. As a result, the windows partially overlap. This random distribution is preferable to the case in which the centers of the windows are equally spaced, because there is a higher variation in the statistics. The dimension of the windows is an important parameter to set: Here, the window areas are about 10% of the area of the original image. Another parameter to set is the number of windows to be used, which influences the average number of votes per pixel. As the number of votes for each pixel increases, the performance of the method improves, until reaching a certain value. From experimental results, the performance of the method improves as the number of the windows increases, until reaching about 30 votes per pixel. Here, 400 windows were generated, which give us about 40 votes per pixel on the average.

In the adopted multiple classifier voting approach, each window corresponds to a single classifier: The thresholding approach (described in Section 2.4) is applied to each sub-image and a change sub-map is obtained. The information stored in each change sub-map needs to be combined in a global change map. For each pixel of the image, all the corresponding classifiers vote for "change" or "no-change" and the classification decision is taken using a weighted sum of the votes. Here, we chose to use a majority vote (i.e., we sum the vote of each classifier with the same weight).

This method compensates for the differences in illumination between the two images to be compared and improves the accuracy of the change detection, especially in the external regions of the image.

2.3 Feature Transformation

The three RGB channels of fundus images contain different information: The red channel is usually the brightest channel but exhibits a very narrow dynamic range; the green channel has the best contrast (the edge of retinal features such as exudates, optic disc, and blood vessels are brighter than in the other channels); the blue channel is non-zero mostly in the areas of the optic disk or of the white spots.

Given an RGB fundus image $I = \{u_{mn} \in \mathbb{R}^3 : m = 0, 1, \ldots, M, n = 0, 1, \ldots, N\}$ of size $M \times N$, a band ratioing between green G and red R channels is applied pixel by pixel. By ratioing these two bands, a new gray-level image is obtained, in which the features of interest are emphasized. In fact, after the application of this operator, vessels and blood regions are darker than the background while white spots are brighter.

2.4 A Thresholding Method for Change Detection in Color Fundus Images

In order to automatically detect temporal changes in each selected sub-image (see Section 2.2), a threshold selection task is addressed by adopting an automatic

change-detection technique, which integrates the image-ratioing approach described in Section 2.3 with a generalization of the K&I's unsupervised minimum-error thresholding algorithm [9] applied to "difference images".

A thresholding approach is a simple classification procedure involving only one input feature, namely, the grey histogram level of a scalar image. Here, such operator is applied to two "difference images" obtained by subtracting pixel-by-pixel the G/R ratio at the second acquisition date by the one at the first date and viceversa. Adopting this approach, the key issue is to choose the threshold in order to keep the number of misclassified pixels as low as possible. We operate in an unsupervised fashion, and therefore the prior probabilities P_1 and P_2 and the parameters of the conditional probability density functions (pdfs) p_1 and p_2 of the classes ω_1 = "no-change" and ω_2 = "change" cannot be estimated through a training set. As a consequence, in place of the global grey level pdf of the difference feature z,

$$p_z(Z) = P_1 p_1(Z) + P_2 p_2(Z), \quad Z \in \mathbb{R}, \tag{2}$$

the histogram $\{h(Z)\}_{Z=0}^{L-1}$ of the considered difference image is used (L denotes the number of levels in the difference image). The selection of an appropriate threshold τ on $[0; L-1]$ is based on the optimization of a given predefined criterion function $J(\tau)$ which averages a cost function $c(\cdot, \tau)$ over the feature histogram $h(\cdot)$ [15]. Kittler and Illingworth proposed a thresholding algorithm [9,15] whose cost function is based on the Bayes decision theory. In particular, they adopted the classification rule for minimum error, under the Gaussian assumption for the class-conditional pdfs (i.e. $p_i(\cdot) = N(m_i, \sigma_i^2)$, where m_i and σ_i^2 are the ω_i-conditional mean and variance, respectively; $i = 1, 2$). Under this hypothesis, the only parameters to be estimated are the class prior probabilities P_1 and P_2, the class means m_1 and m_2, and the class variances σ_1^2 and σ_2^2.

According to the "maximum a posteriori probability" rule, we would like to maximize $P(\omega_i|Z)$ $(i = 1, 2)$. This task is formulated by the K&I method in terms of the threshold τ, by introducing the following cost function [9]:

$$c(Z, \tau) = \frac{[Z - \hat{m}_i(\tau)]^2}{2\hat{\sigma_i}^2(\tau)} - 2 \ln \frac{\hat{P}_i(\tau)}{\hat{\sigma}_i(\tau)}, \tag{3}$$

with $i = 1$ for $z \leq \tau$ and $i = 2$ for $z > \tau$. $\hat{P}_i(\tau)$, $\hat{m}_i(\tau)$ and $\hat{\sigma_i}^2(\tau)$ are histogram-based estimates of the class parameters, which depend on τ $(i = 1, 2)$ [9]. The resulting criterion function is:

$$J(\tau) = 1 + 2 \sum_{i=1}^{2} \hat{P}_i(\tau) \ln \frac{\hat{\sigma}_i(\tau)}{\hat{P}_i(\tau)}. \tag{4}$$

The optimal threshold τ^* is chosen as to minimize $J(\cdot)$, which corresponds to an estimate of the minimum-error threshold. The behavior of the criterion function is strongly related to the scene characteristics, which are represented by the histogram. Typically, only one minimum in the interval $[0, L-1]$ implies histogram bimodality, which reflects the presence of two natural classes (e.g., "change" and "no-change") in the scene.

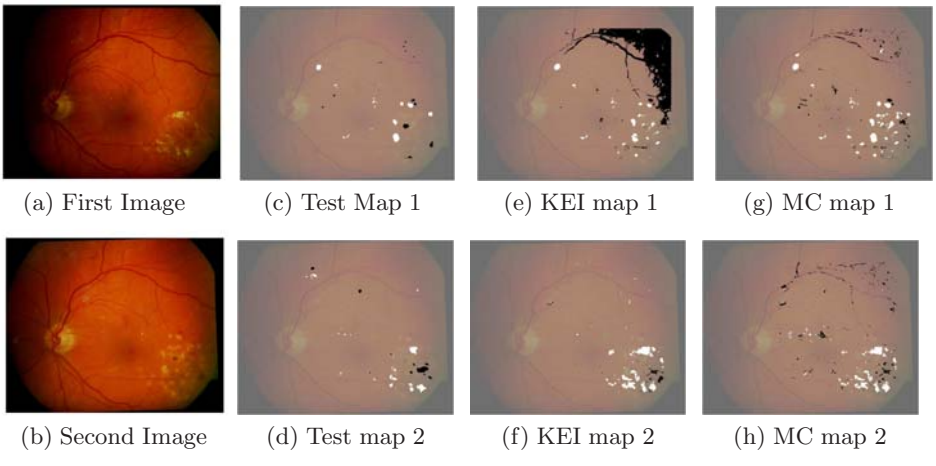
3 Experimental Results

The proposed algorithm has been tested on seven multitemporal fundus images that have been captured by using a ZEISS FF 450plus IR Fundus Camera, connected to a JVC digital camera. The output generated by the camera is an 8-bit RGB color image. In our testing phases no data on age and ethnicity, duration or type of retinopathy were available. In order to compare the results obtained by the algorithm with the performance of a human grader, a test map was created for each data set with the support of a specialist. The images are entirely analyzed, with the exception of a background external region. The results obtained by using the proposed approach based on multiple classifiers (MC) are compared with the change map obtained by applying the K&I thresholding technique to the entire image (KEI). The quantitative evaluation of the results obtained by KEI and MC are shown in Table 1.

The Sensitivity (Sn) is the percentage of pixels which are correctly labeled as "change" in the change map over the number of actual "change" pixels in the test map, as determined by a human observer (i.e., it is an estimate of the detection probability [16]). The Specificity (Sp) is the percentage of pixels which are correctly labeled as "no-change" in the change map (i.e., it is an estimate of $(1 - P_F)$, where P_F is the false-alarm probability [16]). The User's Accuracy for "change" (UA_{ch}) is the percentage of pixels which are correctly assigned to "change" over the total number of pixels labeled as "change" in the change map. A similar definition holds for the User's Accuracy for "no-change" (UA_{nc}) as well. The performance of the method can also be evaluated in terms of correctly identified "change" and "no-change" regions, instead of considering the number of correctly labeled pixels. In particular, the Region-wise Specificity ($R - Sn$) is the percentage of areas which are correctly labeled as "change" in the change map over the number of actual "change" areas in the test map, and the Region-wise User's Accuracy for "no-change" ($R - UA_{ch}$) is the percentage of areas which are correctly assigned to "change" over the total number of areas labeled as "change" in the change map. Sp and UA_{nc} obtained by applying both KEI and MC are generally very high also because the number of true negatives is always high. On the other hand, Sn is more variable because it strictly depends on the quality and similarities in luminance of the input images and is thus affected by sharp differences in the image dynamics at the two dates. The average value of Sn produced by the proposed MC method is in accordance with the British Diabetic Association guidelines that recommend a minimum standard of 80% for Sn and 95% for Sp in the detection of the features which characterize retinopathy [17]. For the user's accuracies, very good values of UA_{nc} were obtained for all data sets, whereas poor values of UA_{ch} were given by the MC method, due to the presence of false-alarm pixels. From this viewpoint, a "minimum-risk" approach could be integrated in the K&I framework (instead of the "minimum-error" approach), in order to gain the capability to explicitly control the tradeoff between false and missed alarms through the use of a cost matrix [16]. However, the application of the MC approach improves the performance in terms of UA_{ch} if compared to the values obtained by using the KEI technique. In fact, the use of multiple classifiers avoids the presence

Table 1. Performances of KEI ans MC applied to seven image pairs, in terms of Sn, Sp, UA_{ch}, UA_{nc}, $R-Sn$, and $R-UA_{ch}$

Method	Parameter	1st pair	2nd pair	3rd pair	4th pair	5th pair	6th pair	7th pair	Average
KEI	Sn	67,7%	39,8%	37,3%	64,2%	4,4%	88,7%	99,6%	57,39%
KEI	Sp	87%	96,8%	97,3%	95,5%	99,9%	97,9%	85,4%	94,26%
KEI	UA_{ch}	5,6%	7%	1,5%	0,6%	8%	8,3%	4,1%	5,01%
KEI	UA_{nc}	99,6%	99,6%	99,9%	99,9%	99,9%	99,9%	100%	99,79%
KEI	$R-Sn$	84,8%	75%	83,3%	84,6%	17,6%	100%	95,4%	77,24%
KEI	$R-UA_{ch}$	54,9%	31,3%	5,7%	18,6%	50%	38%	75%	39,07%
MC	Sn	76,5%	72%	97,1%	97,7%	86,1%	85,6%	45,8%	80,11%
MC	Sp	96,6%	96,9%	98,3%	99,6%	98,1%	99%	99,6%	98,3%
MC	UA_{ch}	21,5%	12,5%	6,2%	10,2%	8%	15%	44%	16,77%
MC	UA_{nc}	99,7%	99,8%	100%	100%	99,9%	99,9%	99,6%	99,84%
MC	R-Sn	94,7%	96,3%	100%	100%	88,2%	94,1%	56,2%	89,93%
MC	R-UA_{ch}	37%	19,5%	2,3%	23%	15,9%	27,1%	100%	32,11%



(a) First Image (c) Test Map 1 (e) KEI map 1 (g) MC map 1

(b) Second Image (d) Test map 2 (f) KEI map 2 (h) MC map 2

Fig. 1. First data set: Registered input images acquired from the same eye in June 4, 2003 (a) and in January 24, 2005 (b). Test maps (c) and (d), change maps generated by KEI (e) and (f), and change maps generated by MC (g) and (h). In order to visualize the different change typologies, for each method two change maps are shown, transparently superposed to the first image. Map legend: White in map 1 = new white spots, black in map 1 = old red spots, white in map 2 = old white spots, black in map 2 = new red spots, background = "no-change".

of wide false alarm areas, otherwise caused by differences in luminance. The values of UA_{ch}, as those of Sn, are higher in the evaluation in terms of regions. In fact, the presence and the position of most "change" areas are correctly detected, even when their shape is not perfectly reconstructed. // The change maps generated by KEI and by proposed multiple classifier approach when applied to the first data set (Figs. 1(a) and (b)) are shown in Figs. 1(e)-(f) and in Figs. 1(g)-(h), respectively.

Several typologies of change are present in this data set, including new and old spots of both types: The related test maps are shown in Fig. 1(c) and (d). A lower value of Sn is obtained in this case (about 76.5%), due to several missed alarms where edges between "change" and "no-change" are present. Anyway, the detection of the changes and their classification among the different typologies, which is our aim (representing clinically relevant information), are achieved, as shown by the higher value of $R - Sn$ in terms of identified regions (about 94.7%).

4 Conclusions

A method able to detect temporal changes in color fundus images using multiple classifiers has been proposed. Seven different data sets (comprising one pair of images), including changes of different sizes and typologies, were taken into account in order to test the performances of the proposed method. The multiple classifier approach for change detection (based on the Kittler & Illingworth's thresholding algorithm) provided quite accurate results. The results are good and in accordance with the performance specifications recommended by the British Retinopathy guidelines.

Moreover, thresholding is a very fast approach: No iterations are needed, but only the calculation of a criterion function, which is defined for L values (e.g., $L = 256$). Given the histogram, the computation time is also independent of the image size.

The use of the multiple classifier approach increases the computational time of the method (from about 1 minute per image pair up to 5 minutes) but at the same time makes it more robust; as a result, the accuracy of the method increases.

Very good accuracies have been obtained for the analyzed images, for which the preprocessing phase effectively corrected the geometrical and radiometrical discrepancies between the two acquisition dates. The main drawback is the possible sensitivity to the presence of undesired modes such as artifacts and glares.

The development and the implementation of a method for automatically identifying ungradable images may be an important next step of this research. A further future development could involve the application of an algorithm to automatically register images, in order to make this pre-processing fully automatic.

Acknowledgments. This work was supported by the Research Fund of the University of Iceland.

References

1. Fritzsche, K.H.: Computer vision algorithms for retinal vessel detection and width change detection. Ph.D. dissertation, Rensselaer Polytechnic Inst. Try, New York (2004)
2. Akita, K., Kuga, H.: A computer method of understanding ocular fundus images. J. Pattern recognition 15, 431–443 (1982)

3. Berger, J.W., Shin, D.S.: Computer vision enabled augmented reality fundus biomicroscopy. J. Ophthalmology 106 (1999)
4. Cree, M.J., Olson, J.A., McHardy, K.C., Sharp, P.F., Forrester, J.V.: A fully automated comparative microaneurysm digital detection system. J. Eye 11, 622–628 (1997)
5. Lalibert, F., Gagnon, L., Sheng, Y.: Registration, fusion of retinal images - an evaluation study. IEEE Transactions on Medical Imaging 22, 661–673 (2003)
6. Usher, D., Dumskyj, M., Himaga, M., Williamson, T.H., Nussey, S., Boyce, J.: Automated detection of diabetic retinopathy in digital retinal images: a tool for diabetic retinopathy screening. J. Diabetic Medicine 21, 84–90 (2003)
7. Hipwell, J.H., Strachan, F., Olson, J.A., McHardy, K.C., Sharp, P.F., Forrester, J.V.: Automated detection of microaneurysms in digital red-free photographs: a diabetic retinopathy screening tool. J. Diabetic Medicine 17, 588–594 (2000)
8. Walter, T., Klein, J.C., Massin, P., Erginary, A.: A contribution of image processing to the diagnosis of diabetic retinopathy - detection of exudates in color fundus images of the human retina. IEEE Transactions on Medical Imaging 21, 1236–1243 (2002)
9. Kittler, J., Illingworth, J.: Minimum error thresholding. J. Pattern Recognition 19, 41–47 (1986)
10. Kuncheva, L.I.: Combining pattern classifiers: Method and algorithms. Wiley Interscience, New Jersey (2004)
11. Troglio, G., Nappo, A., Benediktsson, J.A., Moser, G., Serpico, S.B., Stefansson, E.: Automatic Change Detection of Retinal Images. In: Proceedings of the IUPESM Medical Physics and Biomedical Engineering - World Congress (2009)
12. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice Hall Inc., Englewood Cliffs (2001)
13. Sinthanayothin, C., Boyce, J.F., Cook, H.L., Williamson, T.H.: Automated localisation of the optic disc, fovea, and retinal blood vessels from digital color fundus images. British Journal of Ophthalmology 83, 902–910 (1999)
14. Narasimha-Iyer, H., Can, A., Roysam, B., Stewart, C.V.: Robust detection and classification of longitudinal changes in color retinal fundus images for monitoring diabetic retinopathy. IEEE Transactions on Biomedical Engineering 53(6), 1084–1098 (2006)
15. Chi, Z., Yan, H., Pham, T.: Fuzzy algorithms: with applications to image processing and pattern recognition. World Scientific Publishing, Singapore (1996)
16. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2001)
17. Sinthanayothin, C., Boyce, J.F., Williamson, T.H., Cook, H.L., Mensah, E., Lal, S., et al.: Automated detection of diabetic retinopathy on digital fundus images. J. Diabetic Medicine 19, 105–112 (2002)

A Double Pruning Algorithm for Classification Ensembles

Víctor Soto, Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez

Universidad Autónoma de Madrid, EPS, Calle Francisco Tomás y Valiente, 11, Madrid 28049 Spain

Abstract. This article introduces a double pruning algorithm that can be used to reduce the storage requirements, speed-up the classification process and improve the performance of parallel ensembles. A key element in the design of the algorithm is the estimation of the class label that the ensemble assigns to a given test instance by polling only a fraction of its classifiers. Instead of applying this form of dynamical (instance-based) pruning to the original ensemble, we propose to apply it to a subset of classifiers selected using standard ensemble pruning techniques. The pruned subensemble is built by first modifying the order in which classifiers are aggregated in the ensemble and then selecting the first classifiers in the ordered sequence. Experiments in benchmark problems illustrate the improvements that can be obtained with this technique. Specifically, using a bagging ensemble of 101 CART trees as a starting point, only the 21 trees of the pruned ordered ensemble need to be stored in memory. Depending on the classification task, on average, only 5 to 12 of these 21 classifiers are queried to compute the predictions. The generalization performance achieved by this double pruning algorithm is similar to pruned ordered bagging and significantly better than standard bagging.

Keywords: ensemble pruning, instance-based pruning, ensemble learning, decision trees.

1 Introduction

There is extensive empirical evidence that combining the predictions of complementary classifiers is a successful strategy to build robust classification systems with good generalization performance [1,2,3]. The main disadvantages of ensemble methods are the difficulties in the interpretation of the decisions of the ensemble and their large computational requirements. In particular, the training cost, the storage needs and the time of prediction increase linearly with the number of classifiers that are included in the ensemble. If the errors of the classifiers in the ensemble were uncorrelated, averaging over larger ensembles should improve the accuracy of the predictions, because the errors of a given classifier would be compensated by the correct predictions of other classifiers in the

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 104–113, 2010.

ensemble. In practice, the ensemble members tend to make errors in the same examples. Nonetheless, in a wide range of classification problems, the accuracy of parallel ensembles, such as bagging, improves with the number of classifiers that are included in the ensemble. However, larger ensembles have larger storage needs and longer times of prediction. To alleviate these shortcomings, different ensemble pruning methods can be used [4,5,6,7,8,9,10,11,12,13]. The goal of these methods is to reduce the memory requirements and to speed-up the classification process while maintaining or, if possible, improving the level of accuracy of the original ensemble. Most ensemble pruning methods replace the original ensemble by a representative subset of predictors. Besides needing less storage space and predicting faster, pruned subensembles can actually outperform the original classification ensembles from which they are extracted [5,6,7,8,10].

Using a different approach, the time needed for prediction using a parallel ensemble, such as bagging [14], can be reduced by a dynamical pruning method called instance-based (IB) pruning [11]. In IB pruning the number of classifiers that need to be queried to estimate the final ensemble prediction is determined for each instance separately. Assuming that simple majority voting is used, it is possible to estimate the final decision by querying only a subset of the classifiers in the ensemble. Given a test instance that needs to be classified, the aggregation of the outputs of the ensemble members is halted when the probability that the remaining predictions do not change the current majority class is above a specified confidence level α. Since the overhead needed to determine whether the aggregation process should be halted is negligible, the reduction in the number of queries directly translates in a speed-up of the classification process. This method does not reduce the storage requirements, because all the classifiers in the original ensemble need to be available for potential queries. Since the differences in prediction are below the threshold $1 - \alpha$, the differences between the errors of the dynamically pruned ensemble and of the original ensemble are also necessarily below $1 - \alpha$. Therefore, the generalization performance of the ensemble is only slightly modified by IB-pruning.

The theoretical analysis of majority voting on which IB-pruning is grounded relies on the fact that in parallel ensembles the individual classifiers are generated under the same conditions and independently of each other. The goal of this investigation is to determine whether IB-pruning can be also used in sequential ensembles. When the ensemble is sequential, the classifier that is added at one point in the sequence depends on the classifiers that have been included in the ensemble up to that point. As a result, one introduces correlations among classifiers, which can result in biases in the estimation of the final ensemble prediction on the basis of the outputs of the initial classifiers in the sequence. The results of experiments on benchmark classification problems carried out in this investigation show that the biases introduced by IB-pruning can cause some distortions in the estimation of the error rate of the complete ensemble when ordered aggregation is used. By contrast, IB-pruning is remarkably effective when it is used to halt the aggregation process not in the complete ordered ensemble, but in the subensemble that is obtained by selecting the first $\approx 20\%$ classifiers in the

reordered sequence. We conjecture that the reason for this different behavior is related to the properties of ordered bagging. The curves that trace the dependence of the error rate on the size of the ordered ensemble exhibit a minimum at intermediate ensemble sizes. This means that the first and the last classifiers included in the ordered bagging ensemble have rather different properties. As a matter of fact, the last classifiers that are included in the ordered sequence cause a deterioration instead of an improvement of the error rate. In consequence, estimations based on the first classifiers in the ensemble can be very different from the final decision, which takes into account all the classifiers in the ensemble. By contrast, in the second case, the test error rate monotonically decreases with the size of the ensemble, which implies that the trends detected in the output of the first classifiers tend to be reinforced by the subsequent predictions.

In summary, we propose a double pruning algorithm, in which dynamical IB-pruning is applied to a pruned bagging ensemble built with ordered aggregation. The method combines the advantages of pruning by ordered aggregation and instance-based pruning; namely the generalization performance of the ensemble is improved, the storage requirements are reduced, because only the classifiers in the pruned ensemble need to be stored in memory, and, finally, the efficiency of the classification process is significantly ameliorated, not only because the number of classifiers of the pruned ensemble is smaller, but also because IB-pruning speeds up the prediction of the class label for individual instances.

The paper is organized as follows: Section 2 provides a review of ordered aggregation. The dynamical pruning algorithm IB-pruning is described in Section 3. Section 4 summarizes the results of experiments on benchmark classification tasks that demonstrate the effectiveness of the double pruning algorithm proposed. Finally, the conclusions of this work are exposed in Section 5.

2 Ensemble Pruning Based on Ordered Aggregation

A possible approach to ensemble pruning is to select from the original ensemble a subset of representative classifiers whose combined performance is equivalent or better than the complete ensemble. There are two sources of difficulties in the realization of this goal. The first handicap is that the selection of classifiers has to be based on estimates on the training data. However, the objective is to identify a subensemble that has good generalization performance. Even if we can compute accurate estimates of the generalization accuracy on the basis of the training data only, finding the optimal subensemble is a computationally expensive problem that involves comparing all the possible $2^T - 1$ non-empty subensembles that can be extracted from the original ensemble. A feasible approach is to use a greedy strategy based on modifying the order in which classifiers are aggregated in the ensemble [6,15,10]. Starting from an initial pool of classifiers, in which no particular ordering for combination is specified, ordered aggregation builds a nested sequence of ensembles of increasing size by incorporating at each step the classifier that improves the performance of the enlarged ensemble the most. The first classifier in the sequence is generally the one with the lowest training

error. From the subensemble $\mathcal{S}_{t-1}$ of size $t-1$, the subensemble of size $\mathcal{S}_t$ is constructed by incorporating a single classifier from the remaining pool of classifiers. This classifier is selected by maximizing a measure that is expected to be correlated with the generalization performance of the ensemble. The measures that are effective for this selection take into account the complementarity of the classifiers selected, not only their individual accuracy or their diversity. In this article we use boosting-based ordered bagging [15], which uses the weighted training error defined in boosting to direct the ordering process. The algorithm proceeds iteratively by updating the training example weights as in boosting: the weights of training examples correctly (incorrectly) classified by the last classifier incorporated into the ensemble are decreased (increased) according the AdaBoost prescription [16]. The classifier that minimizes the weighted training error is then incorporated into the ensemble. The results of extensive experimental evaluation using bagging to generate the initial pool of classifiers show that early stopping in the aggregation process allows to identify pruned ensembles, whose size is $\approx 20\%$ of the complete ensemble, which outperform bagging and retain baggings resilience to noise in the class labels of the examples.

3 Instance-Based Pruning

Consider a binary classification problem. Assume that we have built a parallel ensemble composed of T classifiers built independently of each other. Each classifier is induced from the same learning data by repeated applications of a learning algorithm that involves some form of randomization. Consider an arbitrary instance $\mathbf{x}$ that needs to be classified. Assume that only t classifiers have been queried, and that the current (partial) vote count is t_1 for class 1 and t_2 for class 2 ($t_1+t_2=t$). Without loss of generality we can assume that $t_1 \geq t_2$. Using the fact that the classifiers in a parallel ensemble are generated independently of each other, the probability that the class labels predicted by the subensemble of size $t < T$ and by the complete ensemble of size T coincide is [11]

$$\tilde{\mathcal{P}}(t_1,t,T) = \sum_{T_1=max\{t_1,1+\lfloor T/2 \rfloor\}}^{T-t_2} \frac{(T-t)!}{(T_1-t_1)!(T_2-t_2)!} \frac{(t_1+1)_{T_1-t_1}(t_2+1)_{T_2-t_2}}{(t+2)_{T-t}} \tag{1}$$

where $T_1+T_2=T$, and $(a)_n = a(a+1)\cdots(a+n-1)$ is the Pocchammer symbol, or rising factorial, with a and n nonnegative integers. If it is acceptable that, with a small probability $1-\alpha$, the predictions of the partially polled ensemble and of the complete ensemble disagree, the voting process can be stopped when the probability (1) exceeds the specified confidence level α. The final classification is estimated as the combined decision of the polled classifiers only. In particular, the querying process can be halted after t classifiers have been queried, if the vector of class predictions of the current subensemble $t_1^\star(t;T,\alpha)$ is such that $\tilde{\mathcal{P}}(t_1^\star,t,T) \geq \alpha$. For an ensemble of $T=101$ classifiers and a confidence level $\alpha = 99\%$ the first few values of $t_1^\star(t;T=101,\alpha=0.99)/t$ are $6/6, 7/7, 8/8, 8/9, 9/10, 10/11, 10/12, 11/13, \ldots$

4 Experiments

In this section we perform experiments to determine whether IB-pruning can be used in combination with ordered bagging. In the first set of experiments IB-pruning is applied to a standard (randomly ordered) bagging ensemble. As expected, the results of these experiments confirm the effectiveness of IB-pruning in parallel ensembles. A second batch of experiments show that IB-pruning is not effective when applied to ordered bagging because of the differences between the classifiers that appear in the first positions in the ordered ensemble and those that appear in the last positions. Finally, IB-pruning applied to a pruned ensemble that is obtained by selecting the first $\approx 20\%$ classifiers in the ordered bagging ensemble. This last series of experiments illustrates the effectiveness of IB-pruning on the pruned ensemble in the problems investigated.

All the experiments are performed on twelve binary classification problems from the UCI repository [17]. The same learning setup is used to make comparisons possible. In all cases the results reported are averages over 10 independent 10-fold cross validation estimates. The protocol followed in each execution for a partition of the data into training and test is as follows: (i) Build a bagging ensemble composed of $T = 101$ CART trees [18] using the training set. The standard settings for the generation of the decision trees are used. The ordering of the initial ensemble is determined by the order of generation, which is random in bagging. (ii) Estimate the generalization error in the test set for the whole ensemble and for the first 21 trees in the randomly ordered ensemble. Apply IB-pruning to the complete ensemble using $\alpha = 99\%$ recording the test error and the average number of trees used to classify the instances. (iii) Modify the sequence of aggregation of the trees in the ensemble using boosting-based ordering [15]. This method is similar to boosting. However, instead of generating new classifiers at each step, one selects the classifier from the original ensemble that minimizes a weighted error on the training set. The weights of the instances in the formula for the weighted error are specified according to the prescription given by boosting. The test error for the ordered bagging using the first 21 trees of the ensemble. This value for the number of selected trees produces consistently good results in a wide range of datasets [10]. Apply IB-pruning to ordered bagging ensemble of $T = 101$ using $\alpha = 99\%$. Compute the average test error and the average number of classifiers used to classify the instances. (iv) Finally, apply IB-pruning to the first 21 trees of the ordered ensemble ($T = 21$ and $\alpha = 99\%$) recording the number of trees and classification error.

The results of applying IB-pruning to bagging are summarized in Table 1. For each dataset, the table shows the average test error for bagging (BAG101), bagging using the first 21 randomly generated classifiers (BAG21) and IB-pruning applied to the full bagging ensemble (IB-BAG101). The average number of trees used to classify each instance in IB-BAG101 is shown in the last column of the table. The corresponding standard deviations are displayed after the $\pm$ sign. These experiments confirm the results reported in [11]. Table 1 shows that the generalization error of a bagging ensemble with 101 trees is generally better than

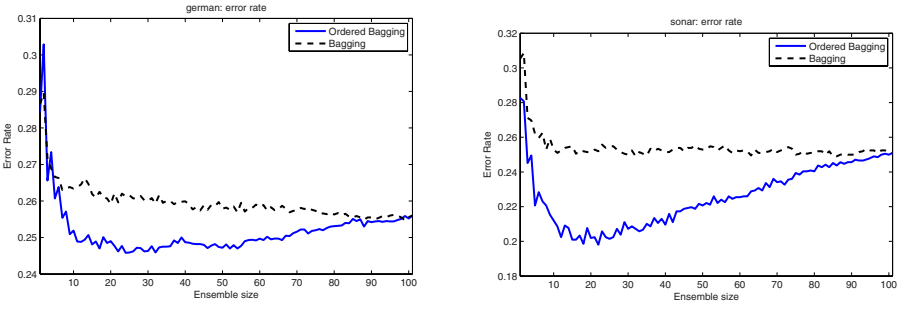


Fig. 1. Test error curves with respect to the number of classifiers for bagging and bagging ordered using boosting based ordering

Table 1. Results for bagging (best methods are highlighted in boldface)

	Test error			# trees IB
Problem	BAG101	BAG21	IB-BAG101	($\alpha = 99\%$)
australian	**14.5±3.8**	**14.5±3.8**	**14.5±3.8**	7.4±1.0
breast	**4.8±2.8**	**4.8±2.6**	**4.8±2.8**	8.6±1.3
diabetes	**24.9±3.9**	**24.9±4.0**	**24.9±3.9**	14.3±2.3
german	**25.6±3.1**	25.9±3.5	25.7±3.1	18.0±2.6
heart	19.6±8.0	19.9±8.0	**19.4±7.8**	18.5±4.8
horse-colic	17.8±6.3	17.9±6.0	**17.7±6.2**	9.9±3.0
ionosphere	**9.7±4.6**	9.8±4.5	**9.7±4.5**	8.6±1.9
labor	**13.4±12.8**	14.1±12.9	13.7±12.6	17.7±8.7
liver	**31.1±6.2**	31.9±6.9	**31.1±6.2**	21.1±5.2
sonar	25.1±9.7	25.1±9.2	**25.0±9.7**	19.4±5.4
tic-tac-toe	**1.6±1.3**	2.2±1.5	**1.6±1.3**	10.1±1.3
votes	**4.4±3.0**	**4.4±3.0**	**4.4±3.0**	6.1±0.3

a bagging ensemble composed of 21 classifiers. This also confirms the observation that increasing the size of parallel ensembles in which the generation of the individual classifiers involves some form of randomization generally improves the generalization performance of the ensemble [19]. In contrast, when IB-pruning is used to determine when to stop querying for the classification of new instances, a performance comparable to BAG101 is achieved in the studied datasets using on average a fairly small fraction of the classifiers. In particular, the average number of trees that need to be queried in IB-BAG101 ranges from 6.1 for *Votes* to 21.1 for *Liver*.

Table 2 compiles the results of the application of IB-pruning to ordered bagging ensembles. The column labeled BAG101 displays the average and, after the ± symbol, the standard deviation of the test error rate obtained by a bagging ensemble composed of 101 trees. The second column presents the results of IB-pruning when applied to the complete ordered bagging ensemble (IB-OB101). The average number of trees used by IB-OB101 is given in the fifth column.

Table 2. Results for ordered bagging (best methods are highlighted in boldface)

	Test error				# trees IB ($\alpha = 99\%$)	
Problem	BAG101	IB-OB101	OB21	IB-OB21	IB-OB101	IB-OB21
australian	14.5±3.8	14.3±3.9	**13.7±3.9**	**13.7±4.0**	11.3±1.7	7.0±0.5
breast	4.8±2.8	4.5±2.6	4.1±2.6	**4.0±2.6**	8.7±1.2	5.9±0.3
diabetes	24.9±3.9	24.7±4.0	**24.3±3.9**	**24.3±3.9**	17.2±2.3	8.7±0.6
german	25.6±3.1	25.2±3.3	24.8±3.7	**24.7±3.8**	21.1±2.5	9.3±0.6
heart	19.6±8.0	18.9±7.6	**18.6±7.2**	**18.6±7.1**	20.2±4.0	9.4±1.0
horse-colic	17.8±6.3	17.5±6.2	**16.3±6.6**	**16.3±6.5**	9.8±2.1	6.6±0.7
ionosphere	9.7±4.6	8.5±4.4	**7.5±4.2**	**7.5±4.1**	10.9±2.0	6.7±0.6
labor	13.4±12.8	10.0±11.3	**8.3±10.0**	8.5±10.0	14.8±7.5	7.9±1.9
liver	31.1±6.2	29.5±6.2	**28.2±6.5**	28.4±6.7	28.0±4.6	11.8±0.9
sonar	25.1±9.7	23.6±9.5	**20.2±10.7**	**20.2±10.7**	26.1±5.3	11.2±1.3
tic-tac-toe	1.6±1.3	**1.4±1.2**	**1.4±1.2**	1.5±1.2	9.4±1.0	6.5±0.4
votes	**4.4±3.0**	**4.4±3.1**	4.7±3.2	4.6±3.2	7.0±0.8	5.6±0.3

The results for a pruned ensemble composed of the first 21 trees of the ordered bagging ensemble are given in the column labeled OB21. These results show that the performance of ordered bagging with 21 classifiers is better than that of full bagging for all the datasets investigated except for *Votes*. Ordered bagging has two advantages over bagging: faster classification, because only a small fraction ($\approx 20\%$) of the original classifiers is used, and, in general, better accuracy in the test set. Instead of using a fixed number of classifiers, IB-pruning individually determines the number of classifiers that are needed to estimate the complete ensemble prediction for each particular instance. When IB-pruning is used in conjunction with ordered bagging (column IB-OB101 in Table 2), the number of queried classifiers is generally lower than the 21 trees used in pruned bagging (OB21). However, it is over the number of elements queried by IB-pruning for randomly ordered bagging (right most column of Table 1). In addition, the accuracy improvement with respect to bagging is not as ample as the improvement of OB21 over BAG101. This poorer performance is a consequence of the fact that IB-OB101 is making inference about the predictions of the complete ensemble on the basis of the predictions of only the first classifiers in the ordered sequence. These classifiers follow a distribution that is different from the overall distribution of classifiers in bagging. These results can be understood analyzing the plots displayed in Fig. 1. The curves depicted trace the dependence of the test error with the size of the ensemble using bagging and ordered bagging for the classification tasks *German* and *Sonar*. This figure shows that by stopping the aggregation of classifiers at $\approx 20 - 30\%$ of the total number of elements in the ensemble, a significant reduction in the classification error is obtained. These error curves are representative of the general behavior of bagging and ordered bagging in all the datasets investigated.

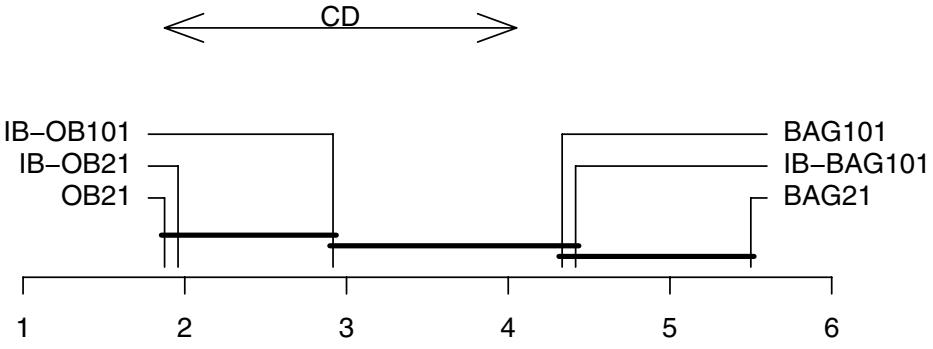


Fig. 2. Comparison of the different methods using the Nemenyi test. Classification systems whose performance are not significantly different according to this test (p-value < 0.05) are connected by a line segment in the diagram.

In the final batch of experiments IB-pruning is applied to a pruned ensemble composed of the first 21 classifiers in ordered bagging. The results of these experiments are displayed in the fourth column of Table 2 (IB-OB21). The last column shows the average number of trees used by IB-OB21. These results, show that the generalization error of IB-pruning applied to OB-21 is equivalent to that of OB21 in the problems analyzed. Small variations of one or two tenths of a percent point both positive and negative can be observed for some datasets. Therefore, the improvements obtained by IB-OB21 over complete bagging (BAG101) are of the same magnitude as the improvements obtained by the pruned ensemble obtained by early stopping in ordered aggregation (OB21). The number of trees that need to be stored in memory is also reduced from 101 to 21 trees. Finally, the average number of trees that need to be queried is further reduced by the application of IB-pruning to the pruned ensemble OB21. Specifically, IB-OB21 employs an average number of trees that ranges from 5.6 (*Votes*) to 11.8 (*Liver*). In summary, the application of IB-pruning to the pruned ensemble obtained from ordered aggregation (OB21) improves the accuracy and reduces the memory requirements of bagging as much as OB21 does. It has the additional advantage that it predicts even faster than OB21.

The overall generalization performance of the different ensemble methods in the classification tasks analyzed is compared using the methodology proposed by Demšar [20]. Fig. 2 displays the rank of each method averaged over the results in the different classification tasks. In this diagram, the differences between methods connected with a horizontal line are not significant according to a Nemenyi test (p-value$<$ 0.05). The critical difference (CD=2.2 for 6 methods, 12 dataset and p-value$<$ 0.05) is shown for reference. The best overall performance corresponds to OB21 and IB-OB21. The performance of these two methods is equivalent in the classifications tasks investigated. According to this test the performance of OB21 and IB-OB21 in terms of average rank is significantly better than standard bagging. The performances of the remaining methods are not significantly different from bagging.

5 Conclusions

In this article we propose to combine two existing pruning strategies to reduce the computational costs associated with the use of ensembles of classifiers for prediction and to improve their generalization performance. The first strategy selects a subset of complementary classifiers from the original ensemble to reduce the storage requirements, speed-up the classification process and improve the generalization performance. The algorithm is based on modifying the order of aggregation of the classifiers in the ensemble: a nested sequence of ensembles of increasing size is built by incorporating at each step the classifier that is expected to improve the classification error the most. The pruned subensemble is obtained by early stopping in the ordered aggregation process. In this article, the weighted error function used in boosting is used to guide the ordered aggregation. Nonetheless, other criteria based on the complementarity of the classifiers incorporated in the ensemble can also be used. Experiments in benchmark classification problems have shown that this strategy can improve the generalization error of bagging ensembles by keeping only $20-30\%$ of the classifiers, the first ones in the ordered sequence. The second strategy, instance-based pruning (IB-pruning), does not require any manipulation of the ensemble. It is applied dynamically when a new instance is classified. Using the fact that the classifiers in a parallel ensemble, such as bagging, are generated independently of each other, it is possible to compute the probability that the majority class obtained after having queried t classifiers will not change when the output the remaining classifiers becomes known. If this probability is above a specified confidence level α. The application of this method does not reduce the storage requirements (all the classifiers need to be stored in memory for potential queries), but it leads to a significant reduction in the average number of queries of ensemble classifiers without a significant modification of the accuracy of the ensemble.

In the problems investigated, IB-pruning applied to the original (randomly ordered) bagging ensemble obtains error rates similar to the complete ensemble and reduces the average number of queries more than the pruned ensemble that is built by selecting the first 21 classifiers in the ordered ensemble. However, its accuracy is lower than the pruned ensemble. When IB-pruning is applied to the complete ordered ensemble its generalization accuracy is better than the complete ensemble. Nevertheless, this accuracy is still inferior to the pruned ordered ensemble. This is due to the fact that the distribution of the predictions of classifiers that appear first in the ordered ensemble is different from the last classifiers included. Therefore, one of the basic assumptions of IB does not hold, leading to suboptimal performance. Finally, when IB-pruning is applied to the pruned ordered ensemble itself, a significant speed-up is achieved with a performance that is similar to the pruned ensemble and much better than bagging. The result is a double pruning algorithm that significantly improves the performance of bagging, achieving similar accuracy as pruned ordered bagging. Furthermore, it reduces the memory requirements, because only the classifiers that are selected in the pruned ordered ensemble need to be accessible for potential queries, and predicts much faster than standard bagging.

References

1. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
2. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40(2), 139–157 (2000)
3. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, Hoboken (2004)
4. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. of the 14th International Conference on Machine Learning, pp. 211–218. Morgan Kaufmann, San Francisco (1997)
5. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence 137(1-2), 239–263 (2002)
6. Martínez-Muñoz, G., Suárez, A.: Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications, pp. 258–263. Acta Press (2004)
7. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proc. of the 21st International Conference on Machine Learning, p. 18. ACM Press, New York (2004)
8. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: Ensemble diversity measures and their application to thinning. Information Fusion 6(1), 49–62 (2005)
9. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338 (2006)
10. Martínez-Muñoz, G., Hernández-Lobato, D., Suárez, A.: An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 245–259 (2009)
11. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: Statistical instance-based pruning in ensembles of independent classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 364–369 (2009)
12. Latinne, P., Debeir, O., Decaestecker, C.: Limiting the number of trees in random forests. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 178–187. Springer, Heidelberg (2001)
13. Sharkey, A., Sharkey, N., Gerecke, U., Chandroth, G.: The Test and select approach to ensemble combination. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 30–44. Springer, Heidelberg (2000)
14. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
15. Martínez-Muñoz, G., Suárez, A.: Using boosting to prune bagging ensembles. Pattern Recognition Letters 28(1), 156–165 (2007)
16. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
17. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
18. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)
19. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
20. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

Estimation of the Number of Clusters Using Multiple Clustering Validity Indices

Krzysztof Kryszczuk and Paul Hurley

IBM Zurich Research Laboratory, Switzerland

Abstract. One of the challenges in unsupervised machine learning is finding the number of clusters in a dataset. Clustering Validity Indices (CVI) are popular tools used to address this problem. A large number of CVIs have been proposed, and reports that compare different CVIs suggest that no single CVI can always outperform others. Following suggestions found in prior art, in this paper we formalize the concept of using multiple CVIs for cluster number estimation in the framework of multi-classifier fusion. Using a large number of datasets, we show that decision-level fusion of multiple CVIs can lead to significant gains in accuracy in estimating the number of clusters, in particular for high-dimensional datasets with large number of clusters.

Keywords: clustering, clustering validity indices, multiple classifier.

1 Introduction

Clustering algorithms are unsupervised machine learning techniques seeking to discover structure in unlabeled data sets. They analyze the similarities between the data and group it into similarity clusters [1,2]. One of the properties of interest when analyzing unknown data is the number of such clusters. Discovering the number of clusters inherently present in the data has important practical applications, including computational biology [3], web text mining [4], etc.

To date no theoretically optimal method for finding the number of clusters inherently present in the data has been proposed. Existing algorithms include stability-based methods [5,6], model-fitting-based algorithms [7], and methods based on *Clustering Validity Indices* (CVI) [1]. A CVI is a measure derived from the obtained clustering solution, which quantifies such properties of a clustering solution as compactness, separation between clusters, etc. The principal usage of CVIs is comparison of alternative clustering solutions. CVIs can be used for the task of cluster number estimation, if the compared clustering solutions use different number of clusters [8].

A recurring problem with the use of CVIs is that each CVI is designed to capture a specific aspect of the clustering solution that suggests how adequate that solution is. At the same time, other aspects can be inadequately represented or ignored altogether. For instance, a CVI that verifies how compact the obtained clusters are will frequently report that elongated clusters are not compact at all,

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 114–123, 2010.

although such clusters may correctly represent the data. No CVI can a-priori be assumed better than its alternatives [8].

In this paper, we propose to overcome this problem by estimating the number of clusters by a fusion of multiple CVIs in order to estimate the number of clusters in the data. We notice an analogy between using multiple CVIs and constructing multiple classifier ensembles [9]. The motivation of our work is that although no single CVI can capture correctly the validity of any clustering solution, a proper conciliation of multiple CVIs should be able to cope with the task. In this work we treat the yield of each considered CVI as a feature that characterizes the quality of a clustering solution. We hypothesize that an ensemble of CVI features is a better predictor of clustering quality than any of the CVIs taken separately. To the best of our knowledge, the only attempt to combine validity indices into a compound index was reported in [10]. However, fusion of CVIs for cluster number estimation is not the main topic of [10]. Consequently, only simple score averaging of CVIs were used, and no comparison with other possible solutions was presented. In this paper, we compare score-, and decision fusion strategies of combining multiple CVIs using a large number of synthetic datasets. The results of our experiments suggest that decision-based multiple CVI fusion techniques offer highest accuracy in detecting the correct number of clusters in a dataset, provided that all used CVIs are individually competent. The result of this paper are particularly relevant to high-dimensional data analysis with multiple clusters, when visual inspection of the clustering solution is impractical or impossible.

The rest of the paper is structured as follows. Section 2 gives an overview of the CVIs used in this work. In Section 3 we provide the framework of cluster number estimation using multiple CVIs. The applied experimental methodology and the experimental results are described in Section 4. Section 5 gives a discussion of the results and concludes the paper.

2 Internal Clustering Validity Indices

In this work, we used three popular, internal clustering validity indices, the Davies-Bouldin, Calinski-Harabasz and Dunn indices, as in [11]. The choice of the CVIs was dictated by the fact that they are well-known measures, frequently used for the task of choosing the best clustering solution. The Davies-Bouldin and the Calinski-Harabasz indices were reported as performing well for the task of comparing clustering solutions [12].

Davies-Bouldin index is computed as

$$s_{DB} = \frac{1}{k}\sum_{i=1}^{k} max_{j=1,\ldots k;i\neq k}\left(\frac{d_i + d_j}{d(c_i, c_j)}\right), \tag{1}$$

where k denotes the number of clusters. If i, j are cluster labels, then d_i and d_j are average distances of all patterns in clusters i and j to their respective cluster centroids, and $d(c_i, c_j)$ is the distance between these centroids [13]. Smaller value

of s_{DB} indicates a "better" clustering solution. In order to be consistent with other two clustering validity indices considered in this work, we used $1 - s_{DB}$ to have a maximum as an indicator of the preferred clustering solution.

Calinski-Harabasz index is computed by

$$s_{CH} = \frac{trace(S_B)}{trace(S_W)} \cdot \frac{n_p - 1}{n_p - k}, \tag{2}$$

where S_B is the between-cluster scatter matrix, S_W the within-cluster scatter matrix, n_p the number of clustered points, and k the number of clusters [14]. When using the CH index to compare clustering solutions, maximal value of s_{CH} identifies the preferred candidate.

Dunn index is defined as

$$s_{Dunn} = \frac{d_{min}}{d_{max}}, \tag{3}$$

where d_{min} is the minimum distance between two points belonging to different clusters, and d_{max} is the maximum distance between any two points selected from the same cluster [15]. The highest value of s_{Dunn} indicates the optimal clustering solution among the considered candidates.

3 Using Multiple CVIs to Compare Clustering Solutions

3.1 Estimation of the Number of Clusters Using Internal CVI

The application of a CVI to determine the number of clusters of data is straightforward. Given a set of alternative clustering solutions, the preferred one is found by comparing the CVIs calculated for several candidate clustering solutions [8,16,17]. In order to find the number of clusters, a clustering algorithm is used with the number of clusters equal to $k \in \Omega$, where Ω is the ordered set of candidate cluster numbers. The preferred clustering solution is obtained by finding the value of k_{est} that maximizes the function $CVI(k)$ over all values from Ω. Under the assumption that the chosen clustering algorithm is able to cluster the data properly, then $k_{true} = \arg_{k \in \Omega} \max(CVI(k))$ is the sought number of clusters.

There is no single theoretically optimal CVI that is guaranteed to work well with all datasets. The reported empirical comparisons of CVI performance argue in favor of different algorithms [12,18,17,11,19]. In the absence of theoretically proven superiority of any CVI over the rest of the pack, we conclude that all CVIs capture certain aspects of a clustering solution that helps in comparing it to its alternatives, but none does it universally better than others.

The situation is strikingly similar to the dilemma of comparing and selecting best classifiers in pattern recognition, where the *no free lunch theorem* rules that there is no universally best classifier [8]. Multi-classifier ensembles have been

shown to offer a powerful alternative to single-classifier solutions [20] as a particularly effective option when the considered individual classifiers are diversified and accurate [9].

In this paper we approach the problem of estimating the number of clusters from a multi-classifier fusion perspective. From this angle, estimation of the number of clusters is a classification problem, where each class corresponds to a particular number of clusters in the considered dataset. We consider individual CVI algorithms as feature extractors. Resulting features can be used individually for cluster number estimation, as it is the common case; or can be used together using a combining fusion classifier.

3.2 Fusion Methods for Clustering Validity Indices

The multi classifier fusion can be realized by means of trained or rule-based combining classifiers. In general, trained fusion classifiers are considered to be more flexible and powerful than their heuristic, rule-based counterparts, given that an adequate training data corpus is available [21]. In the case of the CVI combination, it is not evident how to construct such a training corpus, and therefore heuristic fusion techniques are considered in this paper. In particular, we consider score-, and decision- level fusion schemes [20].

Score fusion-based methods. A combined score s_F is computed using i normalized CVI scores $[s_1, s_2, ..., s_i]$, as follows

- **SF-A**: $s_F = \frac{1}{i} \sum_{m=1}^{i} s_m$,
- **SF-G**: $s_F = (\Pi_{m=1}^{i} s_m)^{\frac{1}{i}}$,
- **SF-H**: $s_F = \frac{i}{\sum_{m=1}^{i} \frac{1}{s_m}}$,
- **SF-Med**: $s_F = \texttt{median}(\{s_1, s_2, ..., s_i\})$.

In order to perform a meaningful score-based fusion it is necessary to normalize the scores $[s_1, s_2, ..., s_i]$ to a common range. We used a min-max normalization, which scaled the scores to a $0 \leq s_i \leq 1$ range. The normalization was performed within each clustered dataset, across all candidate $k \in \Omega$. We also evaluated other score normalization schemes, including z-norm across $k \in \Omega$, as in [10], and global z-norm, where the mean and variance of s_i were estimated using a large range of possible clustering solutions. We obtained the best results with the min-max normalization and this scheme was used in the experiments reported. An example of score-based fusion using the four considered strategies is shown in Figure 1(b).

Decision fusion-based methods. Bezdek and Pal suggested the use of "some voting scheme" for CVIs [18]. Let $\Psi = \{1, 2, ..., n\}$ be a vector of class labels, each corresponding to estimated cluster number from Ω. A set of class decisions $d_i \in \Psi$, originating from i individual CVIs is obtained. If $m \in \Psi$ is a class label, then $d_i = \arg_m \max(s_i(m))$. Consequently, the decision fusion strategies we considered were

- **DF-A**: $d_F = round(\frac{1}{i}\sum_{m=1}^{i} d_m)$,
- **DF-G**: $d_F = round((\Pi_{m=1}^{i} d_m)^{\frac{1}{i}})$,
- **DF-Mod**: $d_F = \texttt{mode}(\{d_1, d_2, ..., d_i\})$,
- **DF-Med**: $d_F = \texttt{median}(\{d_1, d_2, ..., d_i\})$.

At first, computing the mean of decision values may seem counterintuitive. Indeed, in order for $DF-A$, $DF-G$ and $DF-Med$ to return meaningful results, the ordered set $\Omega = \{n_1, n_2, ..., n_m\}$ must be constructed so that $n_1 < n_2 < ... < n_m$. In this situation, the fusion algorithm will compensate for over- and underestimated k_{est}, given that enough single CVI decisions are involved. In our experiments $n_{i+1} - n_i = 1$ for all i, where i are indices of n in Ω. An example where decision fusion returns meaningful results is shown in Figure 1. In Figure 1(a), individual CVIs incorrectly return 9, 9 and 12 as the estimated number of clusters k_{est}, respectively. In Figure 1(b), the decision fusion strategies return $DF_A = DF_G = 10$ (correctly), and $DF_{Mod} = DF_{Med} = 9$.

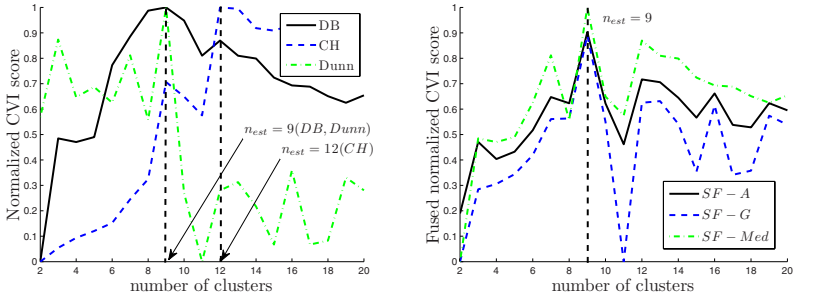


Fig. 1. Normalized CVI scores (a) and, fused scores using considered score-based fusion strategies (b), for $k_{true} = 10$ clusters. In the example shown, all of the CVIs (a) and fused scores (b) point out to the incorrect number of clusters $k_{est} = 9$. In this example, decision fusion strategies return $DF_A = 10$ and $DF_G = 10$, the correct number of clusters.

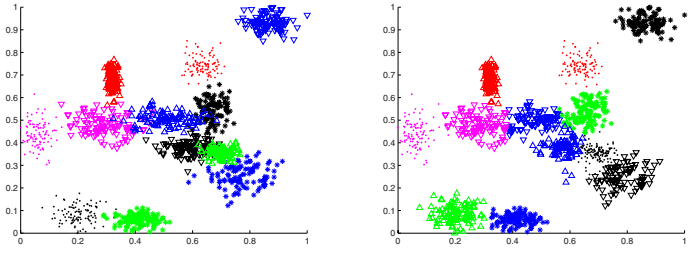
Decision rank fusion-based methods In our experiments we compared a number of decision rank fusion techniques. However, the results we obtained proved to be not competitive in comparison to other reported fusion strategies and we exclude them from this report due to space constraints.

4 Experimental Evaluation of CVI Fusion Techniques

Data generation and normalization. In order to evaluate the performance of the compared CVI fusion schemes we produced a large number of randomly-generated datasets. This choice allowed us to fully control the clustering tendency

present in the clustered data, as well as the data properties such as dimensionality n_{dim}, number of data clusters k_{true} and number of data points per cluster n_p. We also used a clustering algorithm that suits the shape of the actual data clusters, in order to reduce the impact of incorrect clustering on the observed CVI performance. This level of control is hard to achieve using scarce real data.

For each generated dataset, its n_{dim}, k_{true} and n_p were set. For each cluster a centroid vector and a variance vector were generated from a uniform multivariate distribution. Each mean vector was then scaled using a parameter p. So obtained, randomly generated mixture model was accepted only if all distances between the cluster centroids were greater than a threshold t (in this work $t = 0.1$). This procedure ensured that two or more clusters are not randomly placed on top of one another, making it hard or impossible to estimate the true number of clusters. For each cluster, n_p data points were randomly generated from a Gaussian distribution with variance and mean defined by the corresponding centroid and variance vector. Finally, the data was scaled to fit into a unit hypercube with dimensionality equal to the dimensionality of the generated dataset n_{dim}. Figure 2 shows an example of a generated 2-dimensional dataset.



(a) (b)

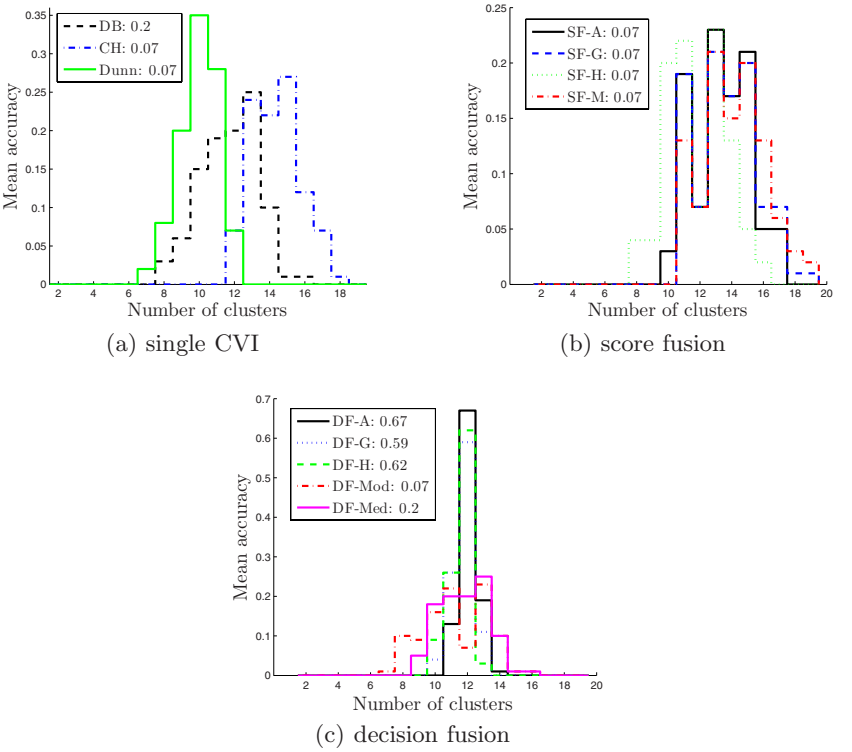
Fig. 2. Sample generated clustering problem for $p = 5$, $k_{true} = 12$ (a), and corresponding solution found by multiple k-means clustering with $k = k_{true}$ (b)

Evaluation method. In the experiments, we compared the accuracy of the cluster number estimation procedures involving single CVIs with the procedures involving considered multi-CVI fusion approaches described in Section 3.2. For each fixed data dimensionality n_{dim} and the number of clusters k_{true}, we generated a batch of 100 datasets, as described in Section 4.

The normalized data from each set of the batch was clustered using the k-means algorithm with Euclidian distance measure [8], with the candidate number of clusters $k = [2, 3, ..., 20]$. The clustering was repeated 10 times for each value of k, each time using a random initialization of prototype cluster centroid positions. The clustering solution with the least mean within-cluster sum of point-to-centroid distances was selected as the best for the given k. Then, the three CVIs (s_{DB}, s_{CH} and s_{Dunn}) were computed for each k. Consequently, for each batch of data we obtained 19 sets of 3 CVI scores, which were normalized

and fused using the methods described in Section 3.2. Finally, as the output for each clustered dataset we obtained a set of estimates k_{est} of the true number of clusters, as described in Section 3.1.

In the assumed classification framework, we compared the performance of the CVIs and CVI fusion methods by computing the accuracy of assigning a correct class label to the obtained CVIs. Figure 3 shows an example of the results obtained from one data batch of 100 random datasets, with dimensionality $n_{dim} = 10$, for $k_{true} = 12$. Each plot shows histograms of classification labels (decisions) k_{est} over 100 datasets, for single and fused CVIs. The histogram value corresponding to $k_{est} = k_{true}$ bin label is the recorded classification accuracy for a given method of cluster number estimation.



(a) single CVI (b) score fusion

(c) decision fusion

Fig. 3. Experimental results of a comparison of CVIs (a), and score- (b), and decision-based (c) CVI fusion methods, for $k_{true} = 12$ clusters, 19 classes ($k = [2, 3, ..., 20]$), averaged over 100 randomly generated datasets

Experimental results. We analyzed the performance of single CVIs versus CVI fusion-based methods for $k_{true} = [5, 6, ..., 15]$, $n_{dim} \in \{2, 10, 20\}$, for constant value of $p = 5$ and constant number of points per cluster $n_p = 20$. The

Table 1. Results of the experimental evaluation of the compared CVIs and CVI fusion strategies, for $p = 5$, $n_p = 20$, $n_{dim} = \{2, 10, 20\}$, the actual number of cluster $k_{true} = [5, 6, ..., 15]$, with 19 candidate number of clusters (classes), $k = [2, 3, ..., 20]$. Maximal accuracy per k_{true} is marked in bold font.

k_{true}	s_{DB}	s_{CH}	s_{Dunn}	$SF-A$	$SF-G$	$SF-H$	$DF-A$	$DF-G$	$DF-Mod$	$DF-Med$
Dimensionality $n_{dim} = 2$										
5	0.41	0.32	0.32	0.4	0.26	0.3	0.21	0.32	0.34	**0.42**
6	0.23	**0.44**	0.15	0.2	0.23	0.42	0.24	0.24	0.16	0.26
7	0.29	**0.41**	0.17	0.26	0.2	**0.41**	0.22	0.23	0.24	0.3
8	0.2	**0.41**	0.14	0.23	0.12	0.39	0.15	0.19	0.2	0.27
9	0.14	**0.45**	0.06	0.14	0.11	0.43	0.15	0.14	0.12	0.18
10	0.14	**0.28**	0.09	0.13	0.06	0.27	0.2	0.11	0.11	0.17
11	0.11	**0.25**	0.05	0.11	0.04	0.24	0.13	0.05	0.06	0.13
12	0.08	**0.21**	0.04	0.08	0.05	0.19	0.12	0.13	0.05	0.09
13	0.19	0.16	0.04	0.13	0.05	**0.22**	0.08	0.06	0.08	0.21
14	0.09	**0.14**	0	0.05	0.02	0.13	0.04	0.04	0.03	0.11
15	0.12	**0.16**	0.03	0.05	0.04	**0.16**	0.07	0.06	0.04	**0.16**
Dimensionality $n_{dim} = 10$										
5	0.98	0.98	0.97	0.97	0.94	0.9	**1**	**1**	0.98	0.98
6	0.89	0.89	0.88	0.89	0.89	0.85	**0.99**	**0.99**	0.89	0.89
7	0.73	0.75	0.75	0.75	0.78	0.68	**0.92**	0.91	0.75	0.75
8	0.44	0.49	0.49	0.49	0.5	0.48	**0.81**	0.76	0.49	0.5
9	0.34	0.32	0.32	0.32	0.36	0.3	**0.73**	0.67	0.32	0.34
10	0.23	0.2	0.2	0.2	0.22	0.2	**0.6**	0.54	0.2	0.23
11	0.28	0.15	0.15	0.15	0.17	0.14	**0.62**	0.51	0.15	0.28
12	0.25	0.18	0.18	0.18	0.2	0.17	**0.62**	0.54	0.18	0.25
13	0.21	0.05	0.05	0.05	0.08	0.05	**0.55**	0.43	0.05	0.21
14	0.16	0.03	0.03	0.03	0.09	0.03	**0.48**	0.31	0.03	0.16
15	0.14	0.05	0.03	0.03	0.22	0.05	**0.34**	0.29	0.03	0.14
Dimensionality $n_{dim} = 20$										
5	0.96	0.96	0.96	0.96	0.96	0.91	**1**	**1**	0.96	0.96
6	0.8	0.8	0.8	0.8	0.82	0.77	**0.97**	**0.97**	0.8	0.8
7	0.71	0.71	0.71	0.71	0.77	0.7	**0.93**	0.91	0.71	0.71
8	0.46	0.46	0.46	0.46	0.51	0.44	**0.87**	**0.87**	0.46	0.46
9	0.36	0.33	0.32	0.32	0.45	0.33	**0.85**	0.75	0.32	0.36
10	0.22	0.16	0.16	0.16	0.22	0.16	**0.71**	0.66	0.16	0.22
11	0.24	0.15	0.16	0.16	0.19	0.15	**0.78**	0.74	0.16	0.24
12	0.18	0.08	0.08	0.08	0.08	0.08	**0.64**	0.54	0.08	0.18
13	0.17	0.03	0.03	0.03	0.06	0.03	**0.61**	0.51	0.03	0.17
14	0.17	0.05	0.04	0.04	0.09	0.05	**0.47**	0.38	0.04	0.17
15	0.18	0.06	0.02	0.02	0.15	0.05	**0.45**	**0.45**	0.05	0.18

number of classes was 19: candidate $k = [2, 3, ..., 20]$. The results of the experiment are given in Table 1.

5 Discussion and Conclusions

The results of the experiments reported in Section 4 show that combining CVIs is a viable way of increasing the robustness and accuracy of cluster number estimation in comparison with using a single CVI. In the reported experiments, the CVI fusion methods outperformed single CVIs in all cases, where all CVIs used in fusion were similarly accurate. An underperforming CVI often compromised the accuracy of the fusion results, as in Table 1 for $n_{dim} = 2$. This effect is due to the fact that the used heuristic fusion methods have no weighting mechanism that could identify and marginalize, or decrease the contribution of the

underperforming CVI in the fusion process. The detrimental effect of using an underperforming CVI was particularly pronounced in the case of the score fusion techniques.

The decision-based fusion rules showed to be in general more robust and yielded higher accuracy than the score-based fusion methods. It can be explained by the fact that for all CVIs, the deviations from their corresponding maximal value $|s_i(k_{est}) - s_i(k \neq k_{est})|$ were of different scale and therefore not comparable quantitatively. However, in the decision-based fusion this deviation is comparable qualitatively: only the information of $\arg_{k \in \Omega} \max s_i(k)$ is preserved.

The overall best-performing scheme in the experiments reported was the mean-rule decision fusion scheme DF_A, closely followed by the DF_G scheme. This result can be attributed to the fact that the decision-based schemes are capable of compensating for CVIs that over- and underestimate k_{est}, while being insensitive to score normalization. This result shows that the individual CVIs indeed make different errors and therefore are diversified enough to merit an application of a multi-classifier fusion scheme. We anticipate that a larger number of fused CVIs would reinforce this effect.

In our experiments, the benefits of using the CVI fusion techniques became evident with the increase of the number of clusters within the hypercube, and with the increase of the dimensionality of the clustered data. In fact, as Table 1 shows, for low k_{true} (good class separation) the benefit of using fusion is immaterial. At the same time, for high-dimensional data ($n_{dim} = 20$) and high value of k_{true}, these benefits are significant, with the DF_A and $DF - G$ schemes yielding more than double the accuracy of the best individual CVI.

The experiments reported in this paper include only a limited number of examples to support the claim that using a fusion of multiple CVIs can result in higher accuracy of cluster number estimation than using a single CVI. Obviously, a single CVI that is well-selected for the problem at hand, can produce accuracy competitive to, of higher than, the fusion result. This effect, well-known from rule-based classifier fusion, is particularly relevant if other used CVIs yield low accuracy. However, in many practical applications the shape and number of clusters is unknown, making a-priori selection of a suitable single CVI impossible. This is particularly the case if the data is of high dimensionality, cannot be easily visualized, and contains many clusters with few members each. Our results suggest that in such scenarios a fusion of CVIs for estimation of the number of clusters can be successfully applied. In the future, we intend to validate our findings using real datasets and using different clustering algorithms.

References

1. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
2. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 255–323 (1999)
3. Dudoit, S., Fridlyand, J.: A prediction-based resampling method for estimating the number of clusters in a dataset. Genome Biology 3(7), 0036.1–0036.21 (2002)

4. Yao, Z., Choi, B.: Automatically discovering the number of clusters in web page datasets. In: Proceedings of the 2005 International Conference on Data Mining, pp. 3–9 (2005)
5. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: Pacific Symposium on Biocomputing, pp. 6–17 (2002)
6. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Computation 16(6), 1299–1323 (2004)
7. Zhang, J., Modestino, J.W.: A model-fitting approach to cluster validation with application to stochastic model-based image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 12(10), 1009–1017 (1990)
8. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 2nd edn. Elsevier, Amsterdam (2003)
9. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. John Wiley & Sons, Chichester (2004)
10. Machado, J.B., Amaral, W.C., Campello, R.: Design of obf-ts fuzzy models based on multiple clustering validity criteria. In: Proc. of the 19th IEEE International Conference on Tools with Artificial Intelligence (2007)
11. Maulik, U., Bandyopadhyay, S.: Performance evaluation of some clustering algorithms and validity indices. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(12), 1650–1654 (2002)
12. Milligan, G.W., Cooper, M.C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika 50, 159–179 (1985)
13. Davies, D.L., Bouldin, D.W.: A clustering separation measure. IEEE Transactions on PAMI 1, 224–227 (1979)
14. Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistic 3, 1–27 (1974)
15. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics 3, 32–57 (1973)
16. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: part i. SIGMOD Rec. 31(2), 40–45 (2002)
17. Halkidi, M., Vazirgiannis, M.: Clustering validity assessment: Finding the optimal partitioning of a data set. In: Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 187–194 (2001)
18. Bezdek, J.C., Pal, N.R.: Some new indexes of cluster validity. IEEE Trans. on Systems, Man, and Cybernetics B 28(3), 301–315 (1998)
19. Shim, Y., Chung, J., Choi, I.C.: A comparison study of cluster validity indices using a nonhierarchical clustering algorithm. In: Proc. of the 2005 Int. Conf. on Comp. Intelligence for Modelling, Control and Automation, and Int. Conf. on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC 2005) (2005)
20. Kittler, J., Hataf, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
21. Duin, R.P.W.: The combining classifier: to train or not to train? In: Proceedings of the International Conference on Pattern Recognition, Quebec, Canada (2002)

"Good" and "Bad" Diversity in Majority Vote Ensembles

Gavin Brown[1] and Ludmila I. Kuncheva[2]

[1] School of Computer Science, University of Manchester, UK
gavin.brown@cs.manchester.ac.uk
[2] School of Computer Science, Bangor University, UK
l.i.kuncheva@bangor.ac.uk

Abstract. Although diversity in classifier ensembles is desirable, its relationship with the ensemble accuracy is not straightforward. Here we derive a decomposition of the majority vote error into three terms: average individual accuracy, "good" diversity and "bad diversity". The good diversity term is taken out of the individual error whereas the bad diversity term is added to it. We relate the two diversity terms to the majority vote limits defined previously (the patterns of success and failure). A simulation study demonstrates how the proposed decomposition can be used to gain insights about majority vote classifier ensembles.

1 Introduction

The topic of 'diversity' has been a favourite buzzword in the multiple classifier systems community for well over a decade [1,2]. Numerous diversity measures have been proposed, measured and maximised, all with the goal to increase ensemble performance by balancing "individual accuracy" against "diversity". It is therefore ironic that after so much time and effort, we still have no uniquely agreed definition for "diversity" [3,4].

In this work we adopt the perspective that a diversity measure should be *naturally* defined as a consequence of two decisions in the design of the ensemble learning problem: the choice of *error function*, and the choice of *combiner function*. When discussing 'diversity', we often overlook these, implicitly adopting the *zero-one loss* (classification error) function, and the *majority vote* combiner. The principle of combining multiple predictions can of course be applied in many learning scenarios, such as regression [5], or unsupervised learning [6]. Depending on the situation, it may be appropriate to adopt other loss functions, such as the cross-entropy, or the squared loss. We might also consider other combiner functions, such as the average or product rule.

These two design decisions turn out to be very interesting for the diversity debate. It turns out that, for particular choices of error/combiner function, a definition of diversity *naturally* emerges. For a real-valued target y, if the

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 124–133, 2010.

ensemble is a linear combiner $\bar{f} = \frac{1}{T}\sum_t f_t$, and we assess it with the squared loss function, it is well appreciated that

$$(\bar{f} - y)^2 = \frac{1}{T}\sum_{t=1}^{T}(f_t - y)^2 - \frac{1}{T}\sum_{t=1}^{T}(f_t - \bar{f})^2. \tag{1}$$

The decomposition [7] states that the squared loss of the ensemble is *guaranteed* to be less than or equal to the average squared loss of the individuals. The difference is what we might call a 'diversity' term, measuring the average squared loss of the individuals from the ensemble prediction. The decomposition *only* holds when the ensemble combiner $\bar{f}$ is the (weighted) arithmetic mean of the individual predictions. In classification problems, it is common to minimize the *cross-entropy* loss function, which is derived from a Kullback-Leibler divergence. If we combine our multiple class probability estimates with a normalized geometric mean[1], then we have the decomposition,

$$D_{KL}(y||\bar{f}) = \frac{1}{T}\sum_{t=1}^{T} D_{KL}(y||f_t) - \frac{1}{T}\sum_{t=1}^{T} D_{KL}(\bar{f}||f_t). \tag{2}$$

The KL divergence of the ensemble from a target distribution, is *guaranteed* to be less than or equal to the average divergence of the individual estimates [8]. The difference is again what we might appreciate as diversity, measuring the average divergence of the individuals from the geometric mean.

In this paper we ask the question, *can a similar decomposition hold for classification error and majority vote combiners?* The answer turns out to be less straightforward than the above, and results in *two* diversity terms, which can both help and hinder the ensemble performance. In section 2 we present the main result, and in section 3 relate it to the patterns of 'success' and 'failure' in classifier combining [9]. Section 4 presents a simulation study monitoring the behaviour of the diversity terms in different situations.

2 Decomposition of the Majority Vote Error

Considers a two-class problem with class labels in the set $\{-1, +1\}$. Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_T\}$ be a set of classifiers, where T is odd. Denote by $h_t(\mathbf{x}) \in \{-1, +1\}$ the output of classifier ϕ_t for input $\mathbf{x}$. Let $y(\mathbf{x}) \in \{-1, +1\}$ be the true label of $\mathbf{x}$. The zero-one loss of ϕ_t for $\mathbf{x}$ is

$$e_t(\mathbf{x}) = \begin{cases} 0, & y(\mathbf{x}) = h_t(\mathbf{x}) \\ 1, & y(\mathbf{x}) \neq h_t(\mathbf{x}) \end{cases} = \frac{1}{2}\left(1 - y(\mathbf{x})\, h_t(\mathbf{x})\right). \tag{3}$$

If Φ is taken to be a classifier ensemble, the majority vote output for input $\mathbf{x}$ is

$$H(\mathbf{x}) = \text{sign}\left(\frac{1}{T}\sum_{t=1}^{T} h_t(\mathbf{x})\right), \tag{4}$$

[1] Equivalent to the product rule.

where $H(\mathbf{x}) \in \{-1, +1\}$. The zero-one loss of the ensemble for input $\mathbf{x}$ is

$$e_{\text{maj}}(\mathbf{x}) = \frac{1}{2}\ (1 - y(\mathbf{x})\ H(\mathbf{x}))\,. \tag{5}$$

Define also the *disagreement* between classifier ϕ_t and the ensemble as

$$\delta_t(\mathbf{x}) = \frac{1}{2}\ (1 - h_t(\mathbf{x})\ H(\mathbf{x}))\,. \tag{6}$$

Take the difference between the ensemble loss and the average individual loss,

$$\begin{aligned}
\Delta &= e_{\text{maj}}(\mathbf{x}) - e_{\text{ind}}(\mathbf{x}) \\
&= \frac{1}{2}\ (1 - y(\mathbf{x})\ H(\mathbf{x})) - \frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\ (1 - y(\mathbf{x})\ h_t(\mathbf{x})) && (7)\\
&= \frac{1}{2} - \frac{1}{2}y(\mathbf{x})\ H(\mathbf{x}) - \frac{1}{2} + \frac{1}{2T}\sum_{t=1}^{T} y(\mathbf{x})\ h_t(\mathbf{x}) && (8)\\
&= -y(\mathbf{x})\ H(\mathbf{x})\frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\left(1 - \frac{h_t(\mathbf{x})}{H(\mathbf{x})}\right). && (9)
\end{aligned}$$

Since $H(\mathbf{x}) \in \{-1, +1\}$, we can write $\frac{h_t(\mathbf{x})}{H(\mathbf{x})} = h_t(\mathbf{x})H(\mathbf{x})$, so we have:

$$\begin{aligned}
\Delta &= -y(\mathbf{x})\ H(\mathbf{x})\frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}\,(1 - h_t(\mathbf{x})\ H(\mathbf{x})) && (10)\\
&= -y(\mathbf{x})\ H(\mathbf{x})\frac{1}{T}\sum_{t=1}^{T}\delta_t(\mathbf{x}). && (11)
\end{aligned}$$

This demonstrates that *the difference between the majority voting loss and the average individual loss can be directly expressed in terms of the average classifier disagreement*. In summary:

$$e_{\text{maj}} = e_{\text{ind}} - y(\mathbf{x})\ H(\mathbf{x})\frac{1}{T}\sum_{t=1}^{T}\delta_t(\mathbf{x}) \tag{12}$$

Equation (12) is the zero-one loss of a majority vote on a single datapoint $\mathbf{x}$. To calculate the majority vote *classification error*, E_{maj}, we need to integrate with respect to the probability density function $p(\mathbf{x})$. In the following we also take advantage of the fact that $y(\mathbf{x})H(\mathbf{x}) = +1$ on datapoints where the ensemble is correct, and $y(\mathbf{x})H(\mathbf{x}) = -1$ where it is incorrect.

$$\begin{aligned}
E_{\text{maj}} &= \int_{\mathbf{x}} e_{\text{ind}}(\mathbf{x}) - \int_{\mathbf{x}} y(\mathbf{x})\ H(\mathbf{x})\frac{1}{T}\sum_{t=1}^{T}\delta_t(\mathbf{x}) && (13)\\
&= \int_{\mathbf{x}} e_{\text{ind}}(\mathbf{x}) - \underbrace{\int_{\mathbf{x}+}\frac{1}{T}\sum_{t=1}^{T}\delta_t(\mathbf{x})}_{\text{good diversity}} + \underbrace{\int_{\mathbf{x}-}\frac{1}{T}\sum_{t=1}^{T}\delta_t(\mathbf{x})}_{\text{bad diversity}} && (14)
\end{aligned}$$

Here the integral has been separated for two subspaces of the data: $\mathbf{x}+$ where the ensemble is correct, and $\mathbf{x}-$ where it is incorrect.

Equation (14) prompts the following interpretation. The majority vote error has a direct relationship with two components of diversity, measured by the disagreement between the classifier decision $h_t(\mathbf{x})$ and the ensemble decision $H(\mathbf{x})$. We label the two diversity components "good" and "bad" diversity. The good diversity measures the disagreement on datapoints where the ensemble is *correct*—and due to the negative sign, any disagreement on these points *increases* the gain relative to the average individual error. The bad diversity measures the disagreement on datapoints where the ensemble is *incorrect*—here, the diversity term has a positive sign, so any disagreement *reduces* the gain relative to individual error.

Another way to think of this is as the number of votes "wasted" by the ensemble members. For an arbitrary datapoint $\mathbf{x}$, assume the ensemble is already correct. If there is little disagreement, then several votes have been 'wasted', since the same correct decision would have been taken had some classifiers changed their votes. This is the "good" diversity term, measuring disagreement where the ensemble is already correct—more disagreement equates to fewer wasted votes. The "bad" diversity term measures the opposite: disagreement where the ensemble is incorrect—any disagreement equates to a wasted vote, as the individual did not have any effect on the ensemble decision. Thus, increasing good diversity and decreasing bad diversity is equivalent to reducing the number of 'wasted' votes.

We now provide alternative formulations of the good/bad diversity terms, that will facilitate a link to the patterns of "success" and "failure" previously used to study the limits of majority voting [9]. Let $\mathbf{v}$ be a T-dimensional binary random variable such that $v_t = 1$ means that classifier ϕ_t labels $\mathbf{x}$ correctly, and $v_t = 0$ means that the assigned label is incorrect. Construct a T-element vector $\mathbf{1} = [1, 1, \ldots, 1]^T$. Then the scalar product $\mathbf{v}^T\mathbf{1}$ will be the number of correct votes in $\mathbf{v}$. The ensemble is correct when $\mathbf{v}^T\mathbf{1} \geq \frac{T+1}{2}$. The "good diversity" in this case is the number of incorrect votes, i.e., $T - \mathbf{v}^T\mathbf{1}$. When the ensemble is incorrect, the "bad diversity" is the number of correct votes, i.e., $\mathbf{v}^T\mathbf{1}$. The integral across the feature space can be replaced by a summation across all possible values of $\mathbf{v}$ leading to

$$E_{\text{maj}} = E_{\text{ind}} - \underbrace{\frac{1}{T} \sum_{\mathbf{v}^T\mathbf{1} \geq \frac{T+1}{2}} (T - \mathbf{v}^T\mathbf{1})\, p(\mathbf{v})}_{\text{good diversity}} + \underbrace{\frac{1}{T} \sum_{\mathbf{v}^T\mathbf{1} < \frac{T+1}{2}} \mathbf{v}^T\mathbf{1}\, p(\mathbf{v})}_{\text{bad diversity}}. \quad (15)$$

In the following section we show these terms can be related to the patterns of 'success' and 'failure' [9] for voting ensembles.

3 Patterns of Success and Failure

The *pattern of success* and the *pattern of failure* were introduced as special cases illustrating the limits of the majority vote [9]. Given a set of classifiers of the same individual accuracy p, the pattern of *success* is the most favorable distribution of the correct votes, leading to the largest improvement on p. To achieve this, we define a probability distribution over all possible combinations of correct/incorrect votes. Each combination where exactly $\frac{T+1}{2}$ votes are correct, appears with probability α. The only other combination of votes with non-zero probability is when all votes are incorrect. In this pattern, there are no wasted votes, as each vote is needed to ensure the smallest majority of correct votes.

For example, consider three classifiers, $\Phi = \{\phi_1, \phi_2, \phi_3\}$, each of accuracy $p = 0.6$. Denote again a correct vote by 1, and an incorrect vote by 0. The pattern of success is constructed by assigning probability α to each of the three combinations of votes 011, 101, 110, and probability $1 - 3\alpha$ to 000. Since each of the classifiers will be accurate in two of these combinations, to make up the individual accuracy of 0.6, α must be 0.3. The majority vote error for this example is $1 - 3\alpha = 0.1$ [9]. This calculation can now be easily demonstrated using equation (15).

$$\begin{aligned} E_{\text{maj}} &= E_{\text{ind}} - \frac{1}{3}(3 \times (3-2) \times \alpha) + \frac{1}{3}(0 \times (1-3\alpha)) \\ &= 0.4 - \frac{1}{3} \times 3 \times 0.3 = 0.4 - 0.3 = 0.1. \end{aligned}$$

In the pattern of failure, the correct votes are distributed in such a way that they are one shy of majority, so the largest quantity of correct votes are wasted. In the example above, the pattern of failure is constructed by assigning probabilities β to combination of votes 001, 010 and 100, and probability $1-3\beta$ to combination 111. Each of the three classifiers is accurate only in combination 111 (probability $1 - 3\beta$) and one of the combinations with one correct vote (probability β). To ensure that the individual accuracy is $p = 0.6$, we have $1 - 2\beta = 0.6$, hence $\beta = 0.2$. Using equation (15) again,

$$\begin{aligned} E_{\text{maj}} &= E_{\text{ind}} - \frac{1}{3}((3-3) \times (1-3\beta)) + \frac{1}{3}(3 \times 1 \times \beta)) \\ &= 0.4 + \frac{1}{3} \times 3 \times 0.2 = 0.4 + 0.2 = 0.6. \end{aligned}$$

In the general definition of the pattern of success, each of the possible $\binom{T}{\frac{T+1}{2}}$ vote combinations where the ensemble is correct appears with probability α, and the combination where $\mathbf{v}^T\mathbf{1} = 0$ appears with probability $1 - \binom{T}{\frac{T+1}{2}}\alpha$. All other vote combinations have zero probability. Then, substituting these values into (15), we have

$$E_{\text{maj}} = (1-p) - \frac{1}{T}\left(\binom{T}{\frac{T+1}{2}} \times \frac{T-1}{2} \times \alpha\right) \tag{16}$$

To ensure that the individual accuracy of all classifiers is p, α must satisfy

$$\alpha = \frac{p}{\binom{T-1}{\frac{T-1}{2}}}. \tag{17}$$

Substituting (17) in (16) and applying simple algebraic manipulation, we recover exactly the expression for the upper bound on majority vote accuracy, defined in [9]

$$E_{\text{maj}} = \max\left\{0, \quad E_{\text{ind}} - \frac{T-1}{T+1}(1 - E_{\text{ind}})\right\}. \tag{18}$$

In a similar way, substituting the appropriate figures for the pattern of failure, (15) leads to the general expression for the lower bound on majority vote error:

$$E_{\text{maj}} = E_{\text{ind}}\left(1 + \frac{T-1}{T+1}\right). \tag{19}$$

Equation (15) gives a direct relationship between the majority vote error and the patterns of success/failure, where the bad and the good diversity partly "neutralise" one another.

4 Simulation Experiment

At this stage it is difficult to recommend a way to use the decomposition straightforwardly in a classifier ensemble algorithm. The simulation study illustrates how the decomposition can help to gain insight into the internal workings of majority vote classifier ensembles.

4.1 Data

The 2-class, 2-d data set is shown in Figure 1. The data is generated uniformly in the unit square and the classes are labelled according to a rotated checker board[2]. The problem has a Bayes error of 0.

4.2 Experimental Protocol

The ensemble was sampled from a pool of linear classifiers. Each classifier was constructed by drawing a random line through a point in the unit square. The two sides of the line were labelled in the two classes in the way that gave accuracy greater than 0.5. Thus the classifiers were only slightly better than chance. The following steps were carried out

[2] Matlab code for N data points, uniform distribution, checkerboard with side a, rotated by θ:

```
function[d,labd]=gendatcb(N,a,theta)
d=rand(N,2);
d_transformed=[d(:,1)*cos(theta)-d(:,2)*sin(theta),d(:,1)*sin(theta)+d(:,2)*cos(theta)];
s=ceil(d_transformed(:,1)/a)+floor(d_transformed(:,2)/a);labd=2-mod(s,2);
```

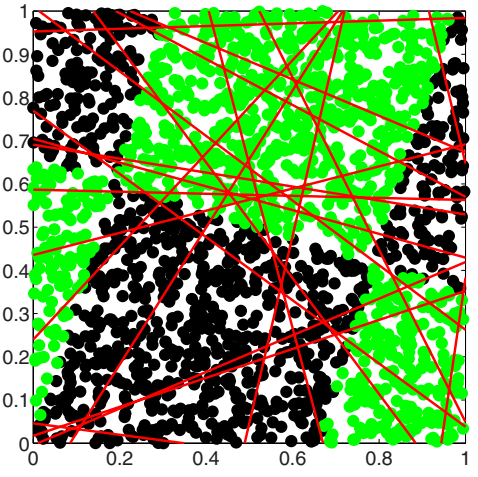


Fig. 1. Rotated checker board data and the boundaries of 20 random linear classifiers. The checker board was generated with side $a = 0.63$ and rotation angle $\theta = 0.3$.

1. Generate an initial pool Φ consisting of 2000 random linear classifiers.
2. The ensemble size T was varied as

$$T \in \{1, 3, 5, 9, 13, 19, 31, 51, 71, 101, 201, 501, 1001\}.$$

3. Generate and test fifty ensembles for each value of T :
   (a) T classifiers were sampled without replacement from the pool Φ.
   (b) A testing data set of 1000 2-d points was generated.
   (c) The testing accuracy of the ensemble, the average individual accuracy, and the two diversity terms were estimated and stored. The ensemble labelled the data points through the majority vote.
4. The stored values were averaged across the 50 runs.

To evaluate the effect of the individual accuracy on the ensemble accuracy and the diversity terms, the above protocol was repeated for a new "selected" pool of classifiers, engineered so as to have individual accuracy significantly better than random. The procedure was to first generate 8000 random classifiers, then select the 2000 with highest accuracy.

4.3 Results

To evaluate the success of the ensemble with increasing T, the majority vote errors for the random and the selected ensembles are plotted in Figure 2. As expected, the error drops with T. The "Random" ensembles show better improvement on the individual error rate but the ensemble error does not reach the one when sampling the ensemble from the selected set.

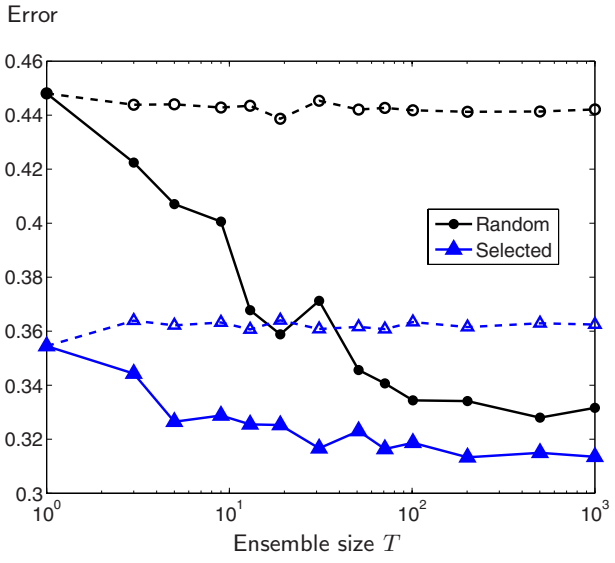


Fig. 2. Majority vote error (solid lines) and average individual error (dotted lines) versus the ensemble size T. "Random" corresponds to the original classifier pool, where the individual classifiers are only slightly better than chance. "Selected" corresponds to the selected pool, engineered to have higher accuracy.

Figure 3 shows the scatter plot of the Good diversity versus Bad diversity terms. Three values of T were chosen for the illustration, 3, 31, and 1001, shown with different markers. Each point represents one ensemble, thus there are 50 points for each marker, for each of the 50 trials. The diagonal line corresponds to when good = bad diversity, in which case they cancel each other out, and ensemble error is just equal to the individual error. Points *above* the diagonal line show ensembles where good diversity is larger than bad diversity, therefore the ensemble improves on the average individual error. The improvement of the ensemble with respect to the average individual error can be read off the plot as the vertical distance from the point to the diagonal line.

The ensembles in subplot (a) are more dispersed than those in plot (b). This reveals that both good and bad diversities reach larger values for the random pool than for the selected pool of classifiers for the same ensemble size T. This can be explained with the fact that higher individual error rate allows for a "weak" majority, where the ensemble is correct but there are many incorrect votes as well. This effect is especially visible for $T = 3$. While the "random" scenario offers a range of very good and very bad ensembles (scattered far above or below the diagonal line), the "selected" pool has a modest range of ensembles. For larger T, both ensemble pools produce compact clusters of points suggesting that the good-bad diversity ratio stabilises with T. Even though the clusters for $T = 1001$ are compact, they are positioned differently with respect to the diagonal line.

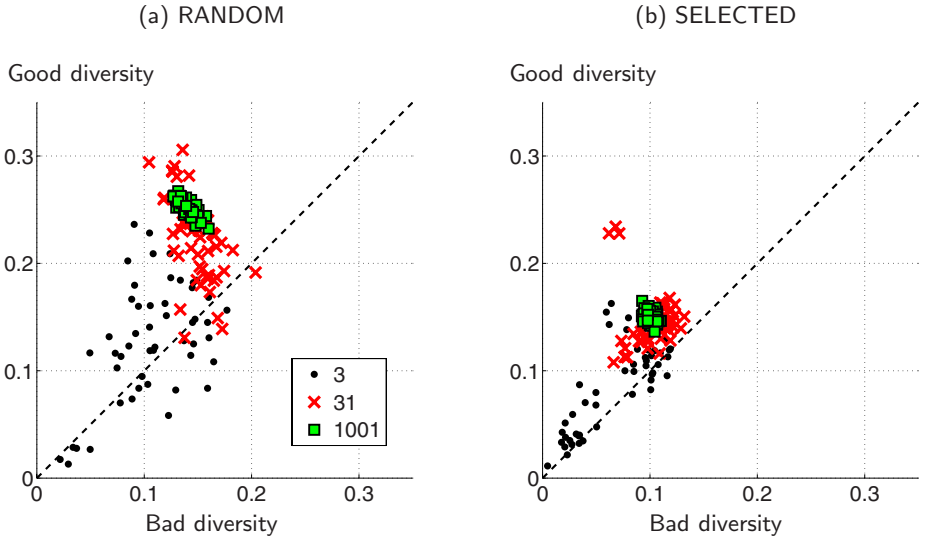


Fig. 3. Scatterplots of good diversity versus bad diversity terms for the "random" and the "selected" pools of classifiers for 3 values of T

The ensembles sampled from the "random" pool are further up compared to these sampled from the "selected" pool, which is mirrored in the larger versus smaller improvement on the individual errors in the rightmost points in Figure 2. While the improvement on the individual error can be gauged from Figure 2, the spread of the values of the good and bad diversity cannot.

Interestingly, the points for $T = 3$ and $T = 31$ are not shaped as coherent clusters but are rather split into sub-clusters (more visible in subplot (b)). The group of crosses far above the main cluster in subplot (b) corresponds to ensembles with dramatic improvement on the individual error rates (close to the pattern of success). This may give a lead towards creating ensembles that magnify good diversity while keeping the bad diversity at bay.

5 Conclusions

In this paper we adopted the perspective that a diversity measure should be naturally derived as a direct consequence of two factors: the loss function of interest, and the combiner function. We presented a decomposition of the classification error, using the majority vote combiner, into three terms: individual accuracy, 'good' diversity, and 'bad' diversity. A larger value of the good diversity reduces the majority vote error, whereas a larger value of bad diversity *increases* the error. We showed a direct relation of these concepts to the upper/lower limits defined on majority voting error [9]. A simulation study illustrated that the diversity terms tend to exhibit a large variance in smaller ensembles, and stabilize with very large ensembles.

The decomposition lends direct support not only to the existing theory of majority voting [9], but also to existing algorithms. The DECORATE algorithm [10] uses artificially constructed data examples to induce a diversity in majority voting by making the individuals disagree wherever possible with the ensemble. The results of this paper suggests it may be possible to construct more targeted algorithms, which directly magnify the "good" diversity while suppressing the "bad" diversity.

References

1. Brown, G.: Ensemble Learning. In: Encyclopedia of Machine Learning. Springer Press, Heidelberg (2010)
2. Kuncheva, L.: Combining pattern classifiers: methods and algorithms. Wiley-Interscience, Hoboken (2004)
3. Kuncheva, L.: That elusive diversity in classifier ensembles. In: Perales, F.J., Campilho, A.C., Pérez, N., Sanfeliu, A. (eds.) IbPRIA 2003. LNCS, vol. 2652, pp. 1126–1138. Springer, Heidelberg (2003)
4. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Information Fusion 6(1), 5–20 (2005)
5. Brown, G., Wyatt, J., Tiňo, P.: Managing diversity in regression ensembles. Journal of Machine Learning Research 6, 1650 (2005)
6. Strehl, A., Ghosh, J.: Cluster ensembles: a knowledge reuse framework for combining multiple partitions. The Journal of Machine Learning Research 3, 583–617 (2003)
7. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in neural information processing systems, pp. 231–238 (1995)
8. Heskes, T.: Bias/variance decompositions for likelihood-based estimators. Neural Computation 10(6), 1425–1433 (1998)
9. Kuncheva, L., Whitaker, C., Shipp, C., Duin, R.: Limits on the majority vote accuracy in classifier fusion. Pattern Analysis & Applications 6(1), 22–31 (2003)
10. Melville, P., Mooney, R.: Creating diversity in ensembles using artificial data. Information Fusion 6(1), 99–111 (2005)

Multi-information Ensemble Diversity

Zhi-Hua Zhou[1] and Nan Li[2]

[1] National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
[2] School of Mathematical Sciences
Soochow University, Suzhou 215006, China
zhouzh@lamda.nju.edu.cn, linan@suda.edu.cn

Abstract. Understanding ensemble diversity is one of the most important fundamental issues in ensemble learning. Inspired by a recent work trying to explain ensemble diversity from the information theoretic perspective, in this paper we study the ensemble diversity from the view of *multi-information*. We show that from this view, the ensemble diversity can be decomposed over the component classifiers constituting the ensemble. Based on this formulation, an approximation is given for estimating the diversity in practice. Experimental results show that our formulation and approximation are promising.

1 Introduction

It is well-known that in order to build a good ensemble, the component classifiers should be *accurate* as well as *diverse*. There are effective processes for estimating the accuracy of component classifiers, however, measuring diversity is not easy since there is no generally accepted formal definition. During the past decade, many diversity measures have been designed; to name a few, the *Q-statistics* [11], the *disagreement* [9], the *double-fault* [7], the κ-*statistic* [5], etc. However, it has been disclosed that existing diversity measures are suspect [10].

Recently, Brown [2] investigated the ensemble diversity from an information theoretic perspective. He found that the ensemble *mutual information* can be naturally expanded into 'accuracy' and 'diversity' terms, and the ensemble diversity exists at multiple orders of correlation. We believe that this is an important step towards the understanding of ensemble diversity. However, the expressions of that information theoretic diversity and its terms, especially the involved *interaction information*, are quite complicated, and there is no proposal of effective process for estimating the multiple orders of correlation in practice.

Inspired by Brown's work [2], in this paper we also study the ensemble diversity from an information theoretic perspective. From the view of *multi-information*, we propose a new formulation where the ensemble diversity and the related terms are simpler. This formulation enables to decompose the diversity over the component classifiers. Based on the formulation, we give an approximation for estimating the ensemble diversity in practice. Experiments show that our formulation and approximation are promising.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 134–144, 2010.

The rest of this paper is organized as follows. Section 2 briefly reviews some basics of information theory and Brown's study. Section 3 introduces our formulation based on multi-information. Section 4 presents an approximation. Section 5 reports on experiments. Finally, Section 6 concludes.

2 Background

The fundamental concept of information theory is the *entropy*, which is a measure of uncertainty. For a variable X, its entropy $H(X)$ is defined as $\sum_x p(x)\log(p(x))$, where x is the value of X, and $p(x)$ is the probability distribution.

Based on the concept entropy, the dependence among multiple variables can be measured by mutual information and its multivariate generalizations. Denote n variables $X_1, \cdots, X_n$ as $X_{1:n}$ and another variable as Y, then,

– *Mutual information* and *conditional mutual information* [4]:

$$I(X_1; X_2) = \sum\nolimits_{x_1,x_2} p(x_1, x_2) \log \frac{p(x_1, x_2)}{p(x_1)p(x_2)} \tag{1}$$

$$I(X_1; X_2 \mid Y) = \sum\nolimits_{y,x_1,x_2} p(y)p(x_1, x_2 \mid y) \log \frac{p(x_1, x_2 \mid y)}{p(x_1 \mid y)p(x_2 \mid y)} \tag{2}$$

– *Multi-information* and *conditional multi-information* [15,14,13]:

$$\mathcal{I}(X_{1:n}) = \sum\nolimits_{x_{1:n}} p(x_1, \cdots, x_n) \log \frac{p(x_1, \cdots, x_n)}{p(x_1)p(x_2)\cdots p(x_n)} \tag{3}$$

$$\mathcal{I}(X_{1:n} \mid Y) = \sum\nolimits_{y,x_{1:n}} p(y)p(x_{1:n} \mid y) \log \frac{p(x_{1:n} \mid y)}{p(x_1 \mid y)\cdots p(x_n \mid y)} \tag{4}$$

– *Interaction information* [12]:

$$I(\{X_{1:n}\}) = \begin{cases} I(X_1, X_2) & \text{for } n = 2 \\ I(\{X_{1:n-1}\} \mid X_n) - I(\{X_{1:n-1}\}) & \text{for } n \geq 3 \end{cases} \tag{5}$$

where $p(x_{1:n})$ is the joint distribution of $X_{1:n}$, $p(x)$ and $p(y)$ are the marginal distributions, and $p(\cdot \mid \cdot)$'s are the conditional distributions. [1]

As described above, *mutual information* measures the mutual dependence of two variables, while both *multi-information* and *interaction information* are its multivariate generation which express the dependence among multiple variables. Like mutual information, multi-information is nonnegative and equals zero if and only if all the variables are independent. Interaction information, however, can be negative; this has likely encumbered its wide application as an information measure.

In ensemble learning, suppose there is a set of classifiers $S = \{X_1, \cdots, X_m\}$ and the target class is Y; our objective is to find a combination function g

[1] We will use $p(\cdot)$ to denote different distributions when the meaning is clear.

that minimizes the probability of error prediction $p\left(g\left(X_{1:m}\right) \neq Y\right)$. Brown [2] bounded the probability of error by two inequalities [6,8], that is,

$$\frac{H(Y) - I(X_{1:m}; Y) - 1}{\log(|Y|)} \leq p\left(g\left(X_{1:m}\right) \neq Y\right) \leq \frac{H(Y) - I(X_{1:m}; Y)}{2}. \quad (6)$$

Thus, to minimize the prediction error, the mutual information $I(X_{1:m}; Y)$ should be maximized. Subsequently, an expansion of $I(X_{1:m}; Y)$ was given based on [3, Thm 1], and the *information theoretic diversity*, *redundancy* and *conditional redundancy* were defined. Denote T_k as a set of size k. Then,

$$\begin{aligned} I(X_{1:m}; Y) &= \underbrace{\sum\nolimits_{i=1}^{m} I(X_i; Y)}_{\text{relevancy}} + \underbrace{\sum\nolimits_{k=2}^{m} \sum\nolimits_{T_k \subseteq S} I(\{T_k \cup Y\})}_{\text{information theoretic diversity}} \quad (7) \\ &= \sum\nolimits_{i=1}^{m} I(X_i; Y) + \underbrace{\sum\nolimits_{k=2}^{m} \sum\nolimits_{T_k \subseteq S} I(\{T_k\}|Y)}_{\text{conditional redundancy}} - \underbrace{\sum\nolimits_{k=2}^{m} \sum\nolimits_{T_k \subseteq S} I(\{T_k\})}_{\text{redundancy}}. \end{aligned}$$

As shown above, the information theoretic diversity naturally emerges as an expression of the interaction information. It can also be found that the ensemble diversity exists at multiple orders of correlation. Since computing high-order interaction information is generally difficult, only pairwise interactions were monitored in [2]. Such a pairwise diversity, however, can only capture the low-order components of the multiple orders of correlation.

3 Multi-information Diversity

Lemma 1. *The multi-information and conditional multi-information can be expanded as a sum of mutual information and conditional mutual information terms, respectively. That is,*

$$\mathcal{I}(X_{1:n}) = \sum\nolimits_{i=2}^{n} I(X_i; X_{1:i-1}); \quad (8)$$

$$\mathcal{I}(X_{1:n} \mid Y) = \sum\nolimits_{i=2}^{n} I(X_i; X_{1:i-1} \mid Y). \quad (9)$$

Proof. The multi-information can be written as [4,13]

$$\begin{aligned} \mathcal{I}(X_{1:n}) &= \sum\nolimits_{i=1}^{n} H(X_i) - H(X_{1:n}) \\ &= \sum\nolimits_{i=1}^{n} H(X_i) - \left[H(X_1) + \sum\nolimits_{i=2}^{n} H(X_i \mid X_{1:i-1})\right] \\ &= \sum\nolimits_{i=2}^{n} \left(H(X_i) - H(X_i \mid X_{1:i-1})\right) = \sum\nolimits_{i=2}^{n} I(X_i; X_{1:i-1}) \ , \end{aligned}$$

which is the result in Eq. 8.

For conditional multi-information, its definition in Eq. 4 can be written as

$$\sum\nolimits_{y, x_{1:n}} p(x_{1:n}, y) \left[\log \frac{p(x_{1:n}, y)}{p(x_1) \cdots p(x_n) p(y)} - \log \frac{p(x_1, y) \cdots p(x_n, y)}{p(x_1) \cdots p(x_n) p(y)^n}\right].$$

Then, by Eqs. 1 and 3, we have

$$\begin{aligned}\mathcal{I}(X_{1:n} \mid Y) &= \mathcal{I}(X_{1:n}, Y) - \sum_{i=1}^{n} I(X_i; Y) \\ &= \sum_{i=2}^{n} [H(X_i, Y) + H(X_{1:i-1}, Y) - H(X_{1:i}, Y) - H(Y)] \\ &= \sum_{i=2}^{n} I(X_i; X_{1:i-1} \mid Y),\end{aligned}$$

which completes the proof. □

Theorem 1. *For a set of m classifiers $S = \{X_1, \cdots, X_m\}$ and the class label Y, the mutual information $I(X_{1:m}; Y)$ can be expanded as*

$$\begin{aligned}I(X_{1:m}; Y) &= \underbrace{\sum_{i=1}^{m} I(X_i; Y)}_{\text{relevance}} + \underbrace{\mathcal{I}(X_{1:m} \mid Y) - \mathcal{I}(X_{1:m})}_{\text{multi-information diversity}} \\ &= \sum_{i=1}^{m} I(X_i; Y) + \underbrace{\sum_{i=1}^{m} I(X_i; X_{1:i-1} \mid Y)}_{\text{conditional redundancy}} - \underbrace{\sum_{i=1}^{m} I(X_i; X_{1:i-1})}_{\text{redundancy}}.\end{aligned} \tag{10}$$

Proof. Based on the properties of mutual information, we have

$$\begin{aligned}I(X_{1:m}; Y) &= H(X_{1:m}) + H(Y) - H(X_{1:m}, Y) \\ &= \sum_{i=1}^{m} H(X_i) + H(Y) - H(X_{1:m}, Y) + H(X_{1:m}) - \sum_{i=1}^{m} H(X_i) \\ &= \mathcal{I}(X_{1:m}, Y) - \mathcal{I}(X_{1:M})\end{aligned}$$

By adding $\sum_{i=1}^{m} I(X_i; Y) - \sum_{i=1}^{m} I(X_i; Y)$ to the right-hand side of above equation, it follows that

$$\begin{aligned}I(X_{1:m}; Y) &= \sum_{i=1}^{m} I(X_i; Y) + \mathcal{I}(X_{1:m}, Y) - \sum_{i=1}^{m} I(X_i; Y) - \mathcal{I}(X_{1:m}) \\ &= \sum_{i=1}^{m} I(X_i; Y) + \sum_{i=1}^{m} I(X_i; X_{i-1}, \cdots, X_1 \mid Y) - \mathcal{I}(X_{1:m}).\end{aligned}$$

From Eq. 8 in Lemma 1, $\mathcal{I}(X_{1:m})$ can be written as $\sum_{i=1}^{m} I(X_i; X_{1:i-1})$. Therefore, above equation becomes Eq. 10, which completes the proof. □

Comparing with Eq. 7, the formulation of Eq. 10 is much simpler while the meanings are easier to understand; that is, the *redundancy* and *conditional redundancy* are multi-information and conditional multi-information, and the multi-information diversity is just their difference.

Both the *redundancy* and *conditional redundancy* in Eq. 10 are in the form of sum of $I(X_i; X_{1:i-1})$'s and $I(X_i; X_{1:i-1} \mid Y)$'s, respectively. One advantage of our formulation is that they are *decomposable* over component classifiers.

Take *redundancy* for example, given an ensemble of size k, its redundancy is $\mathcal{I}(X_{1:k}) = \sum_{i=1}^{k} I(X_i; X_{1:i-1})$. Then, if a new classifier X_{k+1} is added, the new redundancy becomes $\mathcal{I}(X_{1:k+1}) = \sum_{i=1}^{k+1} I(X_i; X_{1:i-1})$, and the only difference is the mutual information $I(X_{k+1}; X_{1:k})$. That is, during the ensemble construction process, each classifier X_i can be characterized by the following measurements:

- **Relevance:** $I(X_i;Y)$ which measures its relevance to the class label, and bounds its prediction error;
- **Redundancy:** $I(X_i;X_{1:i-1})$ which measures the dependency between current classifier and existing classifiers;
- **Conditional Redundancy:** $I(X_i;X_{1:i-1} \mid Y)$ which measures the conditional dependency between current classifier and existing classifiers given the class label;
- **Diversity:** Which is the difference between the conditional redundancy and redundancy, and measures its contribution to the ensemble diversity.

Next, we study the relationship between Eqs. 10 and 7; that is, the relation between multi-information diversity and Brown's result.

Lemma 2. *Given a set of variables* $V = \{X_1, \cdots, X_n\}$*, it always holds that*

$$\sum_{k=2}^{n} \sum_{T_k \subseteq V} I(\{T_k\}) = \mathcal{I}(X_{1:n}). \tag{11}$$

Proof. It has been shown in [1] that the *interaction information* can be expanded as a sum of entropies, i.e.,

$$I(\{X_{1:n}\}) = -\sum_{T \subseteq V} (-1)^{|V \setminus T|} H(T), \tag{12}$$

where $V = \{X_{1:n}\}$ and $\sum_{T \subseteq V}$ denotes a sum over all possible subsets of V. Let

$$\Gamma_k = \sum_{T_k \subseteq V} H(T_k) \quad \text{and} \quad \Theta_k = \sum_{T_k \subseteq V} I(\{T_k\}) \ .$$

Substituting Eq. 12 into Θ_k, we can obtain $\Theta_2 = C_{n-1}^1 \cdot \Gamma_1 - \Gamma_2$ and $\Theta_3 = -C_{n-1}^2 \cdot \Gamma_1 + C_{n-2}^1 \cdot \Gamma_2 - \Gamma_3$, and in more general case,

$$\Theta_k = (-1)^k \cdot C_{n-1}^{k-1} \cdot \Gamma_1 + (-1)^{k-1} \cdot C_{n-2}^{k-2} \cdot \Gamma_2 + \cdots + (-1) \cdot \Gamma_k \ . \tag{13}$$

Substituting Eq. 13 into $\sum_{k=2}^{n} \Theta_k$ which is the left-hand side of Eq. 11, it is easy to find that the coefficient of Γ_i is

$$-\sum_{j=1}^{n-i} C_{n-i}^{j} (-1)^j = -(1-1)^{n-i} = 0, \quad \text{for} \ \ i = 2, \cdots, n-1$$

and the coefficients of Γ_1, Γ_n are 1 and -1, respectively. Consequently, it follows that

$$\sum_{k=2}^{n} \Theta_k = \Gamma_1 - \Gamma_n = \sum_{i=1}^{n} H(X_i) - H(X_{1:n}) = \mathcal{I}(X_{1:n}) \ ,$$

which completes the proof. □

Corollary 1. *Eq. 7 and Eq. 10 are mathematically equivalent.*

Proof. It is obvious that the relevance terms in Eqs. 7 and 10 are the same. Based on Lemma 2, we have $\mathcal{I}(X_{1:m}) = \sum_{k=2}^{m} \sum_{T_k \subseteq S} I(\{T_k\})$. Since $I(X_{1:m}; Y)$ appears in the left-hand sides of both equations, it follows that $\mathcal{I}(X_{1:m} \mid Y) = \sum_{k=2}^{m} \sum_{T_k \subseteq S} I(\{T_k\}|Y)$. Consequently, we can reach the conclusion that two equations are equivalent. □

Hence, although our formulation is simpler, it is mathematically equivalent to Brown's formulation. Moreover, Brown's formulation decomposes the ensemble diversity over different orders of interaction, while our formulation decomposes over the component classifiers. Based on our formulation we have an approximation for estimation, which will be presented in the next section.

4 Approximate Estimation

For estimating the multi-information diversity and its related terms, one straightforward approach is to estimate the joint probability. However, the exponential number of possible variable values will make the estimation of joint distribution infeasible in practice. So we take an approximation.

For redundancy, our task is reduced to estimate $I(X_i; X_{1:i-1})$ for all i's. Here, rather than estimating the joint probabilities, we approximate $I(X_i; X_{1:i-1})$ by

$$I(X_i; X_{1:i-1}) \approx \max_{\Omega_k} I(X_i; \Omega_k) \ , \tag{14}$$

where $\Omega = \{X_{i-1}, \cdots, X_1\}$, and Ω_k is a subset of size k.

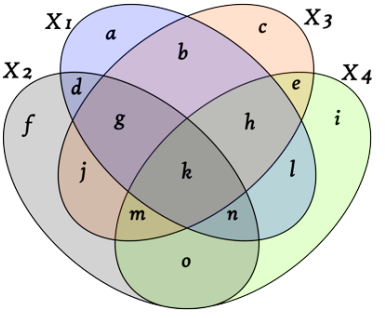
As an illustrative example, Fig. 1 depicts a Venn diagram for four variables, where the ellipses represent the entropies of different variables, while the mutual information can be represented by the combination of regions in the diagram. As shown in the right side of the figure, it can be found that the high-order component $I(X_4; X_3, X_2, X_1)$ shares a large intersection with the low-order component $I(X_4; X_2, X_1)$, where the only difference is the region e. Note that if X_1, X_2 and X_3 are strongly correlated, it is highly likely that the uncertainty of X_3 is covered by X_1 and X_2, that is, the regions c and e would be very small. Thus, $I(X_4; X_2, X_1)$ provides an approximation to $I(X_4; X_3, X_2, X_1)$. Such a scenario often happens in ensemble construction since the component classifiers generally have strong correlations.

With respect to the conditional redundancy and multi-information diversity, we take similar strategies as follows.

$$I(X_i; X_{1:i-1} \mid Y) \approx \max_{\Omega_k} I(X_i; \Omega_k \mid Y) \tag{15}$$

$$I(X_i; X_{1:i-1} \mid Y) - I(X_i; X_{1:i-1}) \approx \max_{\Omega_k} [I(X_i; \Omega_k \mid Y) - I(X_i; \Omega_k)] \tag{16}$$

To accomplish the approximation in Eqs. 14, 15 and 16, an enumeration over all the Ω_k's is desired. In this way, however, for every i we need estimate $I(X_i; \Omega_k)$ and $I(X_i; \Omega_k \mid Y)$ for C_{i-1}^{k} number of different Ω_k's. When k is near $(i-1)/2$, the number will be large, and the estimation of $I(X_i; \Omega_k)$ and $I(X_i; \Omega_k \mid Y)$ will



$$
\begin{aligned}
&I(X_4; X_3, X_2, X_1) = e + h + k + l + m + n + o,\\
&I(X_4; X_2, X_1) = h + k + l + m + n + o,\\
&I(X_4; X_3, X_1) = e + h + k + l + m + n,\\
&I(X_4; X_3, X_2) = e + h + k + m + n + o.
\end{aligned}
$$

Fig. 1. Venn diagram of an illustrative example

become difficult. Therefore, we need to take a trade-off. In our experiments, we restrict k to be one or two.

If considering only pairwise interactions, our method (MTI) and Brown's method [2] estimate $\sum_{i=2}^{n} I(X_i; X_{1:i-1})$ by

$$\sum\nolimits_{i=2}^{n} \max_{k<i} I(X_i; X_k) \quad \text{and} \quad \sum\nolimits_{i=2}^{n} \sum\nolimits_{j=1}^{i-1} I(X_i; X_j) \ ,$$

respectively. In other words, $I(X_i; X_{1:i-1})$ is approximated by $\max_{k<i} I(X_i; X_k)$ and $\sum_{j=1}^{i-1} I(X_i; X_j)$, respectively, by our method and Brown's method.

Take $I(X_4; X_3, X_2, X_1)$ in Fig. 1 for example. It is easy to get that,

$$
\begin{aligned}
&I(X_i; X_{1:i-1}) - \max_{k<i} I(X_i; X_k) = I(X_4; X_{1:3}) - \max_{k<4} I(X_4; X_k)\\
&= (e + h + k + l + m + n + o) - \max\{(h + k + l + n), (k + m + n + o),\\
&(e + h + k + m)\} = \min\{(e + m + o), (e + h + l), (l + n + o)\} \ , \qquad (17)
\end{aligned}
$$

$$
\begin{aligned}
&\sum\nolimits_{j=1}^{i-1} I(X_i; X_j) - I(X_i; X_{1:i-1}) = \sum\nolimits_{j=1}^{3} I(X_4; X_j) - I(X_4; X_{1:3})\\
&= (h + k + l + n) + (k + m + n + o) + (e + h + k + m) - (e + h + k +\\
&l + m + n + o) = 2k + h + m + n \ . \qquad (18)
\end{aligned}
$$

Note that the right-hand sides of both Eqs. 17 and 18 are nonnegative, which implies that our approximation is a lower bound never larger than the true value, while Brown's estimation is an upper bound never smaller than the true value. Moreover, it is easy to get that, at least when the following equation holds,

$$3k + h + m + n > e + l + o \qquad (19)$$

the right-hand side of Eq. 18 is larger than that of Eq. 17, which means that our approximation is closer to the true value than Brown's estimation. It is worth noting in Eq. 19 that the region k represents the uncertainty shared by all the four variables, the regions h, m and n represent those shared by three variables,

while the regions e, l and o represent those shared by two variables. This discloses that when the variables are highly correlated (such as when Eq. 19 holds), our approximation is expected to be closer to the ground truth. In ensemble construction, the component classifiers are generally highly correlated, so we expect that our approach is better for estimating ensemble diversity and related terms in practice. This will be empirically verified in the next section.

5 Experiments

5.1 Synthetic Data

To evaluate our proposed approach, experiments on synthetic data is performed at first since we can get the ground-truth information.

The task is to estimate $\mathcal{I}(X_{1:n})$. The synthetic data is generated as follows. Assume a and b are integers sampled from the interval $[1, 100]$, the variables

$$X_1 \equiv a \pmod 2, \quad X_2 \equiv a + b \pmod 2, \quad X_3 \equiv a - b \pmod 2,$$
$$X_4 \equiv a \times b \pmod 2, \quad X_5 \equiv a^2 + b^2 \pmod 2$$

are correlated. It is easy to get $p(X_i)$'s, $p(X_i, X_j)$'s, $p(X_i, X_j, X_k)$'s and $p(X_{1:5})$, based on which the ground-truth $\mathcal{I}(X_{1:5})$ can be obtained analytically.

We evaluate our method (MTI), where k is restricted to be 1 and 2, denoted by MTI_1 and MTI_2, respectively. We also evaluate Brown's method [2]. Since MTI_1 and MTI_2 consider the pairwise and three-order interactions, for a fair comparison, in addition to the method reported in [2] which considers only pairwise interactions, we extended Brown's method to consider three-order interactions by using Eq. 12 to help estimate the three-order interaction information. These two versions are denoted by Brown_1 and Brown_2, respectively. The estimated as well as the ground-truth multi-information are shown in Table 1, where the *relative error* is the difference between the ground-truth value and the estimated value divided by the ground-truth value.

From Table 1 we can find that Brown_2 and MTI_2 perform much better than Brown_1 and MTI_1, and MTI_2 reaches the ground-truth value. This is easy to understand since Brown_2 and MTI_2 explore the three-order interactions while Brown_1 and MTI_1 consider only the pairwise interactions. Comparing MTI_1 (MTI_2) with Brown_1 (Brown_2), it can be found that the estimation of MTI is more closer to the ground-truth, although they consider the same pairwise (or three-order) interactions. This verifies our argument that MTI is more accurate if the variables are highly correlated.

Table 1. Evaluation on synthetic data

	Ground-truth	**Brown_1**	**Brown_2**	**MTI_1**	**MTI_2**
$\mathcal{I}(X_{1:5})$	1.9485	2.9418	1.9946	1.6019	1.9485
Relative error	—	50.98%	2.37%	-17.79%	0.00%

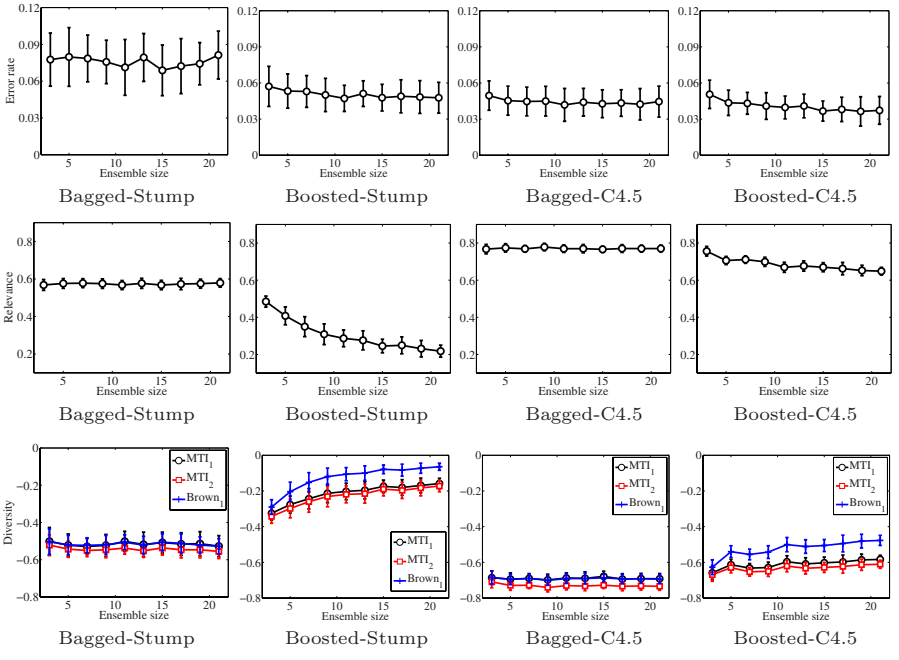


Fig. 2. *Breast Cancer* (1st row: Error rate; 2nd row: Relevance; 3rd row: Diversity)

5.2 Real Data

Next we apply our MTI approach and Brown's method to study the behavior of AdaBoost and Bagging on real data.

In the experiments we have used two types of base classifiers, i.e., *decision stump* and *C4.5 decision tree*. The ensemble sizes are set to 3 to 21. For each configuration, we execute 50 runs of hold-out tests on UCI data set *Breast Cancer* and *Heart Disease*. In each run, two thirds data are used for training while the remaining for testing. The relevance, multi-information diversity, redundancy and conditional redundancy terms are estimated on the training data, and used to explain the ensemble prediction error on test data. Here, MTI_1, MTI_2 and Brown_1 are evaluated.[2] Note that the estimated relevance, redundancy and conditional redundancy are monotonically increasing as the ensemble size increases.

[2] In the experiments on synthetic data, the task is to estimate the interaction information and we can use Eq. 12 to help estimate the three-order interaction information for Brown_2. On the real data, for estimating the diversity, Brown_2 requires to calculate three-order conditional interaction information, yet the calculation is not straightforward. Moreover, Brown_1 averages all the C_n^2 number of pairwise redundancy (and conditional redundancy) terms, while Brown_2 needs to deal with C_n^2 pairwise terms and C_n^3 three-order terms (in our approach there are always only n terms no matter what order is considered), and it is not clear whether the terms of different orders should be averaged together or not. So, in the experiments on real data we have not evaluated Brown_2.

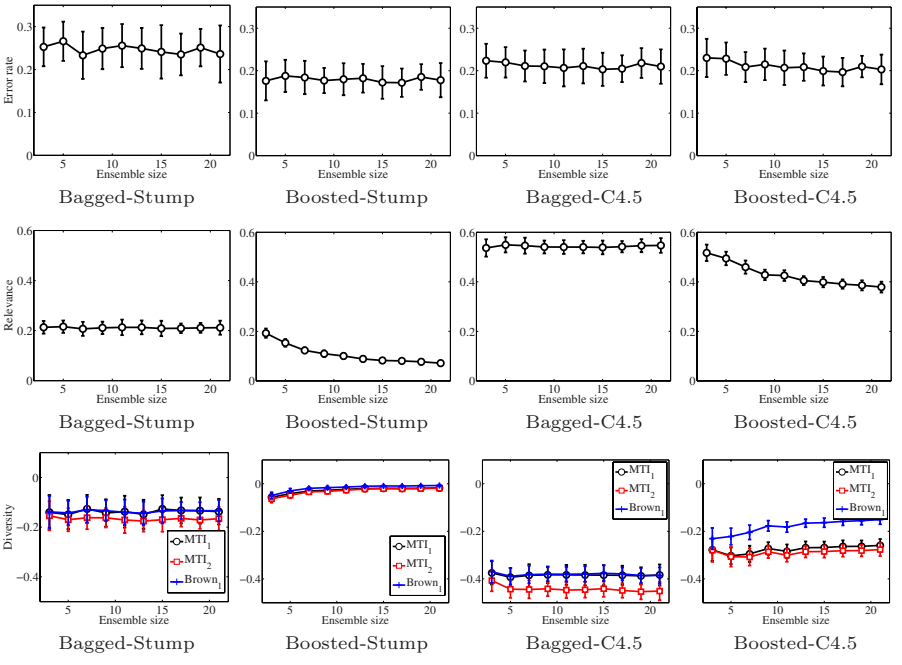


Fig. 3. *Heart Disease* (1st row: Error rate; 2nd row: Relevance; 3rd row: Diversity)

In the following, the mean of $\max_{\Omega_k} I(X_i; \Omega_k)$'s and $I(X_i; X_j)$'s are reported for MTI and Brown_1, respectively. The estimated values are averaged over 50 runs, and the mean values and standard deviations are reported. The results are shown in Figs. 2 and 3, respectively.[3]

We can see that Adaboost decreases the relevancy of its classifiers while increasing the diversity. Bagging has a very different behavior. It maintains almost constant relevancy and diversity. This is accordance with the sequential and parallel generation style of the component classifiers in AdaBoost and Bagging. With respect to ensembles with different base classifiers, it is easy to find that ensembles with decision stumps have lower relevance but higher diversity, while ensembles with C4.5 decision trees have higher relevance but lower diversity. This is easy to understand because decision stumps have lower accuracy and is more sensitive to data samples than C4.5 decision trees.

Comparing MTI_1, MTI_2 and Brown_1, it can be found that in general, all the three methods make similar diversity estimations except that Brown_1 is a little bit divergence on AdaBoost; this suggests that all these methods are useful for measuring ensemble diversity. Moreover, it can be found that the estimated values of Brown_1 is never smaller than that of MTI_1; this is accordance with our argument that MTI estimation lower bounds while Brown's estimation upper bounds the true value.

[3] The redundancy and conditional redundancy are not shown due to space limitation.

6 Conclusion

In this paper, we propose a formulation of ensemble diversity from the view of *multi-information*. This formulation is mathematically equivalent to the previous information theoretic diversity formulation [2], but is simpler and decomposable over component classifiers. Based on the formulation, we present an approximation for estimation. Experimental results show that the approximation can be used to study the behavior of ensemble methods to some extent in practice.

Acknowledgement. The authors want to thank anonymous reviewers for their helpful suggestions. This work was supported by NSFC (60635030, 60721002), JiangsuSF (BK2008018), 973 Program (2010CB327903) and the Startup Foundation in School of Mathematical Sciences at Soochow University.

References

1. Bell, A.J.: The co-information lattice. In: Proc. 4th Intl. Symp. Independent Component Analysis & Blind Signal Separation, pp. 921–926 (2003)
2. Brown, G.: An information theoretic perspective on multiple classifier systems. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 344–353. Springer, Heidelberg (2009)
3. Brown, G.: A new perspective on information theoretic feature selection. In: Proc. 12th Intl. Conf. Artificial Intelligence and Statistics, pp. 49–56 (2009)
4. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, Chichester (1991)
5. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine learning 40, 139–157 (2000)
6. Fano, R.: Transmission of Information: Statistical Theory of Communications. Wiley, Chichester (1961)
7. Giacinto, G., Roli, F.: Design of effective neural network ensembles for image classification purposes. Image and Vision Computing 19, 699–707 (2001)
8. Hellman, M., Raviv, J.: Probability of error, equivocation, and the chernoff bound. IEEE Trans. Information Theory 16, 368–372 (1970)
9. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20, 832–844 (1998)
10. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51, 181–207 (2003)
11. Kuncheva, L.I., Whitaker, C.J., Shipp, C., Duin, R.: Limits on the majority vote accuracy in classifier fusion. Pattern Analysis and Applications 6, 22–31 (2003)
12. McGill, W.: Multivariate information transmission. IEEE Trans. Information Theory 4, 93–111 (1954)
13. Slonim, N., Friedman, N., Tishby, N.: Multivariate information bottleneck. Neural Computation 18, 1739–1789 (2006)
14. Studeny, M., Vejnarova, J.: The multi-information function as a tool for measuring stochastic dependence. In: Jordan, M.I. (ed.) Learning in Graphical Models, pp. 261–298. Kluwer, Dordrecht (1998)
15. Watanabe, S.: Information theoretical analysis of multivariate correlation. IBM Journal of Research and Development 4, 66–82 (1960)

Dynamic Selection of Ensembles of Classifiers Using Contextual Information

Paulo R. Cavalin[1], Robert Sabourin[1], and Ching Y. Suen[2]

[1] École de Technologie Supérieure, 1100 Notre-dame ouest, Montreal(QC), Canada, H3C-1K3

[2] CENPARMI, Concordia University, 1455 de Maisonneuve Blvd West, Montreal(QC), Canada, H3G-1M8

Abstract. In a multiple classifier system, dynamic selection (DS) has been used successfully to choose only the best subset of classifiers to recognize the test samples. Dos Santos et al's approach (DSA) looks very promising in performing DS, since it presents a general solution for a wide range of classifiers. Aiming to improve the performance of DSA, we propose a context-based framework that exploits the internal sources of knowledge embedded in this method. Named DSA^c, the proposed approach takes advantage of the evidences provided by the base classifiers to define the best set of ensembles of classifiers to recognize each test samples, by means of contextual information provided by the validation set. In addition, we propose a switch mechanism to deal with tie-breaking and low-margin decisions. Experiments on two handwriting recognition problems have demonstrated that the proposed approach generally presents better results than DSA, showing the effectiveness of the proposed enhancements. In addition, we demonstrate that the proposed method can be used, without changing the parameters of the base classifiers, in an incremental learning (IL) scenario, suggesting that it is also a promising general IL approach. And the use of a filtering method shows that we can significantly reduce the complexity of DSA^c in the same IL scenario and even resulting in an increase in the final performance.

1 Introduction

Dynamic selection (DS) of classifiers is a very interesting domain for multiple classifier systems (MCS). DS consists of selecting only the best members from the pool of classifiers, denoted as $C = \{c_1, c_2, \ldots, c_N\}$, to recognize the test sample $x_{i,test}$. As a result, the best classification scheme is defined for each sample, so that lower error rates are expected. Note that when more than one classifier is selected, we can refer to this method as dynamic selection of ensembles of classifiers (DSEoC).

A promising approach for DSEoC is Dos Santos et al's approach (DSA) [1], which is able to dynamically select ensembles of classifiers (EoCs) by using only crisp label outputs, i.e. class votes, provided by the base classifiers. DSA is a general DSEoC approach, since any type of classifier that outputs votes can be

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 145–154, 2010.

used as a base classifier. A general approach is very desirable since it can be easily adapted to different base classifiers, and it can be used with combinations of different types of classifiers. This allows us, for example, to combine decisions of neural networks and hidden Markov models, or a classifier with the decision of a human expert.

Nonetheless, the structure of DSA is very rich, and many internal sources of knowledge have not been fully exploited yet. For example, we could improve the way we take advantage of the diversity presented by the members of the pool of EoCs. Also, we could improve the way these EoCs are selected. For example, we could select more than a single EoC to recognize $x_{i,test}$. By improving the way DSA uses these sources of knowledge, we believe that we can reach higher recognition rates, resulting in an approach that is both general and robust. Consequently, we propose a method that tries to use this knowledge in a better way.

The proposed method, to which we refer as DSA^c, includes the evidences produced by the base classifiers to take advantage of a labeled dataset (e.g. a source of contextual information) to indicate which is the best set of EoCs for each test sample. In this case, it is not a single EoC that is selected to recognize the test sample, but the best set that can comprise one or more EoCs. Such a selection uses output profiles, which are represented by the outputs of the base classifiers, to find the samples in the validation set that are those most similar to the test sample. Afterwards, we compute the best set of EoCs to recognize $x_{i,test}$ based on the evidences provided by those most similar validation samples. Furthermore, we also add a switch mechanism to DSA^c, aiming at choosing the best source of knowledge to compute the final decision whenever the answer provided by the dynamically selected EoC is not considered convincing enough. Consequently, the switch works as a tie-breaking approach, which reduces the chances of random decisions by using existing knowledge.

The remainder of this paper is organized as follows. In Section 2, we provide a brief description of DSA. In Section 3, we describe the proposed approach named DSA^c. Next, in Section 4, we report and discuss the results of an experimental evaluation performed on two handwriting recognition problems. Finally, in Section 5, we present conclusions and point out future work.

2 Dos Santos et al's Approach (DSA)

The overall architecture of DSA is depicted in Figure 1. The main objective of this method is to dynamically find the best EoC, whose members are a subset of $C = \{c_1, c_2, \ldots, c_N\}$, to recognize the test sample $x_{i,test}$. This task is performed by considering only the recognition outputs $O_i = \{o_{i,1}, \ldots, o_{i,N}\}$ computed from C. Each output corresponds to a class from the set $\Omega = \{\omega_1, \ldots, \omega_M\}$.

DSA is divided into two phases: the design phase and the operational phase. During the design phase, the architecture that supports the dynamic selection of EoCs is created. In other words, the pool of EoCs $C^{*\prime} = \{C'_1, \ldots, C'_W\}$, where $C'_j \subset C, 1 \leq j \leq W$, is created during this phase and is a subset of all possible

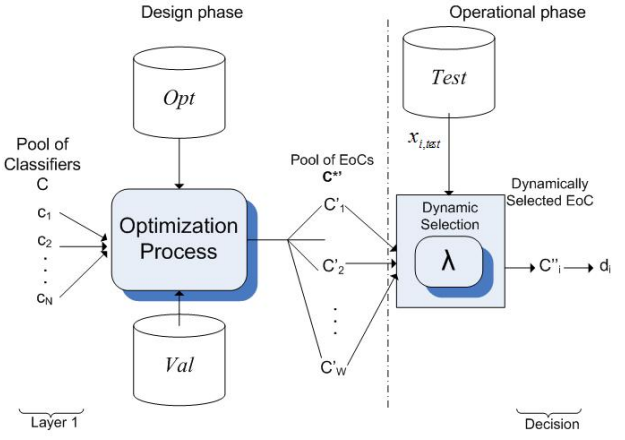


Fig. 1. Dos Santos et al's approach (DSA). The pool of classifiers is organized into a pool of EoCs during the design phase. During the operational phase, the EoC, which is dynamically selected by λ, produces the final decision.

EoCs C^*. The pool $C^{*\prime}$ is generated by a search algorithm, which is a genetic algorithm (GA) in this work. Each individual is represented by a binary vector of N positions, where each bit represents whether or not a classifier is selected to be a member of an EoC. The fitness function, which has to be minimized, uses the error rate on the optimization set Opt, by applying the majority voting method on the EoCs assigned by each individual. In order to avoid overfitting, each individual is also evaluated on the validation set Val, and the best solutions are saved into an archive whose size is W. The archive is then used as $C^{*\prime}$.

The operational phase is composed of the modules that conduct the dynamic selection of the best EoC C''_i, which includes one of the EoCs in $C^{*\prime}$, to recognize $x_{i,test}$. This task is undertaken by using a dynamic selection function (DSF), to which we refer as λ. Afterwards, we compute the votes provided by all members in C''_i, and the class with the highest number of votes represents the final decision d_i.

The DSF λ is related to one DSF, as described in [1], such as Ambiguity-guided dynamic selection (ADS), Margin-based dynamic selection (MDS), and Class-strength dynamic selection (CSDS). In this work, we simply assign DSA to ADS for the sake of simplicity.

3 DSA^c: Enhancing Dynamic Selection by Using Contextual Information and a Switch Mechanism

In DSA, EoCs are dynamically selected by considering DSFs based on the extent of consensus. Despite that the extent of consensus is a well studied concept in literature [2], only the outputs of the most voted and the second most voted

classes are used to select the ensemble. However, the information related to the other classes is wasted, even though such information could help this task. In addition, only one EoC is dynamically selected at a time. Finally, in many cases, the final recognition might not be confident enough, and random guesses are associated to the final result. In order to overcome these drawbacks, we propose DSA^c, depicted in Figure 2.

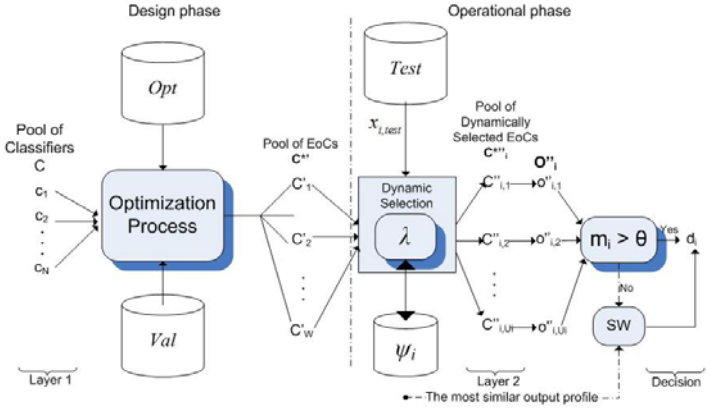


Fig. 2. An overview of the DSA^c approach. This method uses the knowledge provided by Val (converted into the set of output profiles Val').

The main objective of DSA^c is to use the validation database, transformed into output profiles, to point out which EoCs are the most competent to recognize the test sample $x_{i,test}$. An output profile is computed by the transformation T in Equation 1, where $x_i \in \Re^D$, $\tilde{x}_i \in Z^{N+}$, and N is the size of the pool of base classifiers C. Given that we know which EoC correctly recognizes each validation sample, the dynamically selected set of EoCs, denoted by $C^{*\prime\prime}_i = \{C''_{i,1}, \ldots, C''_{i_U}\}$, is composed by the EoCs that correctly classify the validation samples that are the ones most similar to the test samples in considering the output profiles.

$$T : x_i \Rightarrow \tilde{x}_i, \tag{1}$$

In greater detail, this approach works as follows. Consider the pool of EoCs $C^{*\prime}$, generated during the design phase. For each test sample $x_{i,test}$, we compute the best set of EoCs $C^{*\prime\prime}{}_i$, composed of members from $C^{*\prime}$. Each EoCs from $C^{*\prime}$ may appear several times in $C^{*\prime\prime}{}_i$, resulting in an automatic weighting approach. This task is achieved by considering the DSF λ, which is depicted in Figure 3.

The DSF λ works as follows. First, we apply T on $x_{i,test}$, resulting in $\tilde{x}_{i,test}$. Next, we compare $\tilde{x}_{i,test}$ to each output profile in Val', which is a database containing the output profiles of all validation samples in Val, e.g. $\tilde{x}_{j,val}\ \forall x_{j,val} \in Val$,

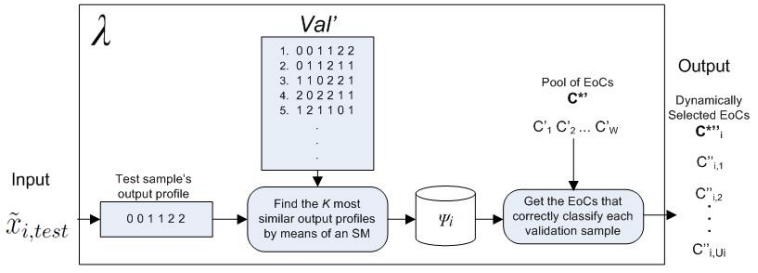


Fig. 3. The DSF λ. For each test sample, we find K validation samples with the most similar output profiles, to form the set ψ_i. The EoCs that correctly classify the validation samples in Ψ_i are used to compose the set $C^{*\prime\prime}$, which is then used to compute the final decision of DSA^c.

computed during the design phase. We compare these samples in terms of similarity, and store the degree of similarity between $\tilde{x}_{i,test}$ and $\tilde{x}_{j,val}$ in $\delta_{i,j}$. Note that we use one of the similarity measures described in Section 3.1 to compute $\delta_{i,j}$. The K most similar output profiles $\tilde{x}_{j,val}$, e.g. the validation samples related to the highest values of $\delta_{i,j}$, are stored in Ψ_i. Next, the EoCs from $C^{*\prime}$ which correctly recognize each sample in Ψ_i are computed. These EoCs are then included in ${C^{*\prime\prime}}_i$. As mentioned, an EoC appears in ${C^{*\prime\prime}}_i$ as many times as the number of samples that it correctly recognizes. Finally, ${C^{*\prime\prime}}_i$ is submitted to the remaining modules of DSA^c.

The last step consists of submitting the outputs of ${C^{*\prime\prime}}_i$ to the switch mechanism, represented by the SW module in Figure 2. Here we employ the concept of margin [2] to identify whether or not the answers provided by $C^{*\prime\prime}_i$ are confident enough, using a threshold θ. In considering the margin m_i, for the test sample $x_{i,test}$, if $m_i > \theta$, then we use the most voted class indicated by ${C^{*\prime\prime}}_i$. Otherwise, we use the label of the most similar validation sample from Ψ_i. The main goal of this scheme is to use contextual information also in the switch mechanism. Note that, hereafter, the margin is represented by the difference between the number of votes of the most voted class and the second most voted one.

In the next section we present the similarity measures used to compute $\delta_{i,j}$.

3.1 Similarity Measures (SMs)

Hereafter, we use the following additional notations: $\tilde{x}_{i,test,k}$ represents the output of classifier k for $x_{i,test}$, and $\tilde{x}_{j,val,k}$ represents the same for $x_{j,val}$. In addition, for each $x_{j,val}$, the set of flags $CC_j = \{cc_{j,1}, cc_{j,2}, \ldots, cc_{j,W}\}$, where each $cc_{j,k}$ is a binary value, represents whether C'_k has correctly classified $x_{j,val}$ or not. In other words, $cc_{j,k} = 1$ if C'_k correctly classifies $x_{j,val}$, otherwise, $cc_{j,k} = 0$.

In this work we consider three different SMs. Note that they are individually used by λ. The three SMs are described below.

Euclidean Distance (ED). The Euclidean distance between the output profile of $\tilde{x}_{i,test}$ and each $\tilde{x}_{j,val}$ $\forall j$, represented by the following equation:

$$ED_{i,j} = \sum_{k=1}^{N} |\tilde{x}_{i,test,k} - \tilde{x}_{j,val,k}| \tag{2}$$

Template Matching (TM). The computation of the number of classifiers that provide the same output. This SM is computed by maximizing Equation 3.

$$TM_{i,j} = \frac{\sum_{k=1}^{N} \alpha_{i,j,k}}{N} \tag{3}$$

$$\alpha_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Oracle-Based Template Matching (OTM). In considering that each $\tilde{x}_{j,val}$ is related to the correct class label $correct_{j,val}$, we compute the number of classifiers that produce the correct class label for $\tilde{x}_{j,val}$ and provide the same output as $\tilde{x}_{i,test}$. Equation 5, which has to be maximized, computes this SM mathematically.

$$OTM_{i,j} = \frac{\sum_{k=1}^{N} \beta_{j,i,k}}{\sum_{k=1}^{N} \gamma_{j,k}} \tag{5}$$

$$\beta_{i,j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{i,test,k} = \tilde{x}_{j,val,k} \text{ and } \tilde{x}_{j,val,k} = correct_{j,val} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$\gamma_{j,k} = \begin{cases} 1, & \text{if } \tilde{x}_{j,val,k} = correct_{j,val} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

These SMs result in three different versions of DSA^c:

1. DSA$^c_{ED}$, where $\delta_{i,j} = 1 - \text{ED}_{i,j}$;
2. DSA$^c_{TM}$, where $\delta_{i,j} = \text{TM}_{i,j}$;
3. DSA$^c_{OTM}$, where $\delta_{i,j} = \text{OTM}_{i,j}$.

In the next section, we present an experimental evaluation of these methods.

4 Experiments

In this section we present a series of experiments whose main goals are: 1) to evaluate whether dynamic methods are better than static ones; 2) to evaluate if DSA^c results in lower error rates than DSA;

The aforementioned evaluation is supported by considering these methods: the original classifier with full representation space (all original features); the best base classifier from C; the fusion of all base classifiers in C by using MV; fusion

of all base classifiers in C by using decision templates (DT) [3], by considering the three proposed SMs; the best EoC from $C^{*\prime}$.

All methods are evaluated on two handwriting recognition problems: digits and uppercase letters, extracted from the NIST-SD19 database. For both problems, the original feature set is composed of 132 features, extracted from concavities and contours [4]. In addition, two different test sets for digits are used for evaluating digit recognition: *NIST-digits-test1* and *NIST-digits-test2*. Table 1 presents a detailed description of each database.

Table 1. Experimental setup. (NC: number of classes; NF: number of features; NFE: number of features in the subspace, after applying the RSS method; VM: validation method;)

Problem	**NC**	**Set** $Train$	**Set** Opt	**Set** Val	**Set** $Test$	**NF**	**NFE**
Digits	10	5,000	10,000	10,000	*t1* 60,089	132	32
					t2 58,646		
Letters	26	43,160	3,980	7,960	12,092	132	32

For each dataset, 100 base classifiers, with 32 features, are generated from the original 132 features based on the random subspaces (RSS) ensemble generation method [5] for the number of features used for each problem). Two different base classifiers are considered: k-nearest neighbors classifiers with $k = 1$ (1NN), and C4.5 decision tree (DTree) classifiers.

To generate the pool of EoCs, GA is used to find an archive with the 25 best solutions on Val, representing $C^{*\prime}$, guided by the optimization set Opt. The following parameters were used in this work: population size: 128; maximum number of generations: 1,000; probability of crossover: 0.8; probability of mutation: 0.01; one-point crossover and bit-flip mutation [1]. The experiments are replicated 30 times, where in each replication the archive provided by GA is generally different. The results represent the mean error rates over the 30 replications.

To validate the results statistically, we use the Kruskal-Wallis nonparametric statistical test. We test the equality among the mean values, using a confidence level of 95%. Dunn-Sidak correction is applied to critical values.

4.1 Results and Discussion

The results in error rates are presented in Table 2. For both digits and letters, DSA^c with $K = 30$. Also, the parameter θ was set to zero after preliminary evaluations.

These experiments show that the proposed approach DSA^c has successfully presented lower error rates than DSA in all problems. This proves that both the

Table 2. Error rates using 1NN and DTree classifiers, at zero-level rejection. Results in bold present the best approach among static MO, DSA, DT, and the proposed DSA^c with K set to 30. Underlined results represent the statistically-significant best method. Highlighted by * are the proposed approaches. Between parentheses is the variance of each approach ($\times 10^{-2}$).

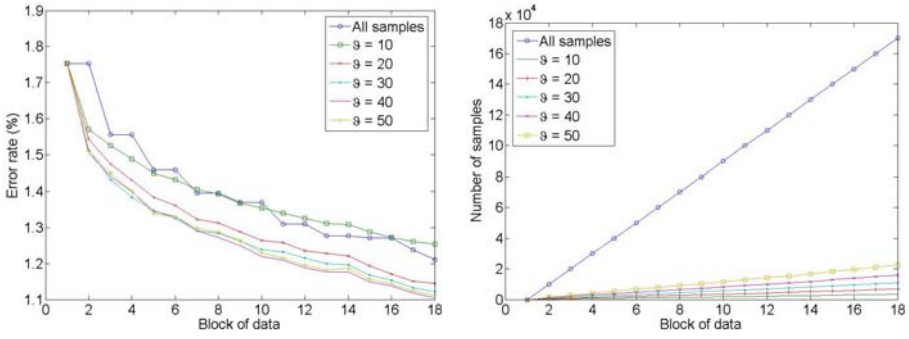
Classifier	**1NN**			**DTree**		
Method	**Digits**		**Letters**	**Digits**		**Letters**
	test1	*test2*		*test1*	*test2*	
Static selection						
Oracle C	0.05	0.17	0.18	0.01	0.04	0.04
All features	6.66	9.76	7.82	11.07	18.20	13.50
Best from C	7.52	13.99	14.47	10.30	19.18	17.13
MV all C	3.72	8.10	6.60	2.92	6.67	6.06
Best from $C^{*\prime}$	3.60 (3.83)	7.77 (7.74)	6.56 (6.73)	2.98 (4.98)	6.77 (1.12)	6.21 (7.82)
DT$_{TM}$ C	2.55	5.74	4.95	2.00	5.00	4.64
DT$_{OTM}$ C	4.74	9.74	7.56	2.70	6.03	7.15
DT$_{ED}$ C	2.97	6.57	6.55	2.56	6.26	7.44
Dynamic selection						
Oracle $C^{*\prime}$	1.97 (0.02)	4.59 (1.14)	3.87 (4.42)	1.87 (1.03)	4.39 (4.36)	4.53 (2.49)
DSA	3.61 (0.08)	7.87 (0.17)	6.43 (0.48)	2.87 (0.06)	6.61 (0.29)	6.06 (0.41)
$*$DSA$^c_{TM}$	**2.37** (0.02)	**5.34** (0.04)	4.62 (0.16)	**1.76** (0.02)	**4.36** (0.04)	4.20 (0.05)
$*$DSA$^c_{OTM}$	2.63 (0.03)	5.88 (0.17)	**4.10** (0.25)	2.16 (0.03)	4.96 (0.08)	**3.89** (0.06)
$*$DSA$^c_{ED}$	2.43 (0.03)	5.43 (0.16)	4.39 (0.28)	1.83 (0.05)	4.64 (0.10)	4.32 (0.09)

use of contextual information to select multiple EoCs, as well as the switch mechanism, were able to better use the sources of knowledge embedded in DSA. Consequently, these results demonstrate that dynamic selection outperforms static selection.

One interesting result from letters with DTree, shows the effectiveness of the switch method to decrease the dependency on the pool of EoCs. In that problem, the final error rates are lower than the oracle of the pool of EoCs. It would be impossible to reach this result without this mechanism.

Evaluation of DSA^c in an incremental learning scenario. One by-product of the proposed DSA^c approach, is the ability to adapt the system to knowledge acquired incrementally by simply adding more data to Val. As a result, we can incrementally learn new data with no need to update the parameters of the base classifiers. For this reason, we evaluate the impact of an increase in the size of Val for DTree classifiers on *NIST-digits-test1*.

We simulated an incremental scenario by incrementally increasing the number of samples in Val. We take advantage of the large set of digits available in the NIST SD19 database, by increasing the size of Val from 10,000 to 180,000 samples. Those are the remaining samples in the hsf_{1-3} series of the database. In addition, we also evaluate a control mechanism to select only samples that present the margin below a threshold ϑ, in considering the margin of the base classifiers, e.g. $m_i < \vartheta$, where m_i represents the difference of votes between the two most voted classes. The main idea is to hold only samples that present uncommon output profiles to reduce the size of Val. As a consequence, the final complexity of DSA^c is also reduced.



(a) Performance (b) Complexity

Fig. 4. Incremental evaluation of DSA^c ($K = 30$) with DTree classifiers on *NIST-digits-test1*

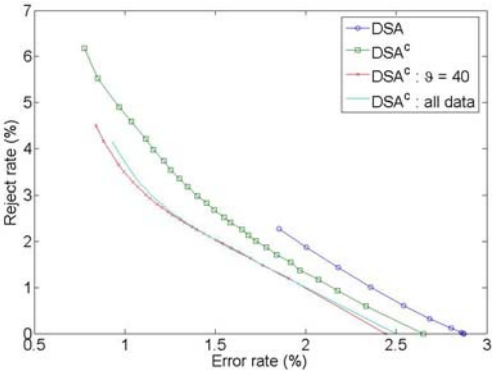


Fig. 5. Impact of a rejection mechanism on *NIST-digits-test1* with DTree classifiers

Figure 4(a) presents the results of these experiments. As shown, the incremental increase of Val is effective in reducing the error rates. By using all samples, the final error rates are reduced from 1.75% to about 1.2%. Furthermore, the proposed control mechanism is also effective in reducing the error rates. With $\vartheta = 40$, the final error rates are reduced to about 1.1%. In addition, we demonstrate in Figure 4(b) the effects of using the margin-based control mechanism. The best approach, represented by $\vartheta = 40$, used only 25,948 samples for training. Comparing with the use of all 180,000 training samples, we can reach better results by using only around 15% of this set and drastically reduce the complexity of DSA^c.

In addition, we also compare the performances of DSA and DSA^c by considering a rejection mechanism on *NIST-digits-test1* with DTree classifiers. The reject uses the margin of the dynamically selected EoC for DSA, and the margin of the dynamically selected set of EoCs for DSA^c. As depicted in Figure 5, DSA^c rejected less samples than DSA to reach the same error rates (for example, with

2.0 % of error, DSA rejects about 2% of samples, and DSA^c rejects only 1.5%.). When more samples are used for training, the DSA^c rejects even less samples. And, the performance of the approach using $\vartheta = 40$ was better than the one that uses all samples, even though the former was trained with much fewer samples.

5 Conclusion and Future Work

In this paper we proposed a novel methodology to improve a state-of-the-art approach for DSEoC, e.g. Dos Santos et al's approach (DSA). The proposed approach, referred to as DSA^c, uses the knowledge provided by output profiles to dynamically select EoCs. Furthermore, a switch mechanism was included to reduce the dependency on the pool of EoCs.

Experiments conducted on two handwriting recognition problems have confirmed that dynamic selection is really promising in improving the use of multiple classifiers. In addition, DSA^c has been effective to improve DSA. Also, the simulation of an incremental learning scenario showed us that we can improve the performance of DSA^c by increasing the size of the validation set only, without changing the parameters of the base classifiers.

As future work, we can focus on reducing the overall complexity DSA^c. We can, for example, use some proposed ideas to reduce the complexity of 1NN classifiers which work in a similar way. In addition, we must evaluate other strategies towards reducing the error rates. This can be achieved using other SMs, by filtering examples from Val, and so forth.

Acknowledgments

The authors would like to acknowledge the CAPES-Brazil and NSERC-Canada for the financial support.

References

1. Dos Santos, E.M., Sabourin, R., Maupin, P.: A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. Pattern Recognition 41, 2993–3009 (2008)
2. Hansen, L.K., Liisberg, C., Salamon, P.: The error-reject tradeoff. Open Systems & Information Dynamics 4(2), 159–184 (1997)
3. Kuncheva, L.I., Bezder, J.C., Duin, R.P.W.: Decision templates for multiple classifier fusion: an experimental comparison. Pattern Recognition 34, 299–314 (2001)
4. Oliveira, L.E.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Automatic recognition of handwritten numeral strings: A recognition and verification strategy. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(11), 1438–1454 (2002)
5. Ho, T.: The random subspace method for construction decision forests. IEEE Trans. on Pattern Analysis and Machine Intelligence 20, 832–844 (1998)

Selecting Structural Base Classifiers for Graph-Based Multiple Classifier Systems*

Wan-Jui Lee[1], Robert P.W. Duin[1], and Horst Bunke[2]

[1] Pattern Recognition Laboratory,
Delft University of Technology, The Netherlands
W.J.Lee@tudelft.nl, r.duin@ieee.org
[2] Institute of Computer Science and Applied Mathematics,
University of Bern, Switzerland
bunke@iam.unibe.ch

Abstract. Selecting a set of good and diverse base classifiers is essential for building multiple classifier systems. However, almost all commonly used procedures for selecting such base classifiers cannot be directly applied to select structural base classifiers. The main reason is that structural data cannot be represented in a vector space.

For graph-based multiple classifier systems, only using subgraphs for building structural base classifiers has been considered so far. However, in theory, a full graph preserves more information than its subgraphs. Therefore, in this work, we propose a different procedure which can transform a labelled graph into a new set of unlabelled graphs and preserve all the linkages at the same time. By embedding the label information into edges, we can further ignore the labels. By assigning weights to the edges according to the labels of their linked nodes, the strengths of the connections are altered, but the topology of the graph as a whole is preserved.

Since it is very difficult to embed graphs into a vector space, graphs are usually classified based on pairwise graph distances. We adopt the dissimilarity representation and build the structural base classifiers based on labels in the dissimilarity space. By combining these structural base classifiers, we can solve the labelled graph classification problem with a multiple classifier system. The performance of using the subgraphs and full graphs to build multiple classifier systems is compared in a number of experiments.

1 Introduction

A multiple classifier system [6] is based on the idea to combine several classifiers such that the combined system achieves better performance than the individual ones. The base classifiers to be combined are required to be sufficiently diverse [5,6]. Data resampling and feature subset selection [5] are two common ways for

* We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (contract 213250).

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 155–164, 2010.

promoting the diversity of base classifiers. Data resampling methods, e.g. bagging and boosting [15], select different training samples for different base classifiers. In feature subset selection, base classifiers are trained using different subsets of features and their solutions are usually very different. Therefore, feature subset selection methods could yield better diversity than data resampling methods. There is another reason besides diversity for feature subset selection, that is, the curse of dimensionality problem. For a dataset with very high dimensionality, it is possible that one single classifier could not find a good solution in this high dimensional vector space. By feature subset selection, the problem can be solved in lower dimensional spaces and there is a higher chance to find better solutions.

However, for structural pattern recognition problems, patterns are not represented with only numerical features and there is also no direct way to embed graphs into a vector space, especially for the structural relationships within one object. Because the space of structural data, e.g., strings, trees or graphs, has not been properly vectorized yet, there are just a few attempts for building multiple classifier systems based on structural representations [1,12,2,8]. Obviously, feature subset selection methods are not applicable to train base classifiers for structural patterns as most other methods developed for statistical pattern recognition problems. But then the question arises what is a good alternative for increasing the diversity and maybe also avoiding the problem of high dimensionality for structural multiple classifier systems?

One of the few examples for creating structural base classifiers is discussed in [14]. The idea is to generate different graph-based classifiers by randomly removing nodes and their incident edges from the training graphs until a maximum number of nodes is reached for all graphs. Because of the randomness, different graph-based classifiers can be created and each becomes a base classifier in the multiple classifier system. However, with this setting, we still need to compute similarity/dissimilarity for labelled graphs using time-consuming techniques such as the maximum common subgraph [2] or the graph edit distance [9] considering a labelled graph classification problem. Unlike graphs with unlabelled nodes, graphs with labelled nodes usually need to be processed and described with more complicated algorithms and structures. Also, classifying graphs with labelled nodes is a more difficult task than classifying graphs with unlabelled nodes. Therefore, a method was proposed in [8] to decompose labelled graphs into sets of unlabelled subgraphs based on label information, and compare the dissimilarity between all pairs of subgraphs in order to create base classifiers in the dissimilarity space [10] for different labels.

Both existing methods described above for building structural multiple classifier systems only consider subgraphs for training base classifiers. One is selecting subgraphs randomly and the other is selecting subgraphs based on label information. Does this mean that subgraph selection is the best way for increasing the diversity of the base classifiers? Does subgraph selection somehow resemble feature subset selection?

In [8], we observed a very interesting phenomenon that all the combiners reaching the lowest error rate have at least one of the global structure base

classifiers as the base classifiers. So it is clear that the global structures can improve the classification performance. The global structure means that the linkages of the given graph are fully preserved. Therefore instead of subgraphs, the full graphs are used for creating structural base classifiers. This suggests that full graphs are beneficial to the multiple classifier system. Therefore, instead of selecting subgraphs for training base classifiers, we propose a method to alter the full graphs and train base classifiers based on different versions of altered full graphs. The goal for this work is to investigate the best way for building diverse structural base classifiers, i.e., whether we should select the subgraphs, alter the full graphs or adopt both.

To derive different full graphs from the same graph and transfer a labelled graph into an unlabelled one, the label information is utilized by us to alter the graph into different forms. The alteration is done by assigning weights to the edges and the label information is also embedded in these weights. We can further ignore the labels on the nodes once the label information is embedded on the edges.

The rest of the paper is organized as follows. A multiple classifier system utilizes the label information of graphs for altering full graphs in order to build structural base classifiers is proposed in Section 2. In Section 3, we recap the JoEig approach for comparing unlabelled graphs. Simulation results are presented in Section 4. Finally, a conclusion is given in Section 5.

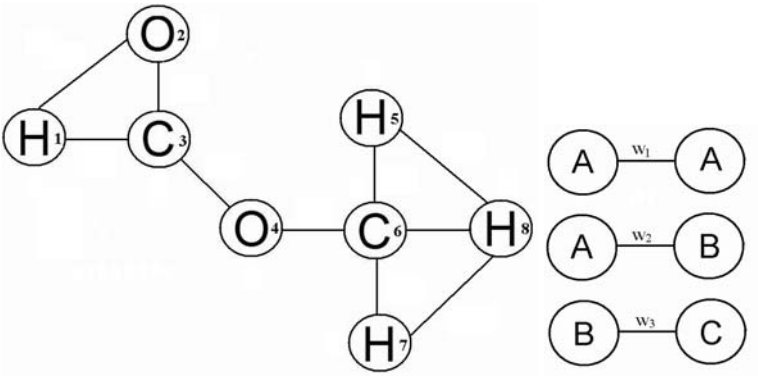
2 Building a Multiple Classifier System Using Altered Full Graphs

Before we introduce the altered full graphs for building structural base classifiers for a multiple classifier system, some definitions and an introduction on graphs are given as in the following.

A graph is a set of nodes connected by edges in its most general form. Consider the undirected graph $G = (V, E, W)$ with the node set $V = \{v_1, v_2, \ldots, v_n\}$, the edge set $E = \{e_1, e_2, \ldots, e_m\} \subset V \times V$, and the weight function $W : E \rightarrow (0, 1]$. If the graph edges are weighted, the adjacency matrix A for the graph G is the $n \times n$ matrix with elements

$$A_{ij} = \begin{cases} W(v_i, v_j), & \text{if } (v_i, v_j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Clearly since the graph is undirected, the matrix A is symmetric. The Laplacian of the graph is defined by $L = D - A$, where D is the diagonal node degree matrix whose elements $D_{ii} = \sum_{k=1}^{n} A_{ik}$. The Laplacian matrix of G is positive semidefinite and singular, and it is more often adopted for spectral analysis than the adjacency matrix because of its properties. We use the example graph shown in Figure 1(a) through this section to explain our method. This example graph is with 8 nodes and each node is labelled with one symbol. There are no attributes on the edges and the elements of the adjacency matrix A given in Eq.(2) of this



(a) (b)

Fig. 1. Examples of (a) a labelled graph; (b) possible linkage combinations in graphs according to label A

graph are either 1 or 0 to indicate whether there is an edge between two nodes or not.

$$A = \begin{pmatrix} 0\,1\,1\,0\,0\,0\,0\,0 \\ 1\,0\,1\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,1\,1\,1\,0 \end{pmatrix}. \tag{2}$$

2.1 Full Graph Alterations

Our goal is to solve the labelled graph classification problem by altering a labelled graph into a set of unlabelled graphs that preserve all the linkage structures. The alteration is given by assigning weights($\neq 0$) to the edges. In order to assign the weights in such a way that the new set of altered full graphs are diverse, the weights are given according to the node label information. For instance, if there are three different labels, i.e., A, B and C, in a graph, we can define three kinds of connection strengths according to label A as shown in Figure 1(b). Suppose label A is considered as the master label, the edge connects two nodes with both master label A will have the weight w_1. The edge connects one node with master label A and the other node which is not A will have the weight w_2, and finally the edge that connects two nodes that are both not master label will have the weight w_3. Without loss of generality, we assume that $1 \geq w_1 \geq w_2 \geq w_3 > 0$. By selecting different labels as the master label, the strengths of the connections

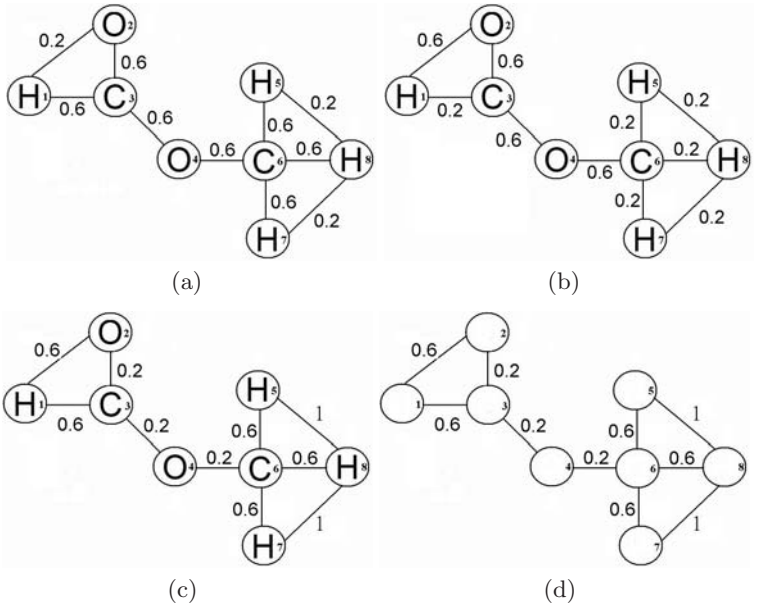


Fig. 2. Examples of altered full graphs with weights assigned based on label (a) C, (b) O and (c) H, respectively, from the graph in Figure 1(a), and (d) the unlabelled version of altered full graph based on label H

in a graph will change but the linkage structure will remain the same because the weights can not be equal to zero by assumption.

For the example in Figure 1(a), there are three different labels, i.e., C, H and O. For each label, we will assign weights to the edges according to this particular label. Let $w_1 = 1$, $w_2 = 0.6$ and $w_3 = 0.2$, Figure 2(a), Figure 2(b) and Figure 2(c) are with weights assigned according to label C, O, and H, respectively. Now that the label information is embedded to the edges with different weights, it means that we can ignore the label within the graph as in Figure 2(d) and fully describe this graph with a connection matrix (which is composed of the weights of the edges). For the example graph in Figure 2(a), its connection matrix A_C will be

$$A_C = \begin{pmatrix} 0 & 0.2 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0.6 & 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0.2 \\ 0 & 0 & 0 & 0.6 & 0.6 & 0 & 0.6 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0.2 \\ 0 & 0 & 0 & 0 & 0.2 & 0.6 & 0.2 & 0 \end{pmatrix}. \tag{3}$$

Notice that if w_1, w_2 an w_3 are all set to 1, the connection matrix A_C equals the adjacency matrix.

2.2 Dissimilarity and Base Classifiers

Given m graphs with n distinctive labels among the graphs, we want to create n base classifiers with respect to the labels. So, for a certain label, we derive an altered full graph and its connection matrix from each graph by assigning different weights as described above. With these m altered full graphs, the dissimilarities are calculated pairwise with the JoEig approach as described in the next Section. As a result, we can obtain an $m \times m$ dissimilarity matrix for each label. With this dissimilarity matrix, we can build a base classifier for this label in the dissimilarity space [10]. In the end, we can construct n label base classifiers by doing the same to each label. Dissimilarity space uses (selected) object dissimilarities as axes and objects as points. That is, axis 1 is the dissimilarity to object 1, axis 2 the dissimilarity to object 2 and so on. Object points are located in this space by their dissimilarities to each (selected) objects. These selected objects are also called the representation set. With this setting, we can project the objects into a vector space and build a classifier in it.

3 JoEig: Graph Comparison in Joint Eigenspace

JoEig [7] projects each pair of two graphs into a joint eigenspace. This joint eigenspace is expanded by both sets of eigenvectors.

Let G and H be weighted undirected graphs and L_G and L_H be their Laplacian matrices, respectively. The eigendecomposition of L_G and L_H are performed as $L_G = V_G D_G V_G^T$ and $L_H = V_H D_H V_H^T$ where V_G and V_H are orthonormal matrices and D_G and D_H are diagonal matrices of the eigenvalues (in ascending order) of G and H, respectively. With the joint projection vector $V_G V_H^T$, both graphs G and H will be projected to their joint eigenspace as $L_G V_G V_H^T$ and $V_G V_H^T L_H$. The difference between two graphs using JoEig is defined as $\|V_G D_G V_H^T - V_G D_H V_H^T\|^2$. The JoEig approach approximates a graph by relocating its eigenvalues in the joint eigenspace constructed by the eigenvectors of both graphs.

There are also three possibilities for setting the number of eigenvectors to compare graphs with different sizes in JoEig. In this work, we choose to make full use of the eigenvectors from the smaller graph and keep the same number of eigenvectors and eigenvalues in the larger graph as in the smaller graph by removing less important eigenvalues and eigenvectors from the larger graph.

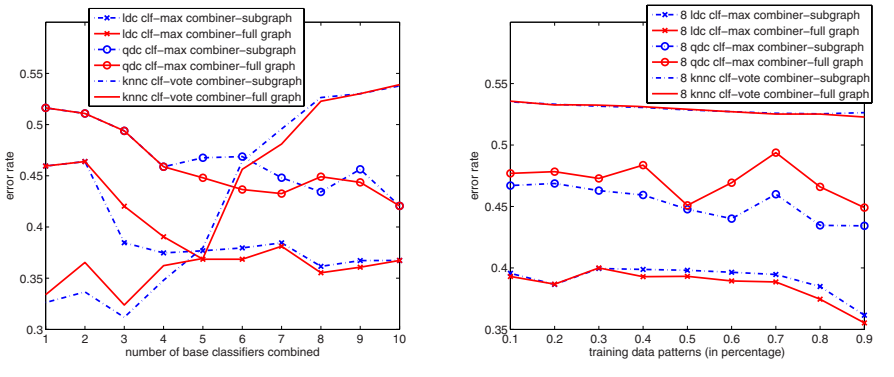
4 Experiments

In this section, we compare the performance of the multiple classifier systems built on the subgraphs and the altered full graphs, respectively. Linear discriminant classifier (ldc), quadratic discriminant classifier (qdc) and k-nearest neighbor classifier (knnc) are adopted to build base classifiers in the dissimilarity space [10], respectively. For knnc, the 3 nearest neighbors are considered. All the base

classifiers and the classifier combiner are built with the PRTOOLS [4]. Two real-world datasets, i.e., Mutagenicity and AIDS [13], are used in the experiments where 60% of objects are randomly selected and used as the training and testing datasets, 20% are used as the validation set for indicating the performance of individual base classifiers, and the other 20% are used as the other validation set for searching the best values for weights, i.e., w_1,w_2, and w_3. We randomly select 15% of training objects and use them as the representative objects to construct the dissimilarity space for both datasets. Also, the eigenvalue diagonal and eigenvector matrices are resized to the size of the smaller graph with the JoEig approach. Moreover, all the results in the following are the average over 50 repetitions of experiments resulting in a very small standard deviation.

4.1 Experiment 1: Mutagenicity Dataset

Mutagenicity is one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 10 different symbols in total. The average number of nodes of a graph is 30.3 ± 20.1, and the average number of edges is 30.7 ± 16.8. The Mutagenicity dataset is divided into two classes, i.e., mutagen and nonmutagen. There are in total 4,337 elements (2,401 mutagen elements and 1,936 nonmutagen elements). In the experiments, 50% of objects are randomly selected and used as the training dataset. In Figure 3 (a), we add the base classifiers (10 base classifiers from subgraphs or altered full graphs) one by one. At each step, the base classifier performing best against the validation set will be selected as the next base classifier to be added. The max combination rule is used for ldc



(a) (b)

Fig. 3. Compare subgraphs and full graphs for building base classifiers with (a) combination results of different number of base classifiers and (b) learning curves of combining best 8 base classifiers for Mutagenicity dataset

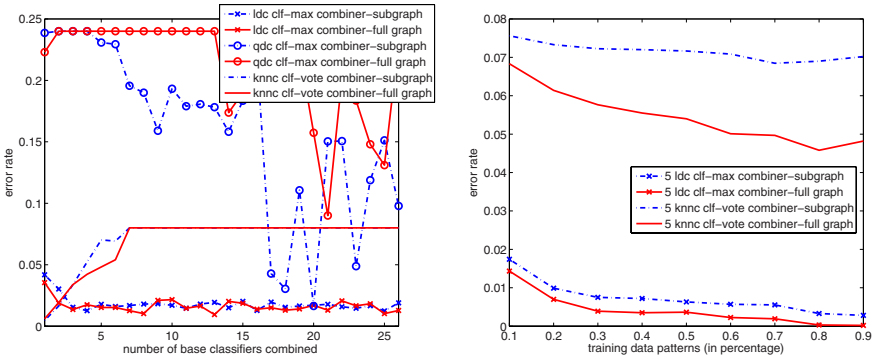
and qdc while the voting combination rule is used for knnc. For chemical compounds, atoms 'C' and 'H' are very common elements among and within objects. Especially for atom 'C', the full graphs or subgraphs constructed based on label 'C' can preserve most structures of the original graphs and therefore the base classifier constructed on label 'C' usually has the best individual performance. At first, knnc has base classifiers that are significantly better than ldc and qdc, which means the data distribution is rather nonlinear and knnc is more suitable for such a problem. But knnc is easily over-trained and also has unreliable confidence and therefore its performance decreases dramatically when more and more base classifiers are combined. If the max combination rule is used for knnc instead of voting, the performance of knnc will decrease even faster. Nevertheless, combining the best 3 individual knnc base classifiers can reach the optimal performance which is much better than all the combination results of ldc and qdc. Therefore, it might be beneficial to use knnc as base classifiers but selecting the best set of base classifiers is a crucial problem.

From Figure 3(a), we can see that base classifiers built on full graphs give better combination results than base classifiers built on subgraphs when more base classifiers are combined. However, with a few base classifiers, subgraphs perform better than full graphs with ldc and knnc. This means full graphs give more information about the structure of the original graph when more different labels are considered. On the other hand, the base classifiers built on subgraphs are more diverse in the beginning. We can also observe from Figure 3(a) that for ldc and qdc, the performance increases when there are more and more base classifiers combined. Because adding base classifiers is like adding features, when there is a sufficient number of objects, combining different base classifiers yields higher possibilities of having better performance than individual classifiers.

To fairly investigate the limitations and capabilities of subgraphs and full graphs, the learning curves of combining the best 8 base classifiers for both methods are drawn in Figure 3(b). Clearly, we can see that the subgraphs work better with small sample sizes and the full graphs on the other hand are better with medium and large sample sizes for ldc. Since graphs are usually with complex structures, it is possible to overfit when the number of objects is not sufficiently large. With subgraph selection, the structures are decomposed into simpler format and this problem might be avoided. In Figure 3(a), we can see that when the number of base classifiers is 8, using full graphs for qdc is worse than using subgraphs and therefore, we can also expect the same from Figure 3(b).

4.2 Experiment 2: AIDS Dataset

The AIDS dataset consists of graphs representing molecular compounds. The graphs are constructed from the AIDS Antiviral Screen Database of Active Compounds (molecules). This dataset consists of two classes, active and inactive, to indicate molecules with activity against HIV or not. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol, and there are 26 labels in total. The average number of nodes of



(a) (b)

Fig. 4. Compare subgraphs and full graphs for building base classifiers with (a) combination results of different number of base classifiers and (b) learning curves of combining best 5 base classifiers for AIDS dataset

a graph is 15.6 ± 13.1, and the average number of edges is 16.1 ± 15.0. There are 2,000 elements in total (1,600 inactive elements and 400 active elements). In the experiments, 50% of objects are randomly selected and used as the training dataset.

In Figure 4(a), the base classifiers (26 base classifiers from subgraphs or altered full graphs) are added one by one using the same technique as described above. The AIDS dataset is a much easier dataset to classify compared to the Mutagenicity dataset, and the best individual ldc classifiers built on subgraphs and full graphs, respectively, already reach very small error rates which makes it difficult for the combiner to improve the individual performance. There is not much difference for both methods in combining different numbers of ldc classifiers. The qdc and knnc classifiers preform significantly much worse than the ldc classifier with this dataset. The learning curves of combining the best 5 base classifiers for both methods are drawn in Figure 4(b). We can see that the full graphs perform better than the subgraphs with a larger number of objects for knnc but the difference is rather small for ldc.

5 Discussions and Conclusions

We solve the labelled graph classification problem with the multiple classifier system by decomposing labelled graphs into unlabelled full graphs based on their labels and building base classifiers from the full graphs. The full graphs preserve the topology from the original graph and therefore carry more information than subgraphs. Therefore using full graphs is beneficial when there is a sufficient number of objects. On the other hand, because of the complex structure of graphs, it is possible to encounter the problem of high dimensionality. Adopting subgraphs is a better solution in this case.

For highly nonlinear problems, knnc is probably a good solution and it is actually commonly adopted in graph classification problems. Therefore, how to select a proper set of knnc classifiers to combine for graph classification problems could be a direction for future study.

References

1. Bunke, H., Irniger, C., Neuhaus, M.: Graph Matching - Challenges and Potential Solutions. In: Roli, F., Vitulano, S. (eds.) ICIAP 2005. LNCS, vol. 3617, pp. 1–10. Springer, Heidelberg (2005)
2. Bunke, H., Riesen, K.: Graph Classification Based on Dissimilarity Space Embedding. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSSPR 2008. LNCS, vol. 5342, pp. 996–1007. Springer, Heidelberg (2008)
3. Dijkstra, E.W.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1, 269–271 (1959)
4. Duin, R.P.W., Juszczak, P., Paclik, P., Pękalska, E., de Ridder, D., Tax, D.M.J.: PRTOOLS 2004, A Matlab Toolbox for Pattern Recognition. Delft University of Technology, ICT Group, The Netherlands (2004), http://www.prtools.org
5. Ho, T.K.: The Random Subspace Method for Constructing Decision Forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
6. Kuncheva, L.I.: Combining Pattern Classifiers. In: Methods and Algorithms. Wiley, Chichester (2004)
7. Lee, W.J., Duin, R.P.W.: An Inexact Graph Comparison Approach in Joint Eigenspace. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSSPR 2008. LNCS, vol. 5342, pp. 35–44. Springer, Heidelberg (2008)
8. Lee, W.J., Duin, R.P.W.: A Labelled Graph Based Multiple Classifier System. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 201–210. Springer, Heidelberg (2009)
9. Neuhaus, M., Bunke, H.: Edit Distance-Based Kernel Functions for Structural Pattern Classification. Pattern Recognition 39, 1852–1863 (2006)
10. Pękalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition. In: Fundations and Applications. World Scientific, Singapore (2005)
11. Qiu, H.J., Hancock, E.R.: Spectral Simplication of Graphs. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3024, pp. 114–126. Springer, Heidelberg (2004)
12. Riesen, K., Bunke, H.: Classifier Ensembles for Vector Space Embedding of Graphs. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 220–230. Springer, Heidelberg (2007)
13. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSSPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
14. Schenker, A., Bunke, H., Last, M., Kandel, A.: Building Graph-Based Classifier Ensembles by Random Node Selection. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 214–222. Springer, Heidelberg (2004)
15. Skurichina, M., Kuncheva, L.I., Duin, R.P.W.: Bagging and Boosting for the Nearest Mean Classifier: Effects of Sample Sizes on Diversity an Accuracy. In: Roli, F., Kittler, J. (eds.) MCS 2002. LNCS, vol. 2364, pp. 62–71. Springer, Heidelberg (2002)

A Support Kernel Machine for Supervised Selective Combining of Diverse Pattern-Recognition Modalities

Alexander Tatarchuk[1], Eugene Urlov[2], Vadim Mottl[1], and David Windridge[3]

[1] Computing Center of the Russian Academy of Sciences,Moscow, Russia
[2] Moscow Institute of Physics and Technology, Moscow, Russia
[3] Centre for Vision, Speech and Signal Processing,
University of Surrey, Guildford, UK

Abstract. The Support Kernel Machine (SKM) and the Relevance Kernel Machine (RKM) are two principles for selectively combining object-representation modalities of different kinds by means of incorporating supervised selectivity into the classical kernel-based SVM. The former principle consists in rigidly selecting a subset of presumably informative support kernels and excluding the others, whereas the latter one assigns positive weights to all of them. The RKM algorithm was fully elaborated in previous publications; however the previous algorithm implementing the SKM principle of selectivity supervision is applicable only to real-valued features. The present paper fills in this gap by harnessing the framework of subdifferential calculus for computationally solving the problem of constrained nondifferentiable convex optimization that occurs in the SKM training criterion applicable to arbitrary kernel-based modalities of object representation.

1 Introduction

In pattern recognition, the term "modality" is employed when speaking about a specific kind of mathematical computer-perceptible object representation. In terms of the measured modality, the hypothetical set of "all" real-world objects of interest $\omega \in \Omega$ is represented by the outputs of the respective sensor as generalized features $x(\omega) \in \mathbb{X}$ in some sensor-specific scale $\mathbb{X}$. In the simplest case, when the scale is the set of real numbers $\mathbb{X} = \mathbb{R}$, the objects are represented by values of a real numerical feature. Multimodal pattern recognition systems utilize several distinct feature modalities, often with different scales $\big(x_i(\omega) \in \mathbb{X}_i, i \in I = \{1, ..., n\}\big)$, to represent specific phenomena [1].

Feature scales $\mathbb{X}_i$ may be quite complicated, so that frequently the only way of treating real-world objects $\omega \in \Omega$ is via pair-wise comparison of their features $\big(x_i(\omega'), x_i(\omega'')\big)$ using modality-specific functions $K_i(x_i', x_i'')$ defined in the respective scales $\mathbb{X}_i \times \mathbb{X}_i \rightarrow \mathbb{R}$. A function $K(x', x'')$ is said to be a *kernel* if it forms a semidefinite matrix for any finite collection of objects. It is well known

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 165–174, 2010.

that a kernel embeds the scale of the respective feature $\mathbb{X}_i$ into a hypothetical linear space $\tilde{\mathbb{X}}_i \supseteq \mathbb{X}_i$ in which it plays the role of inner product.

In particular, when $x_i(\omega) \in \mathbb{X}_i = \mathbb{R}$, the natural kernel will be the product $K_i(x_i', x_i'') = x_i' x_i''$. Support Vector Machines (SVMs), originally designed for two-class pattern recognition learning in $\mathbb{R}^n$, actually combine real-valued modalities by employing a joint kernel $K(\boldsymbol{x}', \boldsymbol{x}'') = \sum_{i=1}^{n} x_i' x_i''$. This analogy is exploited by multi-kernel SVMs when more sophisticated kernel-represented modalities are to be combined [3,4,5].

When fusing several modalities of object representation, the necessity to moderate the inevitable overfitting threat makes it absolutely necessary to combine modality-specific features in a selective mode. We consider here the general case of kernel-induced feature scales $\left\{\tilde{\mathbb{X}}_1, ..., \tilde{\mathbb{X}}_n\right\}$ treated as hypothetical linear closures $\tilde{\mathbb{X}}_i \supseteq \mathbb{X}_i$ of arbitrary scales $\left\{\mathbb{X}_1, , ..., \mathbb{X}_n\right\}$ with respective kernels defined over each of them $\left\{K_i(x_i', x_i''),\ x_i', x_i'' \in \mathbb{X}_i\right\}$. The kernel-based approach removes the mathematical distinction between different kinds of feature scales $\tilde{\mathbb{X}}_i$, so that the kernel selection will boil down to the usual feature selection in the particular case of natively real-valued features $\tilde{\mathbb{X}}_i = \mathbb{X}_i = \mathbb{R}$.

There exist many feature (kernel) selection techniques classed in the literature as *filters*, which are applied to the feature set independently of classification technique, and *wrappers*, which consider feature selection in conjunction with classification [2].

It is the latter way of combining multiple kernels we keep to in this paper. More specifically, we further elaborate the methodology of selectivity supervision by *a priori* assigning the desired level of selectivity, ranging from the complete absence of selection to the adoption of only singular features. In our previous papers [6,7], a way of achieving this range of behaviours was roughly outlined as the idea of incorporating selectivity into the two-class kernel-based Support Vector Machine.

Two principles of incorporating selectivity into the SVM proposed in [6] were called Support Kernel Machine (SKM) and Relevance Kernel Machine (RKM). The former principle consists in rigidly selecting a subset of presumably informative support kernels and excluding the others, whereas the latter one assigns positive weights to all of them.

An algorithm for implementing the RKM principle of selectivity supervision is elaborated in [6] and tested in [7] on the practical problem of signature verification by kernel-based fusing on-line and off-line modalities of signature representation. However, the algorithm described in [6] is applicable only to real-valued features $x_i \in \mathbb{X}_i = \mathbb{R}$.

The purpose of the present paper is to fill in this gap. The idea consists in harnessing the framework of subdifferential calculus [10] for computationally solving the problem of constrained nondifferentiable convex optimization that occurs in the SKM training criterion applicable to arbitrary kernel-based modalities of object representation. This approach allows us to explicitly show the mechanism of selecting the support kernels and excluding the redundant ones relative to the given training set.

2 The Support Kernel and the Relevance Kernel Machines

Let $\{\boldsymbol{x}_j=(x_{1j},...,x_{nj}), y_j,\ j=1,...,N\}$ be the training set of real-world objects $\{\omega_j\in\Omega, j=1,...,N\}$ each of which is represented by the class-membership index $y_j = y(\omega_j) \in \{-1,1\}$ and the values of n modality-specific features measured in the respective scales $x_{ij} = x_i(\omega_j) \in \mathbb{X}_i$ with kernel functions $K_i(x_i', x_i'') : \mathbb{X}_i\times\mathbb{X}_i\to\mathbb{R}$ defined in them. A broad construction of the SVM was proposed in [5,6,7] as an instrument for making the Bayesian decision on the discriminant hyperplane $\sum_{i=1}^{n} K_i(a_i,x_i)+b \gtrless 0$ in the Cartesian product of the kernel-induced hypothetical linear space $\boldsymbol{a} = (a_1,...,a_n) \in \tilde{\mathbb{X}}_1\times ...\times\tilde{\mathbb{X}}_n$, $b\in\mathbb{R}$, with an arbitrary *a priori* density of orientation distribution $\Psi(\boldsymbol{a}) = \Psi(a_1,\ldots,a_n)$.

It was shown that, under some natural assumptions on the pair of class-specific *a priori* distribution densities $\varphi\big(\boldsymbol{x}|y=\pm1,(\boldsymbol{a},b)\big)$ defined by the same discriminant hyperplane in the combined linear feature space $\boldsymbol{x}=(x_1,...,x_n)\in\tilde{\mathbb{X}}_1\times\tilde{\mathbb{X}}_n$ (see [5,6,7] for details), the Bayesian estimate of the hyperplane parameters $(\boldsymbol{a},b)=(a_1,...,a_n,b)$ is the solution of the following optimization problem:

$$\begin{cases} -\ln\Psi(a_1,...,a_n)+c\sum\limits_{j=1}^{N}\delta_j \to \min\big(a_i\in\tilde{\mathbb{X}}_i, b\in\mathbb{R}, \delta_j\in\mathbb{R}\big), \\ y_j\left(\sum\limits_{i=1}^{n} a_i x_{ij}+b\right)\geqslant 1-\delta_j,\ \delta_j\geqslant 0,\ j=1,...,N. \end{cases} \tag{1}$$

It is only the penalty $-\ln\Psi(a_1,...,a_n)$ that distinguishes this generalized training criterion from the classical SVM $\sum_{i=1}^{n}a_i^2 + C\sum_{j=1}^{N}\delta_j \to \min\big(\boldsymbol{a}=(a_1,...,a_n)\in\mathbb{R}^n, b\in\mathbb{R}, \delta_1,...,\delta_n\in\mathbb{R}\big)$ for real-valued feature vectors $\boldsymbol{x}_j=(x_{1j},...,x_{nj})\in\mathbb{R}^n$.

Two parametric families of a priori densities $\Psi(a_1,\ldots,a_n\,|\,\mu)$ were proposed in [6] as two different means of endowing the training criterion (1) with the ability to emphasize informative object-representation modalities and suppress redundant ones under the desired selectivity level which grows with increasing parameter $\mu\geqslant 0$, starting from the full absence of selectivity (ie retaining all the original modalities when $\mu=0$).

These two parametric families had led in [6] to different modality-selective training criteria named the Relevance Kernel Machine (RKM) with supervised selectivity

$$\begin{cases} J_{RKM}(a_1,r_1,...,a_n,r_n,b,\delta_1,...,\delta_N\,|\,\mu) = \\ \sum\limits_{i=1}^{n}\big[\big(1/r_i\big)\big(K_i(a_i,a_i)+1/\mu\big)+\big(1/\mu+1+\mu\big)\ln r_i\big]+ \\ C\sum\limits_{j=1}^{N}\delta_j\to\min\big(a_i\in\tilde{\mathbb{X}}_i,\ r_i\in\mathbb{R},\ b\in\mathbb{R},\ \delta_j\in\mathbb{R}\big), \\ y_j\left(\sum\limits_{i=1}^{n}K_i(a_i,x_{ij})+b\right)\geqslant 1-\delta_j,\ \delta_j\geqslant 0,\ j=1,...,N,\ r_i\geqslant\varepsilon>0,\ i=1,...,n, \end{cases} \tag{2}$$

and the Support Kernel Machine (SKM) with supervised selectivity

$$\begin{cases} J_{SKM}(a_1,...,a_n,b,\delta_1,...,\delta_N\,|\,\mu) = \\ \sum\limits_{i=1}^{n} q(a_i\,|\,\mu) + C\sum\limits_{j=1}^{N}\delta_j \to \min\big(a_i\in\tilde{\mathbb{X}}_i,\ b\in\mathbb{R},\ \delta_j\in\mathbb{R}\big), \\ q(a_i\,|\,\mu) = \begin{cases} 2\mu\sqrt{K_i(a_i,a_i)} & \text{if } \sqrt{K_i(a_i,a_i)} \leqslant \mu, \\ \mu^2 + K_i(a_i,a_i) & \text{if } \sqrt{K_i(a_i,a_i)} > \mu, \end{cases} \\ y_j\left(\sum\limits_{i=1}^{n} K_i(a_i,x_{ij}) + b\right) \geqslant 1-\delta_j,\ \delta_j \geqslant 0,\ j=1,...,N. \end{cases} \tag{3}$$

We consider here only these two training criteria themselves and omit the Bayesian reasoning resulting from their respective a priori assumptions. The statistical justification is to be found in [6].

The Relevance Kernel Machine (2) and the Support Kernel Machine (3) are generalized versions of the classical SVM which implement two different principles of kernel-based modality selection.

The RKM emphasizes some modalities and relatively suppresses the others by assigning continuous positive weights $r_i > 0$ to the respective kernels $i \in I = \{1,...,n\}$ in the resulting discriminant hyperplane

$$\sum_{j:\ \lambda_j>0} y_j\lambda_j \sum_{i\in I} r_i K_i(x_{ij},x_i) \ + \ b \gtrless 0 \tag{4}$$

applicable to any new object $\boldsymbol{x}(\omega) = \big(x_i(\omega)\in\mathbb{X}_i, i=1,...,n\big)$.

Contrary to this, the SKM displays a pronounced inclination toward complete exclusion of a fraction of kernels. It partitions the entire set of modality-specific kernels into two subsets, that of support kernels $I_{supp} = \{i : r_i > 0\} \subseteq I$, which occur in the resulting discriminant hyperplane, and that of excluded ones $I \setminus I_{supp} = \{i : r_i = 0\}$.

3 A Smooth Dual Formulation of the Nondifferentiable SKM Training Problem

For any training set $\big\{(x_{ij}, i\in I), y_j,\ j=1,...,N\big\}$, where $I=\{1,...,n\}$ is the set of all modalities, the objective function $J_{SKM}(a_i, i\in I, b, \delta_j, j=1,...N\,|\,\mu)$ in (3) is convex in its range of definition $\tilde{\mathbb{X}}_1\times\tilde{\mathbb{X}}_n\times\mathbb{R}\times\mathbb{R}^N$, and the inequality constraints carve out a convex region in it. Thus, the SKM problem is that of convex optimization.

We denote as $\lambda_j \geqslant 0$ and $\pi_j \geqslant 0$ the Lagrange multipliers at the inequality constraints, respectively, $y_j\big(\sum_{i=1}^{n} K_i(a_i,x_{ij}) + b\big) - 1 + \delta_j \geqslant 0$ and $\delta_j \geqslant 0$. The convex problem (3) can be shown to be a regular one [10], and, so, it is equivalent to that of finding the saddle point of its Lagrangian

$$\begin{aligned} L(a_i, i\in I, b, \delta_j, \lambda_j, \pi_j, j=1,...N\,|\,\mu) = \frac{1}{2}J_{SKM}(a_1,...,a_n,b,\delta_1,...,\delta_N\,|\,\mu) - \\ - \sum_{j=1}^{N}\pi_j\delta_j - \sum_{j=1}^{N}\lambda_j\left[y_j\left(\sum_{i\in I}K_i(a_i,x_{ij})+b\right) - 1 + \delta_j\right] \to \\ \to \begin{cases} \min\big(a_i\in\tilde{\mathbb{X}}_i, i\in I,\ b\in\mathbb{R},\ \delta_j\in\mathbb{R}, j=1,...,N\big), \\ \max\big(\pi_j\geqslant 0, \lambda_j\geqslant 0,\ j=1,...,N\big). \end{cases} \end{aligned} \tag{5}$$

If $\left[\left(\tilde{a}_i, i\in I, \tilde{b}, \tilde{\delta}_j, , j=1,...N\right); \left(\tilde{\lambda}_j, \tilde{\pi}_j, j=1,...N\right)\right]$ is a saddle point, its left part $\left(\tilde{a}_i, i\in I, \tilde{b}, \tilde{\delta}_j, , j=1,...N\right)$ is a solution of the SKM problem (3), and vice versa, each of its solutions $\left(\tilde{a}_i, i\in I, \tilde{b}, \tilde{\delta}_j, , j=1,...N\right)$ is the left part of a saddle point of the Lagrangian (5).

Expanding the objective function in (5) in accordance with (3) gives the detailed expression of the Lagrangian:

$$L(a_i, i\in I, b, \delta_j, \lambda_j, \pi_j, j=1,...N\,|\,\mu) = \frac{1}{2}\left(\sum_{i\in I} q(a_i\,|\,\mu) + C\sum_{j=1}^{N}\delta_j\right) - \sum_{j=1}^{N}\pi_j\delta_j - \sum_{j=1}^{N}\lambda_j\left[y_j\left(\sum_{i\in I}K_i(a_i, x_{ij}) + b\right) - 1 + \delta_j\right]. \quad (6)$$

It is convenient to introduce special notations for each sum of constituents that depend on the ith modality-specific element a_i of the entire direction vector $\boldsymbol{a} = (a_1, ..., a_n)$:

$$L_i(a_i, \lambda_j, j = 1, ..., N\,|\,\mu) = \frac{1}{2}q(a_i\,|\,\mu) - K_i\left(a_i, \sum_{j=1}^{N} y_j\lambda_j x_{ij}\right). \quad (7)$$

In these terms, the Lagrangian (5) or (6) will have the form

$$\begin{aligned} &L(a_i, i\in I, b, \delta_j, \lambda_j, \pi_j, j=1,...N\,|\,\mu) = \\ &\sum_{i\in I} L_i(a_i, \lambda_j, j = 1, ..., N\,|\,\mu) + \sum_{j=1}^{N}\left(\frac{C}{2} - \pi_j - \lambda_j)\right)\delta_j - \\ &- \left(\sum_{j=1}^{N} y_j\lambda_j\right)b + \sum_{j=1}^{N}\lambda_j\,. \end{aligned} \quad (8)$$

Finding the saddle point of the Lagrangian is equivalent to maximizing the dual function of the Lagrange multipliers

$$\begin{aligned} &W(\lambda_j, \pi_j, j=1, ..., N\,|\,\mu) = \\ &\sum_{j=1}^{N}\lambda_j + \min_{a_i\in\tilde{\mathbb{X}}_i,\, b\in\mathbb{R},\, \delta_j\in\mathbb{R}} L(a_i, i\in I, b, \delta_j, \lambda_j, \pi_j, j=1,...N\,|\,\mu)\,. \end{aligned} \quad (9)$$

However, the minimum value of the second term in (8) exists only if the Lagrange multipliers satisfy the inequalities $C/2 - \pi_j - \lambda_j = 0$, or, with the restrictions $\pi_j \geqslant 0$,

$$0 \leqslant \lambda_j \leqslant \frac{C}{2},\ j = 1, ..., N. \quad (10)$$

Analogously, the third term of (8) has the minimum only if

$$\sum_{j=1}^{N} y_j\lambda_j = 0. \quad (11)$$

Thus, the dual function

$$W(\lambda_j, j=1,...,N\,|\,\mu) = \sum_{j=1}^{N} \lambda_j + \sum_{i\in I} \min_{a_i\in\tilde{\mathbb{X}}_i} L_i(a_i, \lambda_j, j=1,...,N\,|\,\mu) \qquad (12)$$

is to be maximized under constraints (10) and (11).

To accomplish the formulation of the dual problem, it is required to determine how the minimum values of the functions $L_i(a_i, \lambda_j, j = 1,...,N \mid \mu)$ (7) with respect to a_i depend on the Lagrange multipliers λ_i for each of the modalities $i \in I$. But these functions contain, in their turn, nondifferentiable functions $q(a_i\,|\,\mu)$ (3), which makes it necessary to use the notions of subgradient and subdifferential, instead of the usual gradient, to formulate the minimum condition of a convex function [10].

Definition 1. *Vector $d \in \tilde{\mathbb{X}}$ in a linear space $\tilde{\mathbb{X}}$ with inner product $K(x', x'')$ is called a subgradient of the convex function $f: \tilde{\mathbb{X}} \to \mathbb{R}$ at point $a \in \tilde{\mathbb{X}}$ if the inequality $f(x) - f(a) \geqslant K(d, x-a)$ holds for all $x \in \tilde{\mathbb{X}}$.*

Definition 2. *The set of all subgradients of convex function $f: \tilde{\mathbb{X}} \to \mathbb{R}$ at point $a \in \tilde{\mathbb{X}}$ is called the subdifferential $\partial f(a) \subseteq \tilde{\mathbb{X}}$ at this point.*

Property. *The condition that the subdifferential at point $a \in \tilde{\mathbb{X}}$ contains the null element $\phi \in \partial f(a) \subseteq \tilde{\mathbb{X}}$ is necessary and sufficient for this point to be a minimum point of convex function f.*

The latter property creates a mathematical basis for a closed form of the smooth optimization problem (12) dual to the original nondifferentiable SKM problem (3). This is a problem of maximizing a linear function of $N + n$ variables, namely, N Lagrange multipliers λ_j and n auxiliary variables ξ_i, under quadratic and linear constraints.

Theorem 1. *The problem*

$$\begin{cases} W(\xi_i, i\in I, \lambda_j, j=1,...,N) = \dfrac{1}{2}\displaystyle\sum_{i\in I}\xi_i + C\sum_{j=1}^{N}\lambda_j \to \max, \\ \xi_i \leqslant \mu^2 - \displaystyle\sum_{j=1}^{N}\sum_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il})\lambda_j\lambda_l, \;\; \xi_i \leqslant 0, \;\; i \in I, \\ \displaystyle\sum_{j=1}^{N} y_j\lambda_j = 0;\; 0 \leqslant \lambda_j \leqslant \frac{C}{2}, \;\; j = 1,...,N, \end{cases} \qquad (13)$$

is dual to the SKM training problem (3).

The proof depends upon the following lemma which is a result of immediate application of the prevoiusly formulated property of an arbitrary nondifferentiable convex function to the functions $L_i(a_i, \lambda_j, j = 1,...,N\,|\,\mu)$ in (12).

Lemma 1. *The minimum of function* $L_i(a_i, \lambda_j, j = 1, ..., N \,|\, \mu)$ *(7) with respect to variable* $a_i \in \tilde{\mathbb{X}}_i$ *is reached at the points*

$$\begin{cases} \tilde{a}_i = \eta_i \sum\limits_{j=1}^{N} y_j \lambda_j x_{ij}, \quad r_i = 1, & \text{if } K_i\left(\sum\limits_{j=1}^{N} y_j \lambda_j x_{ij},\ \sum\limits_{j=1}^{N} y_j \lambda_j x_{ij}\right) > \mu^2, \\ \tilde{a}_i = \eta_i \sum\limits_{j=1}^{N} y_j \lambda_j x_{ij}, 0 \leqslant r_i \leqslant 1, & \text{if } K_i\left(\sum\limits_{j=1}^{N} y_j \lambda_j x_{ij},\ \sum\limits_{j=1}^{N} y_j \lambda_j x_{ij}\right) = \mu^2, \\ \tilde{a}_i = \phi_i, & \text{if } K_i\left(\sum\limits_{j=1}^{N} y_j \lambda_j x_{ij},\ \sum\limits_{j=1}^{N} y_j \lambda_j x_{ij}\right) < \mu^2, \end{cases} \tag{14}$$

defined in terms of the linear operations and the null element induced by the respective kernel $K_i(x', x'')$ *in the hypothetical linear space* $\tilde{\mathbb{X}}_i$*. At each such point,*

$$\min_{a_i \in \tilde{\mathbb{X}}_i} L_i(a_i, \lambda_j, j = 1, ..., N \,|\, \mu) = L_i(\tilde{a}_i, \lambda_j, j = 1, ..., N \,|\, \mu) = \frac{1}{2} \min \left\{ 0;\ \mu^2 - K_i\left(\sum_{j=1}^{N} y_j \lambda_j x_{ij},\ \sum_{j=1}^{N} y_j \lambda_j x_{ij}\right) \right\}. \tag{15}$$

4 The Resulting Discriminant Hyperplane and Support Kernels

Assume the dual optimization problem (13) has been solved. Only the Lagrange multipliers $\lambda_1 \geqslant 0, ..., \lambda_N \geqslant 0$ are of interest, so the auxiliary values $\pi_1 \leqslant 0, ..., \pi_n \leqslant 0$ may be dropped. In accordance with (14), the discovered solution partitions the set of all kernels $I = \{1, ..., n\}$ into three subsets:

$$\begin{aligned} I^+ &= \left\{ i \in I : \sum_{j=1}^{N} \sum_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il}) \lambda_j \lambda_l > \mu^2 \right\}, \\ I^0 &= \left\{ i \in I : \sum_{j=1}^{N} \sum_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il}) \lambda_j \lambda_l = \mu^2 \right\}, \\ I^- &= \left\{ i \in I : \sum_{j=1}^{N} \sum_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il}) \lambda_j \lambda_l < \mu^2 \right\}. \end{aligned} \tag{16}$$

Theorem 2. *The optimal discriminant hyperplane defined by the solution of the SKM training problem (3) has the form*

$$\sum_{j:\lambda_j > 0} y_j \lambda_j \left(\sum_{i \in I^+} K_i(x_{ij}, x_i) + \sum_{i \in I^0} r_i K_i(x_{ij}, x_i) \right) + b \geqslant 0, \tag{17}$$

where the numerical parameters $\{0 \leqslant r_i \leqslant 1,\ i \in I^0;\ b\}$ *are solutions of the linear programming problem*

$$\begin{cases} 2\mu^2 \sum\limits_{i \in I^0} r_i + C \sum\limits_{j=1}^{n} \delta_j \to \min(r_i, i \in I^0;\, b;\, \delta_1, \ldots, \delta_N), \\ \sum\limits_{i \in I^0} \left(\sum\limits_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il}) \lambda_l \right) r_i + y_j b + \delta_j \geqslant 1 - \sum\limits_{i \in I^+} \sum\limits_{l=1}^{N} y_j y_l K_i(x_{ij}, x_{il}) \lambda_l, \\ \delta_j \geqslant 0,\ j = 1, \ldots, N, \quad 0 \leqslant r_i \leqslant 1,\ i \in I^0. \end{cases} \tag{18}$$

5 The Subset of Support Kernels

The solution $(\hat{r}_i, i \in I^0;\, \hat{b};\, \hat{\delta}_1, \ldots, \hat{\delta}_N)$ of the linear programming problem (18) is completely defined by the training set $\mathcal{X} = \left\{ \boldsymbol{x}_j = (x_{1j}, ..., x_{nj}), y_j,\ j = 1, ..., N \right\} \in \mathbb{X}_1 \times ... \times \mathbb{X}_n \subseteq \tilde{\mathbb{X}}_1 \times ... \times \tilde{\mathbb{X}}_n$. As is seen from criterion (18), some of coefficients $(\hat{r}_i, i \in I^0)$ may equal zero if the respective constraints $0 \leqslant r_i$ are active at the solution point.

However, it can be shown that, if all the linear spaces $\tilde{\mathbb{X}}_i$ are finite-dimensional, the subset of such configurations $\{\mathcal{X}\}$ is of zero Lebesgue measure in the linear space $\tilde{\mathbb{X}}_1 \times ... \times \tilde{\mathbb{X}}_n$. Thus, if the training set is considered as random points defined by a continuous probability distribution, the inequalities $\hat{r}_i > 0$ are met almost certainly for all $i \in I^0$.

This means that without any loss of generality the constraints $\{0 \leqslant r_i \leqslant 1,\ i \in I^0\}$ may be omitted in (18), and, yet, all kernels $i \in I^0$ will occur in the discriminant hyperplane (17) with nonzero weights. It is natural to call the subset $I_{supp} = I^+ \cup I^0 \subseteq I$ the set of support kernels .

The structure of the subsets of kernels (16) explicitly reveals how the subset of support kernels I_{supp} is affected by the parameter μ in the training criterion (3).

If $\mu = 0$, the set of evident support kernels $I^+ \subseteq I$ coincides with the entire set $I = \{1, \ldots, n\}$. In this particular case, the function $q(a_i \mid \mu)$ in (3) is quadratic $q(a_i \mid \mu) = const + K_i(a_i, a_i)$ for all $a_i \in \tilde{\mathbb{X}}_i$, and the training criterion does not differ from the usual SVM without selectivity properties; all the initial kernels are support ones because they all occur in the resulting decision rule.

As μ grows, more and more kernels appear in the set I^- of evident nonsupport kernels (16), and, correspondingly, the set of support kernels $I_{supp} = I^+ \cup I^0$ gets smaller.

Unlimited growth of the selectivity parameter $\mu \to \infty$ drives, finally, all the kernels into I^-, so that no support kernels remain at all: $I_{supp} = \varnothing$.

6 Adjusting the Selectivity Parameter

The selectivity parameter $0 \leqslant \mu < \infty$ is a structural parameter of the SKM training criterion. It determines a sequence of nested classes of training-set models whose dimensionality diminishes as μ grows, starting from the usual SVM model if $\mu = 0$. As it is not determined *a priori*, at present, the most effective method for choosing the value of a structural parameter is Cross-Validation that is based on directly estimating the generalization performance of the training method.

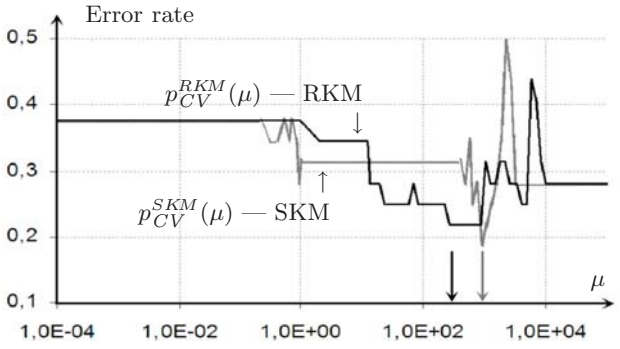


Fig. 1. The result of cross-validation on the lung cancer data set for increasing values of selectivity level μ

7 Experiments on Real-World Data

For the real data experiment, we used the lung cancer data set from the UCI repository [11]. The data set contains feature vectors of $N = 32$ patients partitioned into two subsets $N_{+1} = 9$ and $N_{-1} = 23$, respectively, those diagnosed and those not diagnosed with pathological lung cancer. Each vector consists of $n = 56$ features (a number exceeding the size of the available training set).

As the data set does not contain a test set, the relationship between the generalization performance of the algorithms and the selectivity level μ was estimated by the cross-validation method. The results of the experimental evaluation are shown in Fig. 1.

For small values of μ, both techniques are equivalent to the usual SVM applied to all $n = 56$ variables, so that, the respective error rates have the same value 0.38.

The minimum achievable error rate for the RKM is 0.187, whereas for the SKM it equals 0.219. For the optimal levels of selectivity μ, both techniques decrease their error rates by a factor of 2, but the weights estimated by RKM appreciably differ from zero at 4 features out of 56, whereas SKM retains only 2 of them. This extra feature dimensionality appears to be advantageous for the RKM over the SKM, with a minimum error rate of 0.187 against 0.219.

Finally, when μ becomes too large, both RKM and SKM remove all features, and the error rate of recognition tends towards an asymptotic level of 0.281 determined by the ratio N_1/N_{-1} between the numbers of representatives of the classes in the training set.

8 Conclusions

The Support Kernel Machine (SKM) and the Relevance Kernel Machine (RKM) are two different methods for selectively combining kernel-based modalities of arbitrary kind in multimodal pattern recognition. The former consists in rigidly selecting a subset of presumably informative support kernels and excluding the others, whereas the latter assigns positive weights to all the kernels.

The names Support *Kernel* Machine and Relevance *Kernel* Machine arise from an analogy with the distinction between the Support *Vector* Machine (SVM) [8]

and the Relevance *Vector* Machine (RVM) [9], which differ from each other by the binary verses weighted modelling of the occurance of the training-set objects in the linear decision rule.

The experimental evaluation indicates that the SKM and RKM methods display quite similar generalization performance albeit with a slight quantitative superiority attributable to the RKM. However, the SKM appears to produce this performance with a greater parsimony of modalities.

Acknowledgements

The research leading to these results has received funding from the Russian Foundation for Basic Research, grant no. 08-01-00695-a, and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 215078. We also gratefully acknowledge the support of EPSRC through grant EP/F069626/1.

References

1. Ross, A., Jain, A.K.: Multimodal biometrics: an overview. In: Proceedings of the 12th European Signal Processing Conference (EUSIPCO 2004), Vienna, Austria, pp. 1221–1224 (2004)
2. Guyon, I.M., Gunn, S.R., Nikravesh, M., Zadeh, L. (eds.): Feature Extraction, Foundations and Applications. Springer, Heidelberg (2006)
3. Sonnenburg, S., Rätsch, G., Schäfer, C.: A general and efficient multiple kernel learning algorithm. In: Proceedings of the 19th Annual Conference on Neural Information Processing Systems, Vancouver, Canada, December 5-8 (2005)
4. Sulimova, V., Mottl, V., Tatarchuk, A.: Multi-kernel approach to on-line signature verification. In: Proceedings of the 8th IASTED International Conference on Signal and Image Processing, Honolulu, Hawaii, USA, August 14-16 (2006)
5. Mottl, V., Tatarchuk, A., Sulimova, V., Krasotkina, O., Seredin, O.: Combining pattern recognition modalities at the sensor level via kernel fusion. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 1–12. Springer, Heidelberg (2007)
6. Tatarchuk, A., Mottl, V., Eliseyev, A., Windridge, D.: Selectivity supervision in combining pattern-recognition modalities by feature- and kernel-selective Support Vector Machines. In: Proceedings of the 19th International Conference on Pattern Recognition, Tampa, USA, December 8-11 (2008)
7. Tatarchuk, A., Sulimova, V., Windridge, D., Mottl, V., Lange, M.: Supervised selective combining pattern recognition modalities and its application to signature verification by fusing on-line and off-line kernels. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 324–334. Springer, Heidelberg (2009)
8. Vapnik, V.: Statistical Learning Theory. John Wiley & Sons, Inc., Chichester (1998)
9. Bishop, C.M., Tipping, M.E.: Variational relevance vector machines. In: Bishop, C.M., Tipping, M.E. (eds.) Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 46–53. Morgan Kaufmann, San Francisco (2000)
10. Hiriart-Urruty, J.-B., Lemarechal, C.: Fundamentals of Convex Analysis. Springer, Heidelberg (2001)
11. UCI Machine Learning Repository: Lung Cancer Data Set, http://archive.ics.uci.edu/ml/datasets/Lung+Cancer

Combining Multiple Kernels by Augmenting the Kernel Matrix

Fei Yan, Krystian Mikolajczyk, Josef Kittler, and Muhammad Atif Tahir

Centre for Vision, Speech, and Signal Processing
University of Surrey
Guildford, Surrey, GU2 7XH, UK
{f.yan,k.mikolajczyk,j.kittler,m.tahir}@surrey.ac.uk

Abstract. In this paper we present a novel approach to combining multiple kernels where the kernels are computed from different information channels. In contrast to traditional methods that learn a linear combination of n kernels of size $m \times m$, resulting in m coefficients in the trained classifier, we propose a method that can learn $n \times m$ coefficients. This allows to assign different importance to the information channel per example rather than per kernel. We analyse the proposed kernel combination in empirical feature space and provide its geometrical interpretation. We validate the approach on both UCI datasets and an object recognition dataset, and demonstrate that it leads to classification improvements.

1 Introduction

Since their introduction in the mid-1990s, kernel methods [1,2] have proven successful for many machine learning problems, e.g., classification, regression, dimensionality reduction, clustering. Representative methods such as support vector machine (SVM) [3,2], kernel Fisher discriminant analysis (kernel FDA) [4,5], kernel principal component analysis (kernel PCA) [6] have been reported to produce the state-of-the-art performance in numerous applications. In a kernel method, the choice of kernel is critically important, since the kernel completely determines the embedding of the data in the feature space. In many problems, multiple kernels capturing different "views" of the problem are available. In such a situation, one naturally wants to use these kernels in an "optimal" way.

Multiple kernel learning (MKL) was pioneered by Lancriet et al. in [7], where the key idea is to learn a linear combination of a given set of base kernels by maximising the (soft) margin between two classes or by maximising the "alignment" of the base kernels. In [7], the kernel weights are regularised with an ℓ_1 norm. Following this seminal work, MKL has become one of the most active areas in the machine learning community in the past few years. Various extensions have been made to [7]. For example, the efficiency of MKL is significantly improved in [8,9,10]; a multiclass version and a multilabel version are proposed in [11] and [12] respectively; in [13,14,15], the ratio of the inter- and intra- class scatters of FDA is maximised instead of the margin and kernel alignment; while

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 175–184, 2010.

in [16,17,18,15], ℓ_2 norm and even a general ℓ_p norm regularisation is considered instead of the ℓ_1 norm.

Despite the improvements achieved with these extensions both in terms of efficiency and accuracy, all these MKL methods share one limitation. To see this, let us consider an object categorisation problem as an example. Suppose the number of training samples is m and n training kernels of size $m \times m$ are available. Let these n kernels capture various aspects of the classification problem by using different features such as colour, texture, shape. Since all the MKL methods discussed above learn a linear combination of the base kernels, the learnt composite kernel also has a size $m \times m$. As a result, the learnt decision function has m coefficients, one for each training sample[1]. This means the contribution of a particular feature channel is fixed for all training samples. This is an unnecessarily strong constraint that does not allow to fully exploit the information from every sample. For example, one particular sample may carry more shape information than colour information, and vice versa for another sample. In a linear combination scheme, however, the shape information will be equally weighted in both training samples. Relaxing this constraint will allow to assign different weights to different samples depending on their importance in particular information channel. This effectively means that two different features extracted from the same sample are treated as two different samples of the same class.

In this paper, we present a learning approach that uses multiple kernels but, in contrast to existing MKL approaches, allows training samples to have different contributions in a particular feature channel. Instead of linear combination of the base kernels, we construct an $(n \times m) \times (n \times m)$ training kernel matrix. This leads to $n \times m$ coefficients in the trained decision function, in contrast to m coefficients in the linear combination scheme. As a result, the training samples contribute differently and the decision function is more flexible. We give the geometrical interpretation of our augmented kernel matrix (AKM) scheme and make comparison to that of the linear combination scheme. We show on several UCI datasets and an object recognition dataset that the AKM scheme can outperform linear combination of kernels.

The rest of this paper is organised as follows. We first introduce the concept of empirical feature space in Section 2 as it is important for understanding various kernel combination schemes. In Section 3 we briefly review the linear combination scheme. We then present our AKM scheme in Section 4 and discuss its connection to linear combination both algebraically and geometrically. Experimental results are provided in 5, which validate this new scheme. Finally conclusions are given in 6.

2 Empirical Feature Space

This section introduces the concept of empirical feature space that will be then used to discuss different methods for kernel combination. Let us for the moment

[1] More precisely, the decision function has $m + 1$ coefficients including a bias term b.

consider a single kernel case. We are given a symmetric, positive semi-definite (PSD) $m \times m$ training kernel matrix K and a corresponding $m \times l$ test kernel matrix $\dot{K}$, where K contains the pairwise dot products of the m training samples in some feature space, and $\dot{K}$ contains the pairwise dot products of the m training samples and the l test samples in the feature space. Note that this feature space usually has a very high or even infinite dimension and thus not directly tractable. However, it is shown in [19] that there exists an empirical feature space in which the intrinsic geometry of the data is identical to that in the true feature space, and for many machine learning problems, it suffices to study this empirical feature space.

To compute from K and $\dot{K}$ the training and test samples in the empirical feature space, consider the eigen decomposition of K:

$$K = V \Lambda V^T \tag{1}$$

where Λ is the $r \times r$ diagonal matrix containing the r ($r \leq m$) non-zero eigen values of K, and V is the $m \times r$ matrix containing the r associated eigen vectors. Note that since K is PSD, all the r non-negative eigenvalues of K are positive, and r is also the rank of K. It directly follows that

$$K = V \Lambda^{1/2} (\Lambda^{1/2})^T V^T = ((V \Lambda^{1/2})^T)^T (V \Lambda^{1/2})^T := X^T X \tag{2}$$

where the $r \times m$ matrix X is defined as

$$X = (V \Lambda^{1/2})^T \tag{3}$$

and its i^{th} column is the i^{th} training sample in the empirical feature space. Now let $\dot{X}$ be the $r \times l$ matrix whose i^{th} column is the i^{th} test sample in the empirical space. $\dot{X}$ is given by solving the following linear equation:

$$X^T \dot{X} = \dot{K} \tag{4}$$

We have shown in (3) and (4) given K and $\dot{K}$ how to find the training and test samples in the empirical feature space $\mathbb{R}^r$. In many practical situations, for example, in the case of Radial Basis Function (RBF) kernel, K is full rank, i.e. $r = m$. As a result, the m training samples X and l test samples $\dot{X}$ live in an m dimensional empirical feature space $\mathbb{R}^m$.

3 Linear Combination of Kernels

Now we turn to the case of multiple kernels. Assume we are given n training kernels $K_1, \cdots, K_n$ of size $m \times m$ and n corresponding test kernels $\dot{K}_1, \cdots, \dot{K}_n$ of size $m \times l$. In this section, we consider a linear combination of the base kernels:

$$K = \sum_{j=1}^{n} \beta_j K_j, \beta_j \geq 0 \tag{5}$$

Using the results from the previous section, each of these n kernels is associated with an empirical feature space:

$$K_j = X_j^T X_j$$
$$\dot{K}_j = X_j^T \dot{X}_j \tag{6}$$

where X_j and $\dot{X}_j$ are the training and test samples in the empirical feature space associated with the j^{th} kernel, respectively, and $X_j \in \mathbb{R}^{r_j}, \dot{X}_j \in \mathbb{R}^{r_j}$ for $j = 1, \cdots, n$ where r_j is the rank of K_j.

From the definition of dot product, it directly follows that taking the unweighted sum of the n base kernels is equivalent to taking the Cartesian product of the empirical feature spaces associated with the base kernels. On the other hand, taking the weighted sum of the base kernels as in Eq. 5 is equivalent to taking the Cartesian product of the base empirical feature spaces after scaling these spaces with $\sqrt{\beta_1}, \cdots, \sqrt{\beta_n}$. In this light, the goal of all MKL methods in [7,8,9,10,11,12,13,14,15,16,17,18] is to learn an optimal scaling such that some class separation criterion is maximised.

We illustrate the geometrical interpretation of taking the unweighted sum of two kernels in Fig. 1. Note that for the sake of visualisation we assume in Fig. 1 that the empirical feature spaces of both K_1 and K_2 are 1-dimensional, i.e., the ranks of both K_1 and K_2 are 1. In practice, however, both spaces can be up to m dimensional.

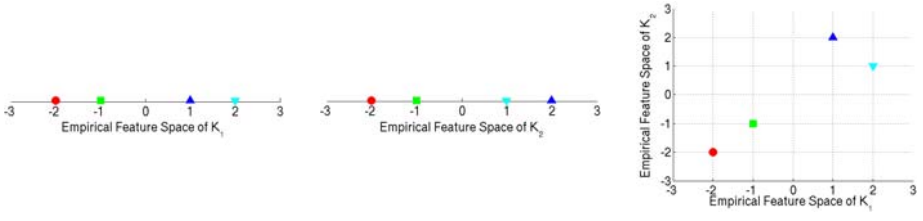


Fig. 1. Geometrical interpretation of taking the sum of two kernels. Left: the empirical feature space of K_1. Middle: the empirical feature space of K_2. Right: the empirical feature space of $K_1 + K_2$.

4 Kernel Combination with Augmented Kernel Matrix

Despite various ways of learning the optimal kernel weights, a linear combination of kernels leads to a composite kernel matrix $K = \sum_{j=1}^{n} \beta_j K_j$ which has a size $m \times m$. If SVM or kernel FDA is used as a classifier in the subsequent step, the decision function is in the form of:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{7}$$

where $K(\mathbf{x}, \mathbf{x}_i)$ is the dot product between a new test sample and the i^{th} training sample in the composite empirical feature space, $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_m)$ and b are learnt by maximising the margin (SVM) or by maximising the ratio between inter- and intra- class scatters (FDA). In both cases, there are m learnt coefficients $\boldsymbol{\alpha}$ if we ignore the bias term b, one for each training sample. This implies that the contribution of a given base kernel (thus a feature channel) is fixed for all training samples, which may be an unnecessarily strong constraint. For example, in an object recognition problem, one particular sample may carry more shape information than colour information and vice versa for another sample.

Instead of linear combination of kernels, we consider a different kernel combination scheme. We define an operation on two symmetric PSD training kernel matrices K_1 and K_2, $K_1 \oplus K_2$, as constructing an augmented block diagonal matrix K such that:

$$K = K_1 \oplus K_2 = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \tag{8}$$

The zeros on the off diagonal reflect the fact that we do not have any knowledge about the cross terms between the two kernel matrices.

Let the eigen decomposition of K_1 and K_2 be:

$$K_1 = V_1 \Lambda_1 V_1^T \tag{9}$$

$$K_2 = V_2 \Lambda_2 V_2^T \tag{10}$$

where Λ_1 and Λ_2 are the diagonal matrices containing the r_1 and r_2 non-zero eigen values of K_1 and K_2 respectively, and V_1 and V_2 are the $m \times r_1$ and $m \times r_2$ matrices containing the r_1 and r_2 associated eigen vectors, respectively:

$$V_1 = \{\mathbf{v}_1^1, \mathbf{v}_2^1, \cdots, \mathbf{v}_{r_1}^1\} \tag{11}$$

$$V_2 = \{\mathbf{v}_1^2, \mathbf{v}_2^2, \cdots, \mathbf{v}_{r_2}^2\} \tag{12}$$

where the m dimensional vector $\mathbf{v}_s^j$ is the s^{th} eigen vector of kernel K_j.

On the other hand, let the eigen decomposition of $K = K_1 \oplus K_2$ be:

$$K = V \Lambda V^T \tag{13}$$

Since K is a block diagonal matrix with K_1 and K_2 on its diagonal, Λ is a diagonal matrix containing the $r_1 + r_2$ eigen values of K, and these are simply the union of the r_1 eigen values of K_1 and the r_2 eigen values of K_2. Without loss of generality, we order Λ such that its first r_1 diagonal elements are the eigen values of K_1, and the last r_2 are those of K_2. Moreover, we order the $2m \times (r_1 + r_2)$ eigen vector matrix V accordingly:

$$V = \{\tilde{\mathbf{v}}_1^1, \tilde{\mathbf{v}}_2^1, \cdots, \tilde{\mathbf{v}}_{r_1}^1, \tilde{\mathbf{v}}_1^2, \tilde{\mathbf{v}}_2^2, \cdots, \tilde{\mathbf{v}}_{r_2}^2\} \tag{14}$$

Using again the property of block diagonal matrix, the columns of V are simply the eigen vectors of K_1 and K_2 padded with m zeros:

$$\tilde{\mathbf{v}}_s^1 = ({\mathbf{v}_s^1}^T, 0, \cdots, 0)^T \quad s = 1, \cdots, r_1 \tag{15}$$

$$\tilde{\mathbf{v}}_s^2 = (0, \cdots, 0, {\mathbf{v}_s^2}^T)^T \quad s = 1, \cdots, r_2 \tag{16}$$

Now the training vectors in the empirical feature spaces associated with K_1, K_2 and K, i.e., X_1, X_2 and X, can be computed using Eq. 3. Exploiting the relation between Λ_1, Λ_2 and Λ, and that between V_1, V_2 and V, it directly follows that X is an $(r_1 + r_2) \times 2m$ block diagonal matrix with X_1 and X_2 on its diagonal:

$$X = \begin{pmatrix} X_1 & 0 \\ 0 & X_2 \end{pmatrix} \tag{17}$$

where the $r_1 \times m$ matrix X_1 and $r_2 \times m$ matrix X_2 are the training vectors in the empirical feature spaces associated with K_1 and K_2, respectively.

The geometrical interpretation of this AKM scheme for kernel combination is illustrated in Fig. 2, where for the sake of visualisation we assume that the empirical feature spaces of both K_1 and K_2 are 1-dimensional. In practice, however, both spaces can be up to m dimensional. It is clear in Fig. 2 that by combining two kernels using the AKM scheme we have $2m$ training samples. This results in $2m$ coefficients in the decision function trained using the augmented kernel matrix, and as a result it allows training samples to have different contribution through the feature channels. We will show the benefit of this experimentally in the next section.

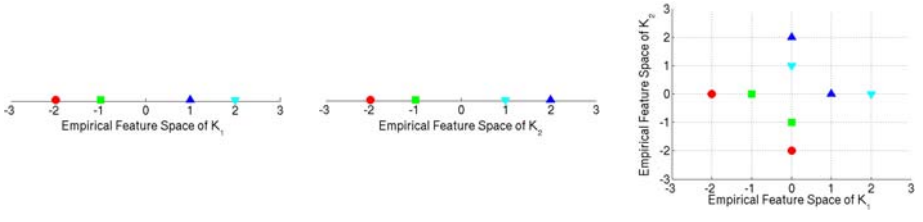


Fig. 2. Geometrical interpretation of augmenting the kernel matrix using Eq. 8. Left: the empirical feature space of K_1. Middle: the empirical feature space of K_2. Right: the empirical feature space of $K_1 \oplus K_2$.

For test kernels, the $\oplus$ operation is defined as:

$$\dot{K} = \dot{K}_1 \oplus \dot{K}_2 = \begin{pmatrix} \dot{K}_1 \\ \dot{K}_2 \end{pmatrix} \tag{18}$$

As a result the composite test kernel $\dot{K}$ has a size $2m \times l$. By applying the decision function, which has $2m$ coefficients, on $\dot{K}$, we obtain one score for each test sample.

5 Experiments

In this section, we validate the usefulness of the proposed AKM kernel combination scheme on both UCI datasets and an object recognition dataset. SVM and

kernel FDA are the two most popular kernel based classification methods. It has been shown [20,4] that SVM and kernel FDA have strong connections. In fact, the only difference between them is that SVM uses a hinge loss for computing the empirical loss while FDA uses a squared loss. In our experiments we choose kernel FDA as classifier to compare several kernel combination schemes: the ℓ_1 multiple kernel FDA (MK-FDA) of [14], the ℓ_2 MK-FDA of [15], ℓ_∞ MK-FDA where all the base kernels get equal weights, and the AKM scheme proposed in this paper.

5.1 UCI Datasets

We show in this section results on four datasets from the UCI machine learning repository [21], namely, *sonar*, *heart*, *iris* and *wine*. Among these datasets, the first two are binary problems while the last two are multiclass problems. For each dataset, we first normalise each feature in the input space to between -1 and 1. We then construct 10 RBF kernels using the normalised features with the following kernel function $K_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp^{-||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma^2}$, where σ is set to $\{10^{-1/2}, 10^{-1/3}, 10^{-1/6}, 10^0, 10^{1/6}, 10^{1/3}, 10^{1/2}, 10^{2/3}, 10^{5/6}, 10^1\}$. All the kernels are then centred in their empirical feature spaces [6]. For each dataset, we randomly split all samples (or equivalently the kernel matrix) into a training set and a test set using a ratio of 8 : 2. We repeat experiments 1000 times using 1000 random splits and report the mean error rate and standard deviation.

The first three methods under comparison all use linear combination of kernels. In ℓ_1 MK-FDA and ℓ_2 MK-FDA, the optimal kernel weights are learnt; while in ℓ_∞ MK-FDA the kernel weights are ones for all kernels. Once the kernel weights are obtained, the composite training kernel and test kernel can be computed. For the proposed AKM scheme, augmented training kernel and test kernel are constructed using Eq. 8 and Eq. 18, respectively.

Once the training and test kernels have been obtained using the four methods, we apply FDA to find the optimal projection and compute the classification error rate. In our experiments, the spectral regression based FDA implementation in [22] is employed for its efficiency. In this implementation, a γ parameter controls the trade-off between empirical error and generalisation of the decision function. For each dataset and each of the 1000 splits, we repeat 11 times using 11 γ values: $\{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^{+1}\}$.

We report the error rate and standard deviation of the four kernel combination methods in Table 1 and Table 2. For each γ value, we compute the mean error rate of the 1000 runs, and report in Table 1 the smallest error rate and the associated standard deviation for each method. We also select the optimal γ for each of the 1000 runs and report the error rate and standard deviation in Table 2. Briefly speaking, results in Table 1 and Table 2 are obtained with γ optimised over the mean of all 1000 runs and over each individual run, respectively.

From both tables we can see that MK-FDAs with different regularisation norms can be advantageous on different datasets. This is because different norm tend to produce kernel weights with different levels of sparsity: the smaller the norm, the higher the sparsity. As a result, MK-FDA with different norms are

suitable for kernel sets with various levels of intrinsic sparsity. On the other hand, the proposed AKM scheme outperforms all versions of MK-FDA, which are already the state-of-the-art classifiers, on two out of four datasets, and is comparable on the other two.

Table 1. Mean error rate. γ optimised over the mean of 1000 runs.

	ℓ_1 MK-FDA	ℓ_2 MK-FDA	ℓ_∞ MK-FDA	AKM FDA
sonar	13.5±5.0	14.2±5.2	13.9±5.1	**11.9 ± 4.6**
heart	17.5±4.7	**17.0 ± 4.6**	17.2±4.6	17.9±4.7
iris	5.1±3.8	4.7±3.5	4.6±3.6	**4.1 ± 3.2**
wine	5.6±9.7	**1.5 ± 2.0**	**1.5 ± 2.0**	2.5±2.6

Table 2. Mean error rate. γ optimised over each individual run.

	ℓ_1 MK-FDA	ℓ_2 MK-FDA	ℓ_∞ MK-FDA	AKM FDA
sonar	11.9±4.6	12.9±4.7	12.9±4.7	**9.9 ± 4.0**
heart	16.5±4.5	**16.1 ± 4.4**	16.3±4.4	16.6±4.4
iris	4.3±3.4	4.1±3.2	4.0±3.3	**2.9 ± 2.7**
wine	4.9±9.5	1.2±1.7	**1.1 ± 1.7**	1.9±2.3

5.2 Pascal VOC08 Dataset

The Pascal visual object classes (VOC) challenge provides a yearly benchmark for comparison of object recognition methods, with one of the most challenging datasets in the object recognition / image classification community [23]. The VOC 2008 development dataset consists of 4332 images of 20 object classes such as aeroplane, cat, person, etc. The set is divided into a pre-defined training set with 2111 images and a validation set with 2221 images. In our experiments, the training set is used for training and the validation set for testing.

The classification of the 20 object classes is treated as 20 independent binary problems. Average precision (AP) [24] is used to measure the performance of each binary classifier. The mean of the APs of the 20 classes, MAP, is used as a measure of the overall performance.

SIFT descriptor [25] and codebook technique [26] are used to generate kernels. The combination of two sampling techniques (dense and Harris-Laplace), five colour variants of SIFT descriptors [27], and three ways of dividing an image into spatial location grids results in $2 \times 5 \times 3 = 30$ base kernels.

We show in Table 3 the MAPs of the four kernel combination methods. The γ parameter is set to 0, with which optimal MAPs are achieved for all four methods. The poor performance of ℓ_1 MK-FDA indicates that the base kernels carry complementary information. In such a case, non-sparse kernel selection result is favoured since it does not lead to information loss. The proposed AKM scheme outperforms ℓ_2 and ℓ_∞ MK-FDAs by seemingly small margins. However, it is

Table 3. MAPs of the four kernel combination methods with 30 base kernels

	ℓ_1 MK-FDA	ℓ_2 MK-FDA	ℓ_∞ MK-FDA	AKM FDA
MAP	45.1	46.3	46.2	**46.4**

worth noting that a difference of 0.1 in MAP is more significant than it may appear to be. For example, the leading methods in PASCAL VOC classification competitions typically differ only by a few tenths of a percent in MAP. Moreover, uniform FDA was used by the method that produced the highest MAP in PASCAL VOC 2008 classification challenge [23]. This means the proposed AKM scheme improves over the state-of-the-art classifier for object recognition.

In both the experiments on UCI datasets and on VOC08 dataset, ℓ_1 and ℓ_2 MK-FDAs are implemented in Matlab and the associated optimisation problems are solved with the Mosek optimisation software [2]. The stopping threshold ϵ in ℓ_1 and ℓ_2 MK-FDAs is set to 5×10^{-4}.

6 Conclusions

In this paper we have presented a novel approach to combining multiple kernels where the kernels are computed from different information channels. In contrast to traditional methods that learn a linear combination of n kernels of size $m \times m$, resulting in m coefficients in the trained classifier, we propose a method that can learn $n \times m$ coefficients. This allows to assign different importance to the information channel per example rather than per kernel. We analyse the proposed kernel combination in empirical feature space and provide its geometrical interpretation. We validate the approach on both UCI datasets and an object recognition dataset, and demonstrate that it leads to classification improvements.

Acknowledgements

This work has been supported by EU IST-2-045547 VIDI-Video Project.

References

1. Scholkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
2. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
3. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1999)
4. Mika, S.: Kernel fisher discriminants. PhD Thesis, University of Technology, Berlin, Germany (2002)
5. Baudat, G., Anouar, F.: Generalized discriminant analysis using a kernel approach. Neural Computation 12, 2385–2404 (2000)

[2] http://www.mosek.com

6. Scholkopf, B., Smola, A., Muller, K.: Kernel principal component analysis. In: Advances in Kernel Methods: Support Vector Learning, pp. 327–352 (1999)
7. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.: Learning the kernel matrix with semidefinite programming. JMLR 5, 27–72 (2004)
8. Bach, F., Lanckriet, G.: Multiple kernel learning, conic duality, and the smo algorithm. In: ICML (2004)
9. Sonnenburg, S., Ratsch, G., Schafer, C., Scholkopf, B.: Large scale multiple kernel learning. JMLR 7, 1531–1565 (2006)
10. Rakotomamonjy, A., Bach, F., Grandvalet, Y., Canu, S.: Simplemkl. JMLR 9, 2491–2521 (2008)
11. Zien, A., Ong, C.: Multiclass multiple kernel learning. In: ICML, pp. 1191–1198 (2007)
12. Ji, S., Sun, L., Jin, R., Ye, J.: Multilabel multiple kernel learning. In: NIPS (2008)
13. Kim, S., Magnani, A., Boyd, S.: Optimal kernel selection in kernel fisher discriminant analysis. In: ICML (2006)
14. Ye, J., Ji, S., Chen, J.: Multi-class discriminant kernel learning via convex programming. JMLR 9, 719–758 (2008)
15. Yan, F., Kittler, J., Mikolajczyk, K., Tahir, A.: Non-sparse multiple kernel learning for fisher discriminant analysis. In: ICDM (2009)
16. Kloft, M., Brefeld, U., Laskov, P., Sonnenburg, S.: Non-sparse multiple kernel learning. In: NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels (2008)
17. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: Efficient and accurate lp-norm mkl. In: NIPS (2009)
18. Cortes, C., Mohri, M., Rostamizadeh, A.: L2 regularization for learning kernels. In: UAI (2009)
19. Scholkopf, B., Mika, S., Burges, C., Knirsch, P., Muller, K., Ratsch, G., Smola, A.: Input space versus feature space in kerenl-based methods. IEEE Transactions on Neural Networks 10(5), 1000–1017 (1999)
20. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer, Heidelberg (2002)
21. Newman, D., Hettich, S., Blake, C., Merz, C.: Uci repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
22. Cai, D., He, X., Han, J.: Efficient kernel discriminant analysis via spectral regression. In: ICDM (2007)
23. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results (2008), http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html
24. Snoek, C., Worring, M., Gemert, J., Geusebroek, J., Smeulders, A.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: ACM Multimedia Conference, pp. 421–430 (2006)
25. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. PAMI 27(10), 1615–1630 (2005)
26. Gemert, J., Geusebroek, J., Veenman, C., Smeulders, A.: Kernel codebooks for scene categorization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 696–709. Springer, Heidelberg (2008)
27. Sande, K., Gevers, T., Snoek, C.: Evaluation of color descriptors for object and scene recognition. In: CVPR (2008)

Class-Separability Weighting and Bootstrapping in Error Correcting Output Code Ensembles

R.S. Smith and T. Windeatt

Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, Surrey GU2 7XH, UK
{Raymond.Smith,T.Windeatt}@surrey.ac.uk

Abstract. A method for applying weighted decoding to error-correcting output code ensembles of binary classifiers is presented. This method is sensitive to the target class in that a separate weight is computed for each base classifier and target class combination. Experiments on 11 UCI datasets show that the method tends to improve classification accuracy when using neural network or support vector machine base classifiers. It is further shown that weighted decoding combines well with the technique of bootstrapping to improve classification accuracy still further.

1 Introduction

The use of error-correcting output code (ECOC) ensembles [5,8] has proved to be highly successful in solving multi-class classification problems. In this approach the multi-class problem is decomposed into a series of 2-class problems, or dichotomies, and a separate base classifier trained to solve each one. These 2-class problems are constructed by repeatedly partitioning the set of target classes into pairs of super-classes so that, given a large enough number of such partitions, each target class can be uniquely represented as the intersection of the super-classes to which it belongs. The classification of a previously unseen pattern is then performed by applying each of the base classifiers so as to make decisions about the super-class membership of the pattern. Redundancy can be introduced into the scheme by using more than the minimum number of base classifiers and this allows errors made by some of the classifiers to be corrected by the ensemble as a whole.

The operation of the ECOC algorithm can be broken down into two distinct stages - the coding stage and the decoding stage. The coding stage consists of applying the base classifiers to the input pattern $\mathbf{x}$ so as to construct vector of base classifier outputs $\mathbf{s}(\mathbf{x})$; the decoding stage consists of applying some decoding rule to this vector so as to make an estimate of the class label that should be assigned to the input pattern.

A commonly used decoding method is to base the classification decision on the minimum distance between $\mathbf{s}(\mathbf{x})$ and the vector of target outputs for each of the classes, using a distance metric such as Hamming or L^1. This, however, treats all base classifiers as equal, and takes no account of variations in their reliability. In

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 185–194, 2010.

this paper we describe a method for weighting the base classifier outputs so as to obtain improved ensemble accuracy. The weighting coefficients are computed from a statistic, known as the class-separability statistic. This algorithm assigns different weights to each base classifier and target class combination. Class-separability weighting (CSEP) was shown in [12] to be useful in the field of face-expression recognition. Here we show that it can also be beneficial when applied to general classification problems, as exemplified by 11 UCI datasets [9].

One of the advantages of the ECOC approach is that it makes it possible to perform multi-class classification by using base classifier algorithms that are more suited to solving 2-class problems. In this paper we investigate experimentally three types of base classifier, namely multi-layer perceptron (MLP) neural networks [1], Gaussian kernel support vector machines (SVMs) and polynomial kernel SVMs [3]. It is useful to regard each of these base classifier types as being controlled by two main parameters which respectively determine the *capacity* and the *training strength* of the learning algorithm. The term *capacity* [3] refers to the ability of an algorithm to learn a training set with low or zero training error. By *training strength* we mean the amount of effort that is put into training the classifier to learn the details of a given training set. For the three types of base classifier considered, the capacity parameter is, respectively, the number of hidden nodes, the Gaussian gamma parameter and the polynomial degree parameter. The training strength parameter is the number of training epochs for MLPs and the cost parameter for both types of SVMs.

A generally desirable property of multiple classifier systems, of which ECOC is an example, is that there should be diversity among the individual classifiers in the ensemble [2,11]. By this is meant that the errors made by component classifiers should, as far as possible, be uncorrelated so that the error correcting properties of the ensemble can have maximum effect. One way of encouraging this is to apply *bootstrapping* [7] to the training set so that each base classifier is trained on a unique bootstrap replicate. These are obtained from the original training set by repeated sampling with replacement. This creates a training set which has, on average, 63% of the patterns in the original set but with some patterns repeated to form a training set of the same size. Previous work [10] has shown that bootstrapping often reduces ensemble error and, in particular, it tends to avoid the problem of overfitting the data at high training strength values. A further potential benefit of bootstrapping is that each base classifier is trained on only a subset of the available training data and this leaves the remaining data, known as the out-of-bootstrap (OOB) set, to be used for other purposes such as parameter tuning. Note, however, that the OOB set is unique to each base classifier.

The remainder of this paper is structured as follows. The technique of applying class-separability weighting to the decoding of outputs from ECOC ensembles is described in detail in section 2. An experimental investigation of the effect of using this weighting scheme, with and without bootstrapping, is presented in section 3. Finally, section 4 summarises the conclusions to be drawn from this work.

2 ECOC Weighted Decoding

The ECOC method consists of repeatedly partitioning the full set of N classes Ω into L super-class pairs. The choice of partitions is represented by an $N \times L$ binary *coding matrix* $\mathbf{Z}$. The rows $\mathbf{Z}_i$ are unique *codewords* that are associated with the individual target classes ω_i and the columns $\mathbf{Z}^j$ represent the different super-class partitions. Denoting the jth super-class pair by S^j and $\overline{\mathrm{S}^j}$, element Z_{ij} of the coding matrix is set to 1 or 0[1] depending on whether class ω_i has been put into S^j or its complement. A separate base classifier is trained to solve each of these 2-class problems.

Given an input pattern vector $\mathbf{x}$ whose true class $y(\mathbf{x}) \in \Omega$ is unknown, let the soft output from the jth base classifier be $s_j(\mathbf{x}) \in [0,1]$. The set of outputs from all the classifiers can be assembled into a vector $\mathbf{s}(\mathbf{x}) = [s_1(\mathbf{x}), \ldots, s_L(\mathbf{x})]^{\mathrm{T}} \in [0,1]^L$ called the *output code* for $\mathbf{x}$. Instead of working with the soft base classifier outputs, we may also first harden them, by rounding to 0 or 1, to obtain the binary vector $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]^{\mathrm{T}} \in \{0,1\}^L$. The principle of the ECOC technique is to obtain an estimate $\hat{y}(\mathbf{x}) \in \Omega$ of the class label for $\mathbf{x}$ from a knowledge of the output code $\mathbf{s}(\mathbf{x})$ or $\mathbf{h}(\mathbf{x})$.

In its general form, a *weighted* decoding procedure makes use of an $N \times L$ weights matrix $\mathbf{W}$ that assigns a different weight to each target class and base classifier combination. The class decision, based on the L^1 metric, is made as follows:

$$\hat{y}(\mathbf{x}) = \arg\min_{\omega_i} \sum_{\mathrm{j}=1}^{\mathrm{L}} \mathbf{W}_{\mathrm{ij}} \left| \mathrm{s_j}(\mathbf{x}) - \mathbf{Z}_{\mathrm{ij}} \right|, \tag{1}$$

where it is assumed that the rows of $\mathbf{W}$ are normalized so that $\sum_{j=1}^{L} \mathbf{W}_{ij} = 1$ for $i = 1 \ldots N$. If the base classifier outputs $s_j(\mathbf{x})$ in eqn. 1 are replaced by hardened values $h_j(\mathbf{x})$ then this describes the weighted Hamming decoding procedure.

The values of $\mathbf{W}$ may be chosen in different ways. For example, if $\mathbf{W}_{ij} = \frac{1}{L}$ for all i, j then the decoding procedure of eqn. 1 is equivalent to the standard unweighted L^1 or Hamming decoding scheme. In this paper we make use of the class separability measure [11,12] to obtain weight values that express the ability of each base classifier to distinguish members of a given class from those of any other class.

In order to describe the class-separability weighting scheme, the concept of a correctness function must first be introduced: given a pattern $\mathbf{x}$ which is known to belong to class ω_i, the correctness function for the j'th base classifier takes the value 1 if the base classifier makes a correct prediction for $\mathbf{x}$ and 0 otherwise:

$$C_j(\mathbf{x}) = \begin{cases} 1 & \text{if } h_j(\mathbf{x}) = \mathbf{Z}_{ij} \\ 0 & \text{if } h_j(\mathbf{x}) \neq \mathbf{Z}_{ij} \end{cases}. \tag{2}$$

We also consider the complement of the correctness function $\overline{C}_j(\mathbf{x}) = 1 - C_j(\mathbf{x})$ which takes the value 1 for an incorrect prediction and 0 otherwise.

[1] Alternatively, the values +1 and -1 are often used.

For a given class index i and base classifier index j, the class-separability weight measures the difference between the positive and negative correlations of base classifier predictions, ignoring any base classifiers for which this difference is negative:

$$\mathbf{W}_{ij} = \max\left\{0, \frac{1}{K_i}\left[\sum_{\substack{\mathbf{p}\in\omega_i \\ \mathbf{q}\notin\omega_i}} C_j(\mathbf{p})\,C_j(\mathbf{q}) - \sum_{\substack{\mathbf{p}\in\omega_i \\ \mathbf{q}\notin\omega_i}} \overline{C}_j(\mathbf{p})\,\overline{C}_j(\mathbf{q})\right]\right\}, \qquad (3)$$

where patterns $\mathbf{p}$ and $\mathbf{q}$ are taken from a fixed training set T and K_i is a normalization constant that ensures that the i'th row of $\mathbf{W}$ sums to 1. The algorithm for computing $\mathbf{W}$ is summarised in fig. 1.

Inputs: matrix of training patterns $\mathbf{T} \in \mathbb{R}^{P\times M}$, binary coding matrix $\mathbf{Z} \in \{0,1\}^{N\times L}$, trained ECOC coding function $E : \mathbb{R}^M \mapsto [0,1]^L$.
Outputs: weight matrix $\mathbf{W} \in [0,1]^{N\times L}$ where $\sum_{j=1}^{L} \mathbf{W}_{ij} = 1$, for $i = 1 \ldots N$.
Apply E to each row of $\mathbf{T}$ and round to give prediction matrix $\mathbf{H} \in \{0,1\}^{P\times L}$.
Initialise $\mathbf{W}$ to $\mathbf{0}$.
for $c = 1$ to N
  for $i =$ indices of training patterns belonging to class c
    for $j =$ indices of training patterns *not* belonging to class c
      let d be the true class of the pattern $\mathbf{T}_j$.
      for k $= 1$ to L
        if $\mathbf{H}_{ik} = \mathbf{Z}_{ck}$ and $\mathbf{H}_{\mathrm{jk}} = \mathbf{Z}_{\mathrm{dk}}$, add 1 to $\mathbf{W}_{ck}$
          as the predictions for both patterns $\mathbf{T}_i$ and $\mathbf{T}_j$ are correct.
        if $\mathbf{H}_{ik} \neq \mathbf{Z}_{ck}$ and $\mathbf{H}_{\mathrm{jk}} \neq \mathbf{Z}_{\mathrm{dk}}$, subtract 1 from$\mathbf{W}_{ck}$
          as the predictions for both patterns $\mathbf{T}_i$ and $\mathbf{T}_j$ are incorrect.
      end
    end
  end
end
Reset all negative entries in $\mathbf{W}$ to 0.
Normalize $\mathbf{W}$ so that each row sums to 1.

Fig. 1. Pseudo-code for computing the class-separability weight matrix for ECOC

3 Experiments

In this section we present the results of performing classification experiments on 11 multi-class datasets obtained from the publicly available UCI repository [9]. The characteristics of these datasets in terms of size, number of classes and number of features are given in table 1.

Table 1. Experimental datasets showing the number of patterns, classes, continuous and categorical features

Dataset	Num. Patterns	Num. Classes	Cont. Features	Cat. Features
dermatology	366	6	1	33
ecoli	336	8	5	2
glass	214	6	9	0
iris	150	3	4	0
segment	2310	7	19	0
soybean	683	19	0	35
thyroid	7200	3	6	15
vehicle	846	4	18	0
vowel	990	11	10	1
waveform	5000	3	40	0
yeast	1484	10	7	1

For each dataset, ECOC ensembles of size 200 were constructed using each of three base classifier types and a range of capacity and training strength parameters. Each such combination was repeated 10 times with and without CSEP weighting and with and without bootstrapping. Each run used a different randomly chosen stratified training set and a different randomly generated ECOC coding matrix; for neural network base classifiers another source of random variation was the initial network weights. When bootstrapping was used, each base classifier was trained on a separate bootstrap replicate drawn from the complete training set for that run. The CSEP weight matrix was, in all cases, computed from the full training set. In each run the data was normalized so that the training set had zero mean and unit variance. The ECOC code matrices were constructed in such a way as to have balanced numbers of 1s and 0s in each column. Training sets were based on a 20/80 training/test set split.

The base classifier types employed were single-hidden layer MLP neural networks using the Levenberg-Marquardt training algorithm, SVMs with Gaussian kernel and SVMs with polynomial kernel. The MLPs were constructed as a single hidden layer of perceptrons, with the number of hidden nodes ranging from 2 to 16 and the number of training epochs from 2 to 1024. For Gaussian SVMs the width parameter gamma was varied between 1 and 8, whilst for polynomial SVMs degrees of 1,2,3 and 4 were used. The cost parameter of SVMs was varied between 10^{-3} and 10^{3}. In all cases, apart from polynomial degrees, the base classifier parameters were varied in geometric progression.

Table 2 compares the effect, on ensemble generalisation accuracy, of using CSEP weighted decoding and bootstrapping in different combinations. For each such combination and each base-classifier algorithm it shows the number of datasets for which rank 1 accuracy was achieved. It also shows the mean ranking, taken over the 11 datasets, achieved by each combination together with the mean best-case ensemble error and the percentage reduction in this

error[2]. The evidence of this table is that both bootstrapping and CSEP weighting on their own do tend to produce some improvement in classifier accuracy, with the latter algorithm being somewhat more effective than the former. This is shown by higher rank 1 counts, lower mean rank values and lower test errors. It is striking, however, that the greatest benefit is obtained when *both* techniques are used, indicating that their effects are mutually complementary so that they can be combined to good effect. It is also noticeable that, perhaps due to its more stochastic nature, the MLP base classifier shows the greatest reduction in mean test error, with the deterministic SVM classifiers benefitting to a lesser degree.

Table 2. Comparison of the best ensemble test error rates obtained from four combinations of algorithms; these were: standard ECOC, ECOC with bootstrapping (BS), ECOC with CSEP weighted decoding and ECOC with both bootstrapping and weighted decoding

	Standard	BS	CSEP	BS+CSEP
Rank 1 count (out of 11)				
MLP	0	1	3	**7**
Gaussian SVM	1	2	3	**7**
Polynomial SVM	1	1	3	**6**
Mean rank				
MLP	3.18	2.82	2.36	**1.64**
Gaussian SVM	3.09	2.91	2.18	**1.55**
Polynomial SVM	3.18	2.55	2.55	**1.73**
Mean best-case ensemble test error (%)				
MLP	16.54	16.23	16.16	**15.78**
Gaussian SVM	15.84	15.88	15.77	**15.65**
Polynomial SVM	16.98	16.81	16.88	**16.63**
Relative decrease in mean test error (%)				
MLP	-	1.87	2.30	**4.59**
Gaussian SVM	-	-0.25	0.44	**1.20**
Polynomial SVM	-	1.00	0.59	**2.06**

Further evidence for these findings can be seen in Fig. 2. This shows the mean ensemble test error, taken over all datasets, at the optimal base classifier capacity and over a range of training strength values. It is immediately apparent, from an inspection of this figure, that the best results tend to be obtained using CSEP weighted decoding and bootstrapping in combination. Bootstrapping alone tends to reduce ensemble error and also makes the ensemble less susceptible to overtraining at high values of the training strength parameter. When CSEP weighting is added to bootstrapping there is a further consistent reduction in ensemble error over the range of training strength values. This improvement tends

[2] Calculated as 100 x (original error - new error)/original error.

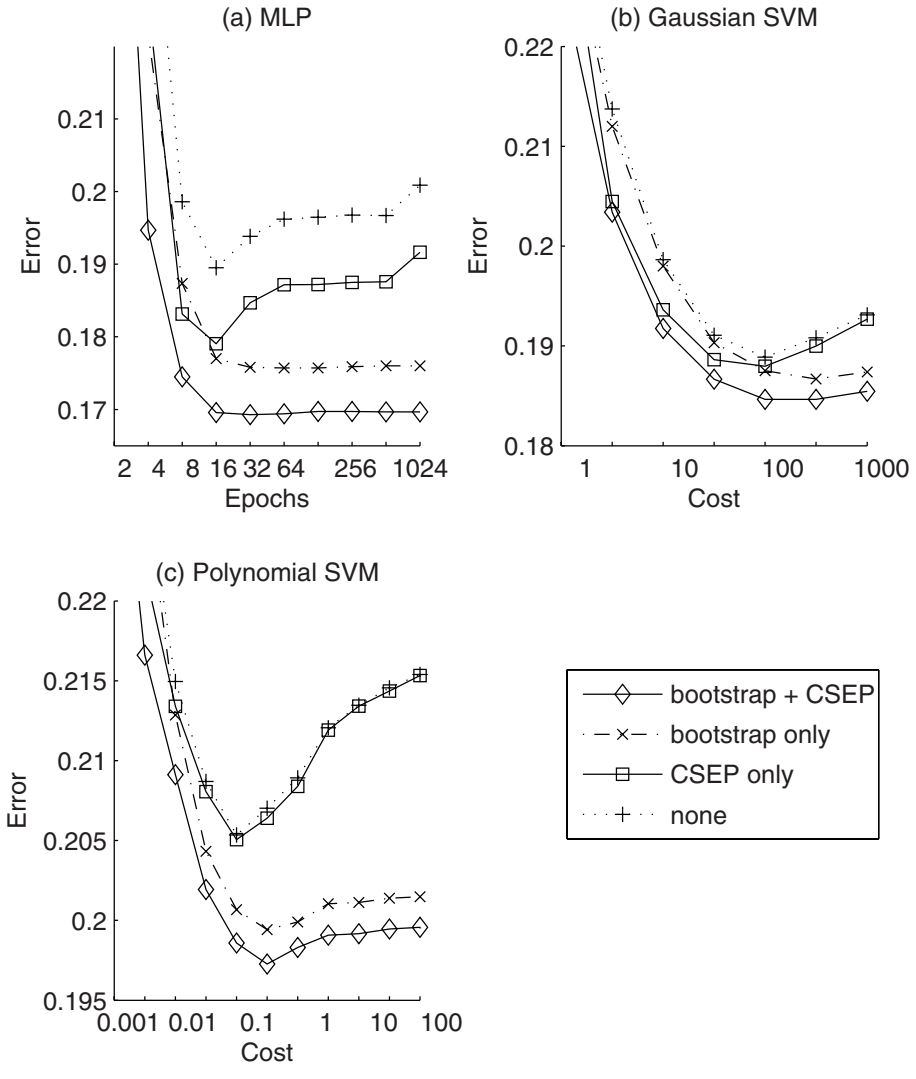


Fig. 2. The effects of class-separability weighting and bootstrapping on ensemble test error over a range of training strength values. These graphs show the mean error rate, taken over all datasets, at optimal base classifier capacity. The capacity parameters were set to: (a) 8 hidden nodes, (b) gamma = 4, (c) degree = 2.

to be most pronounced at low values of training strength, but is still observable at higher values of this parameter. In the absence of bootstrapping, CSEP weighting still leads to a reduction in ensemble error but the effect is more classifier dependent, with MLPs gaining the greatest benefit and polynomial SVMs the least. Again, the error reduction achieved by CSEP weighting is greatest at low values of training strength.

Table 3. Comparison of lowest ensemble error attained using standard ECOC and bootstrapped ECOC with weighted decoding (BS+CSEP). All values are expressed as percentages.

Data Set	MLP			Gaussian SVM			Polynomial SVM		
	Std. ECOC	BS + CSEP	relative decrease	Std. ECOC	BS + CSEP	relative decrease	Std. ECOC	BS + CSEP	relative decrease
dermatology	4.86	**3.07**	36.83	2.97	**2.90**	2.35	3.21	**2.97**	7.56
ecoli	17.48	**15.08**	13.73	14.66	**14.00**	4.50	15.68	**14.44**	7.89
glass	37.16	**36.64**	1.40	35.76	**35.69**	0.22	38.57	**37.71**	2.22
iris	5.25	**5.00**	4.76	5.83	**5.08**	12.86	**5.58**	5.75	-2.99
segment	**3.92**	3.94	-0.51	5.64	**5.59**	0.96	6.08	**5.69**	6.50
soybean	9.39	**9.04**	3.73	**7.90**	8.06	-2.10	**8.24**	8.25	-0.02
thyroid	2.57	**1.95**	24.12	2.77	**2.69**	2.94	3.39	**2.90**	14.54
vehicle	22.22	**20.76**	6.57	22.38	**22.04**	1.52	23.66	**23.08**	2.44
vowel	**21.14**	22.42	-6.05	**20.83**	20.86	-0.12	**25.85**	25.86	-0.05
waveform	16.70	**14.75**	11.68	14.48	**14.41**	0.45	14.59	**14.45**	0.97
yeast	41.22	**40.97**	0.61	41.02	**40.85**	0.41	41.90	**41.83**	0.18
mean	16.54	**15.78**	8.81	15.84	**15.65**	2.18	16.98	**16.63**	3.57

Table 4. Optimal numbers of hidden nodes for MLP base classifiers with and without bootstrapping plus weighted decoding. The rank 1 count shows the number of datasets for which each method required the smallest number of hidden nodes. Also shown is the mean optimal number of nodes across all 11 datasets.

	Standard	BS	CSEP	BS+CSEP
Rank 1 count (out of 11)	3	5	6	**8**
Mean number of nodes	9.5	9.1	8.4	**7.1**

To explain the behaviour shown in Fig. 2 we must consider separately the effects of bootstrapping and CSEP weighting. Bootstrapping operates during the ECOC coding stage and, by subsetting the training data, acts so as to reduce the degree to which each base classifier learns the distribution of this data. Whilst this may reduce base classifier accuracy, it nevertheless increases ensemble diversity and this can lead to an improvement in ensemble accuracy. This is particularly true at high training strengths where, without bootstrapping, there is a tendency to over-fit the data. By contrast, CSEP weighting operates during the ECOC decoding stage and serves to compensate for base classifier inaccuracy by individually weighting the base classifiers (on a per-class basis) so as to attach greater importance to the decisions of those classifiers that proved to be more accurate on the training set. Whilst this does tend to improve ensemble accuracy, it does nothing to solve the problem of overtraining at high training strengths. When bootstrapping is combined with CSEP weighting the benefits, we believe, are two-fold. Firstly, because the effects of the two techniques are essentially orthogonal, the advantages gained from using each method individually still

apply. A second consideration is that the CSEP weights matrix is more reliable by virtue of the fact that it is calculated on the full training set and this includes some data (the OOB set) which is independent of that used for base classifier training.

In the remainder of this section we look in more detail at the effects of applying bootstrapping and CSEP weighted decoding in combination. Table 3 shows the error levels measured on each of the test sets for each of the base classifier types when the base classifier parameters were optimised so as to minimise ensemble test error. Also shown is the percentage relative reduction in error achieved by bootstrapping plus CSEP weighting.

It can be seen from this table that, in the majority of cases (26/33), bootstrapping plus CSEP weighting did lead to a reduction in ensemble error. The size of this reduction was greatest when using an MLP base classifier but was nevertheless observable for the SVM base classifiers.

There is also evidence that, for MLP base classifiers, bootstrapping plus CSEP weighted decoding has the desirable property that it tends to require simpler classifiers with fewer hidden nodes than standard ECOC. Table 4 compares the number of hidden nodes required to minimise test error and it can be seen that bootstrapping and CSEP weighting individually lead to some improvement in these figures. As with ensemble error, however, the largest gain occurs when both techniques are used in combination.

4 Discussion and Conclusions

In this paper we have shown, by performing experiments on 11 multi-class datasets, that the techniques of bootstrapping and class-separability (CSEP) weighting each tend to reduce ECOC ensemble error. Bootstrapping affects the coding stage; it tends to increase diversity and to make the ensemble resistant to overfitting, especially at high values of the training strength parameter. CSEP weighting affects the decoding stage by taking account of the different performances of the base classifiers with respect to each target class.

It has been shown that these two algorithms complement each other and thus combine together well to produce a greater reduction in ensemble error than either of them individually. One reason for this may be related to the fact that a side-effect of bootstrapping is to reduce the training set of each base classifier to a subset of the available training set. It seems likely that this benefits CSEP weighting because the weight matrix, which is calculated using the full training set, will tend to be more representative because some of the training patterns (i.e. the OOB set) will not have been used for base classifier training. In effect this is similar to using a hold-out set for CSEP training.

The greatest benefit from CSEP weighting plus bootstrapping was observed when using MLPs as base classifiers. In this context it was also observed that the method has the desirable property that it tends to lead to simpler MLPs, requiring fewer hidden nodes for optimal performance. When deterministic base

classifier algorithms such as SVMs were used, CSEP weighting plus bootstrapping was still found to be of benefit but to a lesser degree.

Future work will focus on characterizing how CSEP weighting improves performance in terms of a bias-variance decomposition of error.

Acknowledgements

This work was supported by EPSRC grant E061664/1. Thanks are also due to the providers of the prtools [6] and libsvm [4] software.

References

1. Bishop, M.C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
2. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity Creation Methods: A Survey and Categorisation. Journal of Information Fusion 6(1) (2005)
3. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Knowledge Discovery and Data Mining 2(2) (1998)
4. Chang, C.-C., Lin, C.-J.: LIBSVM : a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
5. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
6. Duin, R.P.W., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D.M.J., Verzakov, S.: PRTools 4.1, A Matlab Toolbox for Pattern Recognition, Delft University of Technology (2007)
7. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, Boca Raton (1993)
8. James, G.: Majority Vote Classifiers: Theory and Applications. PhD Dissertation, Stanford University (1998)
9. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
10. Smith, R.S., Windeatt, T.: The Bias Variance Trade-off in Bootstrapped Error Correcting Output Code Ensembles. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 1–10. Springer, Heidelberg (2009)
11. Windeatt, T.: Accuracy/ Diversity and Ensemble Classifier Design. IEEE Trans. Neural Networks 17(4) (July 2006)
12. Windeatt, T., Smith, R.S., Dias, K.: Weighted Decoding ECOC for Facial Action Unit Classification. In: 18th European Conference on Artificial Intelligence (ECAI), Patras, Greece, July 2008, pp. 26–30 (2008)

Boosted Geometry-Based Ensembles

Oriol Pujol

[1] Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, 08007, Barcelona, Spain
[2] Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Barcelona, Spain

Abstract. Geometry-based ensembles is a newly proposed algorithm based on the concept of *characterizing boundary points*. These points are found from the geometry of the data set and belong to the optimal boundary between classes under a certain notion of robustness. The characterizing boundary points can be used to build a classifier. Based on these points, a set of locally robust linear classifiers is defined and assembled in an additive model to create a final decision rule. As a result a strong classifier able to compete with nowadays state-of-the-art classifiers is obtained. The main drawback of the original proposal comes from the fact that the complexity of the created model can be arbitrarily high and depends on the data set. Moreover, outliers and noise may increase this number. In this article, small complexity models with strong generalization capability are explored. Two incremental non-parametric additive building algorithms are considered: boosting and least squared residual fitting approaches. Moreover, the last method is extended to deal with incremental L2 penalized solutions (which implicitly combines the advantages of sparse models and smooth ones due to the complexity limit). The validation of the approach on the UCI database achieves very promising results assessing the validity of CBP based classifiers ensembles.

1 Introduction

The new challenges in the machine learning community such as online learning, adaptation to changing environments [7] or large scale learning [6] have increased the interest for adapting the state-of-the-art techniques as well as developing new techniques for dealing with these new scenarios.

In this context, geometry-based ensembles [1] appear as a new and simple classification algorithm with strong intuitive geometrical meaning. It automatically deals with the non-linear boundaries and its accuracy statistically outperforms some of the most well-known machine learning strategies, being on-par with kernel approaches. Geometry-based ensembles (GE) are built upon the notion of *characterizing boundary points* (CBP) — points that belong to the optimal boundary between classes following certain definitions of robustness and margin. Intuitively, an hyperspherical "area of influence" is laid around each data point, characterizing the amount of noise that a point is able to handle without

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 195–204, 2010.

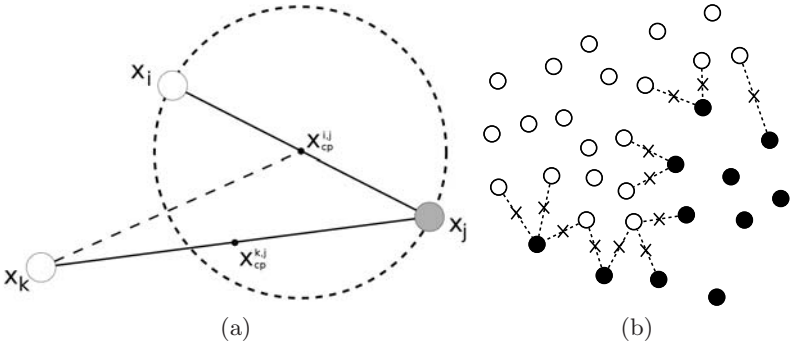


Fig. 1. (a) Illustration of the definition of characteristic boundary points. $x_{cp}^{i,j}$ is a CBP since there is no other point inside the hypersphere centered in it with radius $\|x_i - x_j\|/2$. (b) Example of CBPs (crosses)

ambiguity. The spatial locations where those "areas of influence" from different classes collide define the characteristic boundary points.

From these points a predefined set of classifiers can be obtained and assembled into an additive model. The optimization of the weights of the ensemble is formulated using a L2-penalized squared loss function constrained to non-negative weights, and solved using Singular Value Decomposition with filter factors.

Although the method is compelling, its main drawback comes from the moderate computational complexity, and the fact that there is no model complexity control. Moreover, in presence of very noisy data the amount of created classifiers can be very high with respect to those that are really useful.

In this article, incremental methods with explicit model complexity control using the geometry-based ensemble architecture are explored. Two incremental non-parametric additive building algorithms are considered: adaboost and least squared residual fitting. The last method is extended to deal with incremental L2 penalized solutions — which implicitly combines the advantages of sparse models and smooth ones due to early stopping.

The layout of the paper is as follows: in section 2, Geometry-based ensembles are briefly described. Section 3, provides the background of incremental function approximation and its adaptation to geometry-based ensembles. Section 4 validates the classification methods on fifteen data sets from the UCI repository. And, section 5 concludes the paper.

2 Geometry-Based Ensembles

Given a labelled training set of M points $S = \{(x_i, y_i)\}$, where $x_i \in \mathbb{R}^d$ belonging to class $y_i \in \{+1, -1\}$, $i = 1 \dots M$, a characteristic boundary point $x_{cp}^{i,j} \in \mathbb{R}^d$ is defined between any two training points (x_i, x_j) that fulfill the following conditions:

– **Necessary condition:** Both points (x_i, x_j) are from different classes.
– **Local optimality:** There is no closer example to the candidate boundary point x_{cp} than the ones that define it, namely (x_i, x_j).
Given any point $p : \{p \in \mathbb{R}^d | (p, l) \in S\}$, then,

$$\|x_i - x_{cp}\| \leq \|p - x_{cp}\| \tag{1}$$

$$\|x_j - x_{cp}\| \leq \|p - x_{cp}\| \tag{2}$$

– **Robustness to noise:** The candidate boundary point is located at the maximum distance from both generating points (the middle point between x_i and x_j).

$$x_{cp}^{i,j} = \frac{1}{2}(x_i + x_j) \tag{3}$$

Observing the CBPs one can easily see that by definition they are a set of points that lay in the locus of the space where no other point influence is present, thus they are good candidates for modelling the transition between classes. Figure 1(a) shows an illustration of the notion of CBP and Figure 1(b) depicts the CBPs for a toy example.

The original proposal of geometry-based ensembles (GE) builds an additive model using a linear combination of linear classifiers based on the CBPs. This set of classifiers are simply obtained as a hyperplane in the following way:

$$\pi_{x_{cp}^{i,j}}(x) = (x - x_{cp}^{i,j})\boldsymbol{n}_{x_{cp}^{i,j}}, \quad \boldsymbol{n}_{x_{cp}^{i,j}} = \frac{x_i - x_j}{\|x_i - x_j\|} \tag{4}$$

where $x_i : \{(x_i, y_i) \in S | y_i = +1\}$ and $x_j : \{(x_j, y_j) \in S | y_j = -1\}$ correspond to the elements of class labelled as $+1$ and -1, respectively.

The ensemble is created by means of an additive model $F : \mathbb{R}^d \rightarrow \mathbb{R}$ of base classifiers $h_k(x) = \text{sign}(\pi_k(x))$ related with the characterizing boundary points,

$$F(x) = \sum_{k=1}^{N} \rho_k h_k(x) = \sum_{k=1}^{N} \rho_k \text{sign}(\pi_k(x)) \tag{5}$$

where N is the number of CBP and sign stands for the signum function. The final decision rule can be simply obtained by thresholding the ensemble combination,

$$\hat{y} = \text{sign}(F(x) - \rho_0) \tag{6}$$

where $\hat{y}$ is the estimated label and ρ_0 is the global threshold value.

The influence of each base classifier in the ensemble is governed by the weighting vector ρ in Eq. 5. The optimization of the weighting vector is formulated as an L2 regularized problem in matrix form as follows,

$$\arg\min_{\rho} \|y - A\rho\|^2 + \lambda^2 \|\rho\|_2^2 \quad s.t. \quad \rho > 0 \tag{7}$$

where $A(k, i) = h_k(x_i) = \text{sign}(\pi_k(x_i)), k \in \{1 \ldots N\}, i \in \{1 \ldots M\}$ and y is the label vector of the training set, λ controls the weight of the residual norm. The non-negativity constraint ensures that the proces does not change the decision rule of any particular local classifier. The optimization is performed by means of Singular Value Decomposition and filter factors.

3 Incremental Optimization

In this article we explore and adapt to the GE framework some incremental model control techniques. In particular we explore the use of adaptive boosting and matching pursuit or gradient boosting with squared loss function for obtaining a controlled complexity ensemble.

3.1 AdaBoost Incremental Optimization

From the point of view of incremental optimization, Adaboost [5] and, in particular, its real version can be seen as a additive model fitting procedure that approximates the optimization of an exponential loss-function [4] $\mathcal{L}(y, F_t(x)) = e^{-yF_t(x)}$.

1 Initialize $D_1(i) = 1/M$.
for $t = 1 \dots T$: **do**
2 Train weak learner using distribution D_t
3 Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error $\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$.
4 Choose $\rho_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
5 Update weight distribution: $D_{t+1}(i) = \frac{D_t(i) e^{-\rho_t y_i h_t(x_i)}}{Z_t}$ where Z_t is a normalization factor.
6 Update model: $F_t(x) = F_{t-1}(x) + \rho_t h_t(x)$
end

Algorithm 1. Adaboost

The procedure is described in Algorithm 1. The adaptation of this process using the classifiers obtained by the GE is straightforward since in step 3, one simply has to select the one that attains minimum error from the set of predefined classifiers. One important practical consideration regards the fact that for this method to converge correctly, one must respect the original polarity of local classifiers as defined in Eq. 4.

3.2 Gradient Boosting

In [2] a general framework for attacking the problem of step-wise approximation building is presented. Let $F_{t-1}(x)$ be the approximation at step $t-1$ and y the desired value. We want to approximate y according to some loss function $\mathcal{L}(y, F_t(x))$

In the general case, this strategy replaces a potentially difficult function optimization problem $(\rho_t; a_t) = \arg\min_{\rho,a} \sum_{i=1}^{N} \mathcal{L}(y_i, F_{t-1}(x_i) + \rho h(x_i; a))$ by a two step process: first, learning the weak classifier using a least squares loss approximation (Algorithm 2 line 3), followed by weight optimization (Algorithm 2 line 4) based on the general loss criterion.

In the particular case of geometry-based ensembles, the weak classifiers are predefined and directly provided by the CBPs. Thus, as in the former case, the weak hypothesis learning process is replaced by a simple selection process. This allows to simplify the formulation since, instead of optimizing with respect to

1 $F_0(x) = \arg\min_\rho \sum_{i=1}^{N} \mathcal{L}(y_i, \rho)$
for $t = 1$ *to* T*:* **do**
2 $\quad \hat{y}_i = -\left[\frac{\partial \mathcal{L}(y_i, F(x_i))}{\partial F(x_i)}\right] \quad i = 1, M$
3 $\quad a_t = \arg\min_{a,\rho} \sum_{i=1}^{M} [\hat{y}_i - \rho h(x_i; a)]^2$ // Selection of the function in the family that best approximates the neg gradient
4 $\quad \rho_t = \arg\min_\rho \sum_{i=1}^{N} \mathcal{L}(\hat{y}_i, F_{t-1} + \rho h(x_i; a_t))$
5 $\quad F_t(x) = F_{t-1}(x) + \rho_t h(x; a_t)$
end

Algorithm 2. General Gradient Boost

two sets of parameters – the classifier parameters and the weight of the ensemble – we just have to select the candidate weak classifier and find the weight by minimizing with respect to the general loss function.

L2-penalized least squares incremental residual fitting. The simplest approach to gradient boosting optimization comes from the residual minimization. This approach is defined by using a least-squares loss function, $\mathcal{L}(y, F_m(x)) = \frac{(y-F)^2}{2}$. Due to the selection-optimization split in the case of Geometry based ensembles we can find the optimum analytically. By reversing the order of lines 3 and 4 of Algorithm 2, one can first find the optimal weighting value for each candidate classifier $h(x; a)$, $\frac{\partial((\hat{y}-\rho_a^T h(x;a))^T(\hat{y}_i-\rho_a^T h(x;a)))}{\partial \rho_a} = 0$ which leads to $\hat{y}^T h(x; a) = \rho_a^T h(x; a)^T h(x; a) = \rho_a^T M$, observe that since $h(x; a)$ is $\{+1, -1\}$, the dot product $h(x; a)^T h(x; a)$ is simply the number of training examples M. Once the optimal set of values is found, a simple selection of the classifier that best approximates the negative gradient is performed. In the current formulation parameter a represents a selection variable.

1 $F_0(x) = y$
for $t = 1$ *to* T*:* **do**
2 $\quad \hat{y}_i = y_i - F_{t-1}(x_i) \quad i = 1, M$
3 $\quad \rho = \frac{\hat{y}^T A}{M+\lambda}$ // ρ is a vector, one value per classifier
4 $\quad a_t = \arg\min_a \sum_{i=1}^{M} (\hat{y}_i - \rho_a h(x_i; a))^2$
5 $\quad F_t(x) = F_{t-1}(x) + \rho_t h(x; a_t)$
end

Algorithm 3. L2-penalized Least squares GE boost.

Direct optimization of ill-posed problems usually yield to poor results. In the original geometry-based ensembles paper, a penalized formulation was used. Following the same guidelines, we can perform the same optimization incrementally by formulating a penalized loss function in the same terms as Equation 7. However, as we will see in the experimental results section, in this incremental approach we can allow the change of polarity of a weak hypothesis without hindering the convergence of the process. Following the former case derivation, it is simple to obtain the new weighting vector $\rho = \frac{\hat{y}^T A}{M+\lambda}$. Algorithm 3 shows the regularized incremental geometry-based ensemble procedure. Observe that by setting λ to zero we obtain the non-regularized algorithm

4 Results

Data sets: Fifteen bi-class data sets from the UCI repository [3] have been used to assess the performance of the different methods. Table 1 shows the details of the data sets.

Table 1. UCI data sets used in the experiments with the number of examples and dimensionality d

Database	code	Examples	d	Database	code	Examples	d
Breast Winsc.	(bcw)	699	10	Statlog Heart	(shd)	270	13
Bupa liver	(bld)	345	6	Tic Tac Toe	(ttt)	958	9
Heart Clev.	(hec)	920	13	Spect	(spt)	267	22
Ionosphere	(ion)	351	34	New thyroid	(nth)	215	5
Breast	(bre)	569	32	Pima indians	(pid)	768	8
Voting records	(vot)	435	16	Sonar	(snr)	208	60
Monks complete	(mon)	432	7	Statlog Au Credit	(sac)	690	14
Credit	(cre)	690	15				

Methods: Four incremental geometry-based ensembles are used in the experiments: Adaptive Boosted Geometry-based ensemble (BGE) and Regularized Gradient Boosted Geometry-based ensembles (GBGE) with $\lambda = 0$, $\lambda = 100$ and $\lambda = 300$. These methods are compared with Adaboost with decision stumps, Support Vector Machines with Radial Basis Function kernel and the original proposal of the Geometry-based Ensembles (OGE).

Parameters Settings: The values for C and σ from the SVM-RBF approach and λ from the OGE are found using 5-fold cross validation on the training set.

Experimental Settings: Ten randomized runs of stratified ten-fold cross validation (for a total of 100 experiments/database) are performed and the mean accuracy is computed for ensembles ranging from 1 to 400 classifiers.

Figures 2 and 3 show the evolution of the test mean accuracy as the number of classifiers in the ensemble increase. Dashed lines correspond to reference methods: horizontal black dashed line corresponds to OGE, horizontal gray dashed line to SVM, and light gray dashed line to Adaboost. Dotted line shows the performance of the GE optimized using Adaboost (BGE). Solid lines show the evolution of the accuracy for the regularized approaches – the darker the line is, the higher the lambda value is set.

At first glance, one observes that the incremental methods suffer from overfitting – including Adaboost – in six of the fifteen data sets. Summarizing the plots, regularized versions of the incremental version of GE perform very well in eleven out of the fifteen data sets. The worst performances are found in (ion), (bre), (snr) and (vot). Observe that, in general, higher values of the regularization parameter delay overfitting as well as achieve better results than smaller values in most cases. It is worth commenting that the values of λ in the incremental versions are much bigger than in OGE where the optimal value ranged from 1 to 100, while in the regularized gradient boosted version it ranges from 100 to 500.

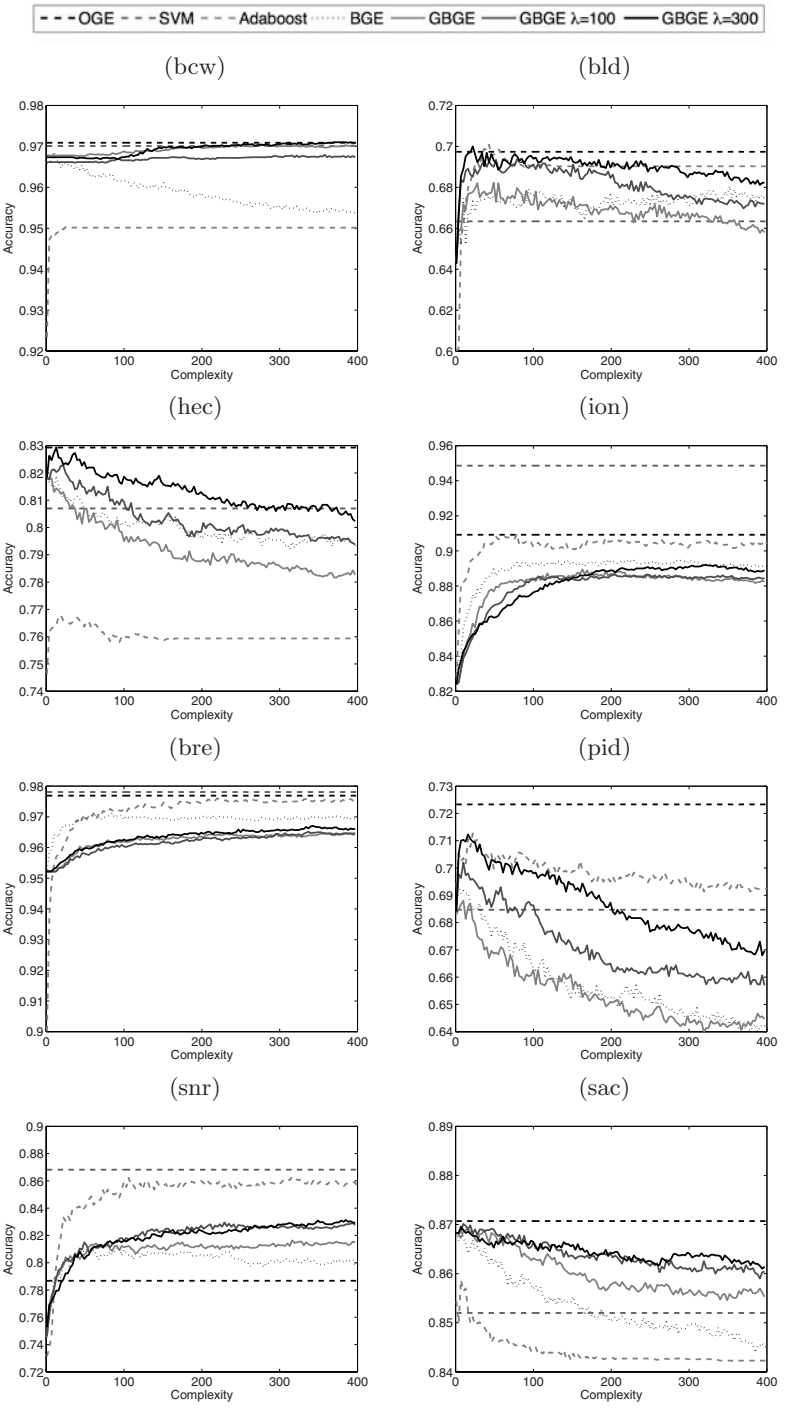


Fig. 2. Test accuracy results with respect to the number of classifiers in the ensemble

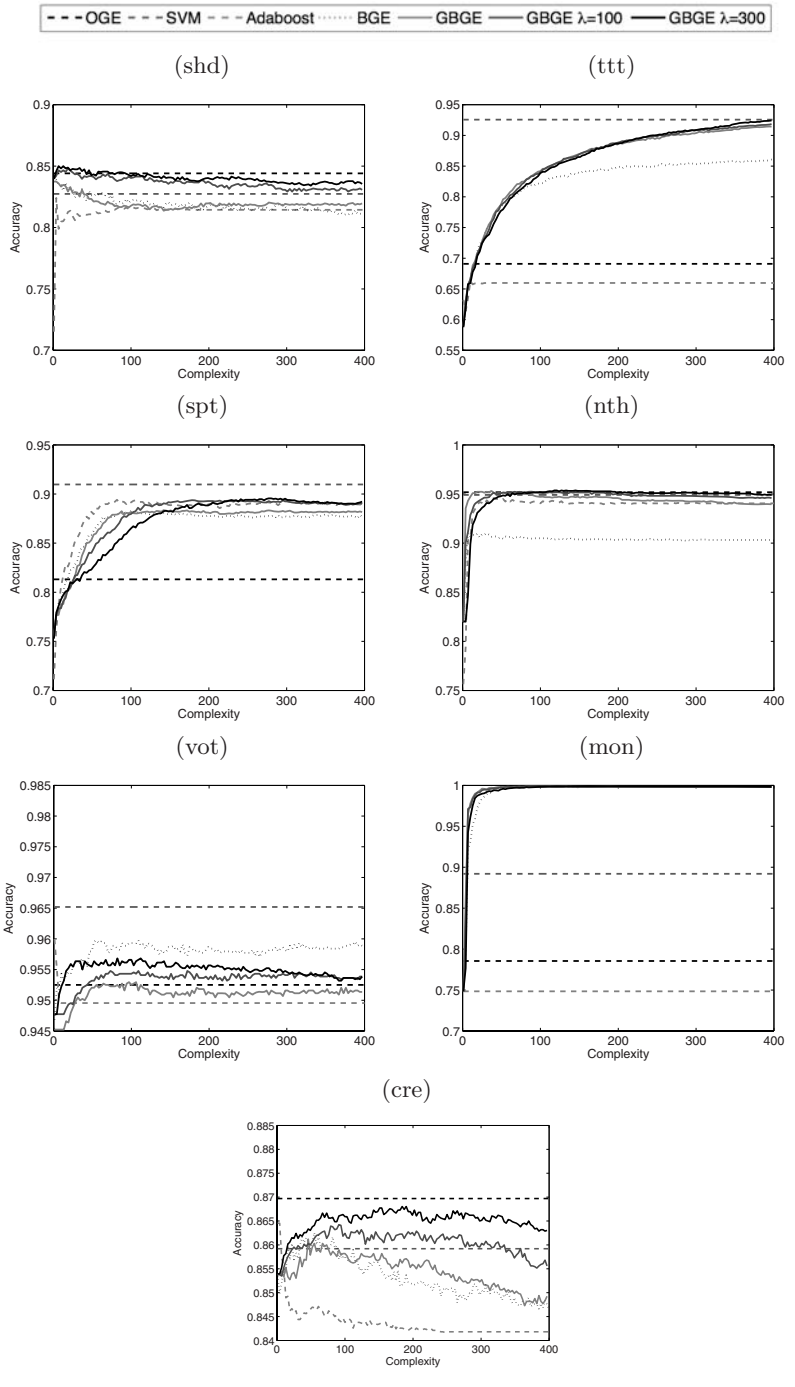


Fig. 3. Test accuracy results with respect to the number of classifiers in the ensemble

Table 2. Comparison among the different methods in terms of (Wins/Draws/Loses). One method is considered to win another if it displays a higher accuracy than the maximum obtained by another in at least 25% of the search space.

	Regularized	GBGE	BGE	Adaboost	SVM	OGE
Regularized	-	10/5/0	11/1/3	8/3/4	7/3/5	6/3/6
GBGE	0/5/10	-	5/6/4	7/2/6	2/4/9	4/1/10
BGE	3/1/11	4/6/5	-	6/2/7	3/0/12	5/0/10

In the following lines, the general performance of the methods with respect to the reference methods is quantitatively compared. The notation (wins/draws/loses) is used to show the approximate behavior of a method with respect to another. One method is considered to outperform another if it displays a higher accuracy than the maximum obtained by the other in at least 25% of the search space. Table 2 shows the values for the methods considered in the experiment. The regularized versions of the incremental version clearly outperfom the rest of the proposed methods. With respect to the reference classifiers, regularized versions are approximately on par with OGE and SVM, and improve Adaboost performance. GBGE and BGE are clearly outperformed by OGE and SVM, but are on par with Adaboost. There are no significant differences between BGE and GBGE.

5 Discussion and Conclusion

In this article, incremental methods for controlling complexity of geometry-based ensembles are explored. Adaboost and Least squares residual fitting are used as base methods for selecting and combining CBP-based classifiers. Results show that both methods, though on par with the performance of Adaboost with decision stumps, display a poor behavior when compared with SVM and OGE. Following the same idea than in the original geometry-based ensembles article, a L2-penalization term is added to the formulation of the last method. As a result, much better behavior with small complexity is obtained, being on par with SVM and OGE. This result verifies the initial hypothesis of the article regarding the validity of CBP-based classifiers ensembles with small complexity.

One interesting point that must be observed in the formulation of the current methods is that incremental geometry-based ensembles are based on a predefined set of classifiers, and the creation of the ensemble can be regarded as a forward selection of the best classifier at a given time and the optimization of its weight in the ensemble. Note that the creation of the set of classifiers and the incremental optimization are decoupled, making the algorithm potentially useful for online and parallel extensions.

Finally, observe that on few data sets the performance of the strategy is clearly inferior to the rest of the methods. I conjecture that this is due to a lack of classifiers for the selection process. This fact would also explain the overfitting

tendency in some data sets. As a future research it may be useful to explore this effect and ways to overcome it. One possible way to address this problem is to change the notion of CBP. Up to this moment, the creation of the CBP rely on the Gabriel graph. The Gabriel graph considers edges between points as long as there is no other one inside the hypersphere with diameter the distance between the creating points. However, this notion can be easily generalized if we allow up to k points to be inside the hypersphere. This change effectively increase the number of CBPs and define an incremental space of classifiers.

References

1. Pujol, O., Masip, D.: Geometry-Based Ensembles: Toward a Structural Characterization of the Classification Boundary. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(6), 1140–1146 (2009)
2. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics 29(5), 1189–1232 (2001)
3. Murphy, P.M., Aha, D.W.: UCI Repository of machine learning databases. University of California, Dept. of Information and Computer Science, Irvine (1994)
4. Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics 28(2), 337–407 (2000)
5. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proc. XIIIth ICML, pp. 148–156 (1996)
6. Bottou, L., Bousquet, O.: The Tradeoffs of Large Scale Learning. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S. (eds.) NIPS. NIPS Foundation, vol. 20, pp. 161–168 (2008), http://books.nips.cc
7. Kuncheva, L.: Classifier ensembles for changing environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)

Online Non-stationary Boosting

Adam Pocock, Paraskevas Yiapanis, Jeremy Singer,
Mikel Luján, and Gavin Brown

School of Computer Science, University of Manchester, UK
{apocock,pyiapanis,jsinger,mlujan,gbrown}@cs.manchester.ac.uk

Abstract. Oza's Online Boosting algorithm provides a version of AdaBoost which can be trained in an online way for stationary problems. One perspective is that this enables the power of the boosting framework to be applied to datasets which are too large to fit into memory. The online boosting algorithm assumes the data distribution to be independent and identically distributed (i.i.d.) and therefore has no provision for concept drift. We present an algorithm called Online Non-Stationary Boosting (ONSBoost) that, like Online Boosting, uses a static ensemble size without generating new members each time new examples are presented, and also adapts to a changing data distribution. We evaluate the new algorithm against Online Boosting, using the STAGGER dataset and three challenging datasets derived from a learning problem inside a parallelising virtual machine. We find that the new algorithm provides equivalent performance on the STAGGER dataset and an improvement of up to 3% on the parallelisation datasets.

1 Introduction

Many real-world problems change their characteristics over time. This is known as learning in non-stationary environments, where the function which maps from the inputs to the outputs changes over time [4,5,7,10]. This change is known as *concept drift*. Ensemble frameworks provide a modular style where components can be added, removed or modified which is particularly useful when tracking changing concepts. The aim of this work is to demonstrate a novel online learning algorithm, capable of being deployed in a *real-time* environment, requiring online learning with thousands of examples per second. The domain we work with is that of *adaptive Java compilers*, which dynamically optimise how Java code is executed on modern *multi-core* computers.

We are investigating the application of ML techniques to *automatic parallelisation* problems, running inside a Java virtual machine. Automatic parallelisation problems have several interesting characteristics that constrain the space of applicable learning algorithms. Training data is taken from runtime analysis of Java benchmarks, recording properties of the machine state at the start/end of each method execution. The prediction task is, given the machine state, and the source code for a new Java method, *should this method be executed in parallel?* This is an important task, as parallel execution can potentially fail if it causes a

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 205–214, 2010.

resource conflict with other currently running methods; and solving this problem well can potentially halve the runtime of a piece of Java code.

Since decisions are taken at the start of every method execution, this results in a dataset generating potentially millions of examples *per second*, rendering many standard offline training algorithms infeasible. The exemplar of this is our DaCapo Bloat data [1], which generates 1,273,359 datapoints in 386 milliseconds of runtime. Predictions required by the parallelisation system therefore need to be generated on the order of microseconds otherwise the parallelisation opportunity is lost in the overhead created by the ML system. Additionally, a training cycle must not 'lock' the ML system preventing it from providing a prediction. A further problem is that the data is not independent and identically distributed, as decisions made by the parallelisation system using ML predictions will alter the future behaviour of the system. In summary we require a ML system which:

- Learns in a 'true' online fashion[1].
- Can adapt to a changing distribution.
- Can generate predictions quickly.
- Can be updated without preventing the generation of predictions.

We propose a system based Oza's Online Boosting [8], with modifications to enable the updating of ensemble members, adapting the system to changing data. Online Boosting meets the requirements using simple base models which can generate predictions quickly, and while each ensemble member is being trained, the remainder of the ensemble members could be used to generate a prediction.

The layout of this paper is as follows: Section 2 provides a description of the related literature. Section 3 provides a description of our algorithm and how it relates to the literature. Section 4 provides our experimental setup and testing results, and Section 5 contains the conclusions and future directions for our algorithm.

2 Related Work

Learn^{++}.NSE [2,7] provides an algorithm for generating a boosted classifier on streaming data. It generates a series of classifiers using batches of examples, by converting the online datastream into a series of chunks of a fixed size. At each time step one new classifier is trained on a batch of new examples, using an example weighting distribution similar to AdaBoost based upon the performance of the current ensemble, then all the ensemble members are reweighted according to their performance on the current batch of examples. After each classifier has been trained it becomes immutable, though its weight in the majority vote may change dependent on its current performance.

Knowledge-based Sampling Stream (KBS-Stream) [10] is a boosting algorithm similar to Learn^{++}.NSE, as it generates a series of classifiers by creating batches

[1] By 'true' online, we mean a system which learns from and then discards each training example one-by-one.

of examples from an online datastream. A key difference from Learn^{++} is the classifier weighting scheme, based on a probabilistic correlation measure. Another important difference is the way it makes use of new data: in Learn^{++}, a new classifier trained every K examples, whereas in KBS-Stream, a new classifier is only trained if the data distribution is deemed to have drifted.

It is important to note that both Learn^{++}.NSE and KBS-Stream assume that data in each batch are independent and identically distributed. In our parallelisation problem the distribution is unknown, and thus no such guarantee can be made. This leads to a problem if we were to apply these algorithms to the parallelisation datasets described in Section 4.1, as each new example would hypothetically require the generation of a new classifier. In contrast to such 'batching' algorithms are algorithms which update the classifiers on presentation of each example, which we term 'true' online learning algorithms.

Online boosting developed by Oza [8] provides a boosting algorithm which mimics the sampling with replacement variant of AdaBoost [3] when applied to an online stream of examples. Each example is presented to an ensemble member r times, where r is drawn from a Poisson(λ) where λ is a weight derived from previous performance on that example. Effectively, if the example is misclassified by an earlier classifier it is presented more often to classifiers later on in the ensemble. This assumes that the ensemble members are capable of learning incrementally and that repeated training on the same example will have a cumulative effect on the classifier. The ensemble is initialised with a fixed number of members which remain throughout the training process. The algorithm is proven to return the same ensemble as AdaBoost in the limit of infinite examples, when using a Naive Bayes classifier as the weak learning algorithm. At each stage the algorithm is approximating the performance of AdaBoost trained upon the same examples, which limits the performance when the distribution is changing because offline classification tasks are not subject to concept drift.

AdaBoost performs a greedy forward search in the space of classifiers. Alternative search methods which are less greedy can be fitted in the place of this forward search. One such algorithm is FloatBoost [6] which incorporates a sequential forward floating search [9] in place of the greedy forward search. This enables the removal of classifiers which are hindering the performance of the ensemble or are made redundant by a combination of other ensemble members. Like AdaBoost, it is an offline binary classification algorithm. The sequential forward floating search increases the runtime of the algorithm significantly whilst providing an increase in the accuracy.

The properties of the various online algorithms described in this section are summarised in Table 1.

3 Non-stationary Boosting

We present an algorithm that, like Online Boosting, uses a static ensemble size without generating new members each time new examples are presented, and also adapts to a changing data distribution. It is based upon a combination of

Table 1. Comparison of related algorithms and ONSBoost. By "True Online" we mean that the algorithm can be trained on single examples, without chunking them and assuming i.i.d. within the batch.

Algorithm	"True" Online	Fixed Ensemble Size	Concept Drift
Learn^{++}.NSE	✗	✗	✔
KBS-Stream	✗	✗	✔
Online Boosting	✔	✔	✗
ONSBoost	✔	✔	✔

FloatBoost [6] and Online Boosting [8] which we call Online Non-Stationary Boosting, or ONSBoost. It incorporates a floating search into the Online Boosting algorithm, which enables the addition of new classifiers and the removal of poorly performing classifiers. This lets the algorithm follow a changing data distribution.

The algorithm, ONSBoost, provides a simple extension of Online Boosting to allow the resetting of outdated and inaccurate classifiers. The key parameters are: K, determining how often the classifiers are checked to see if a reset is necessary; W, the size of the window used to determine if a reset is necessary; and P, a "protection" period, where a classifier cannot be reset. The protection period is necessary to allow a classifier sufficient training examples to learn the new concept.

The algorithm reduces to Oza's Online Boosting in the case when K is greater than the number of examples in the training dataset[2]. When K is less than the number of examples N there are $\lfloor \frac{N}{K} \rfloor$ classifier removal steps.

Window Size

In a true online environment the base assumption is that there is an infinite stream of examples being generated for training and classification. This means it is not feasible to store all the previous examples and use those to determine performance. A common technique for evaluating online classifiers is to use a window of the most recent examples to determine the current performance [4]. This can also improve the accuracy on data with concept drift as it forces the classifier to adapt to the most recent data. In common with these other techniques we use a window to determine the current performance of the ensemble and to decide which, if any, members need replacing to improve the ensemble.

Update Period

A parameter is introduced to control how often a search is performed of the classifiers to check if the ensemble performance is being hindered. Ideally this step would be performed after each example has been presented for training, however this introduces problems. The backwards search requires $n \times j$ evaluations, where n is the window size and j is the number of classifiers which makes it very

[2] Note that in our current implementation we have used a modified pseudo-count method for error estimates.

computationally intensive compared to training. Thus performing a search after each example would greatly increase the time complexity of each training cycle.

This parameter effectively controls how quickly poorly performing classifiers are replaced, and thus has an effect on the speed with which the algorithm adapts to concept drift. The searches are then limited so they are only performed after the algorithm has seen K new examples. For example, if the parameter is set to 50, then each time 50 examples have been processed then a backwards search for poorly performing classifiers will be executed.

Protecting New Classifiers

One further parameter is added to the system, which is necessitated by the fact that newly created classifiers will perform poorly until they have been trained on a sufficient number of examples. They are "protected" from removal by the backwards search until they have been sufficiently trained. This is not a problem in an offline algorithm as each new classifier is trained on the whole dataset. It is parameterised because the amount of time a classifier needs to train to a given standard on a dataset is in general unknown.

How to place classifiers?

A subtle point in all boosting algorithms, but particularly in Oza's Online Boosting is that the classifier ordering matters whilst the system is being trained. In Online Boosting each classifier is implicitly dependent on the output of all classifiers before it, as they are trained in a fixed sequential order. The performance of a previous classifier dictates the number of times that the current classifier is trained on any given example. To modify Online Boosting to allow for the replacement of classifiers thought needs to be given to the placement of the new classifier.

The current algorithm places the reset classifier at the end of the ensemble, and the alternative is in place replacement, where the reset classifier directly replaces the old one. The former makes the $(i + 1)$th classifier adapt to the distribution of examples that the ith classifier was learning when the ith classifier is replaced. In the latter the new ith classifier will initially have poor performance and thus make all examples have higher weights for any classifiers which are sequentially after the ith classifier. The choice between these two behaviours is interesting, but not explored in this paper. The two different methods are shown in Figure 1.

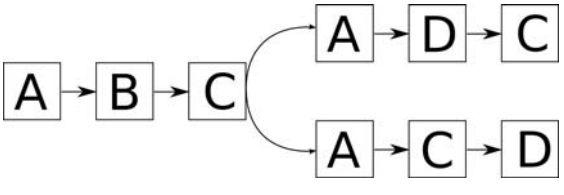


Fig. 1. Classifier placement. If classifier B is replaced with a fresh classifier D, D can be put in two different places.

Algorithm 1. ONSBoost

Variables: H = the ensemble, h is an ensemble member, $\epsilon(H)$ is the ensemble error on the window
Parameters: M = number of classifiers, K = update period, W = window size P = number of examples to "protect" a new classifier
Initialisation: $\forall m \in \{1, 2, ..., M\}, \lambda_m^{sc} = 0, \lambda_m^{sw} = 0, k = 0$

ONSBoost (H, train,(x, y))
PHASE I: Oza's Online Boosting
Set the current example's "weight" $\lambda = 1$
Increment the example counter $k = k + 1$
for all $h_m, (m \in \{1, 2, ..., M\})$ in H **do**
  Set r sampled from $Poisson(\lambda)$
  for $i = 0, i < r$ **do**
    $h_m \leftarrow \text{train}(h_m, (x, y))$
  end for
  if $y = h_m(x)$ **then**
    $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda$
    $\epsilon_m \leftarrow \frac{\lambda_m^{sw}+1}{\lambda_m^{sc}+\lambda_m^{sw}+1}$
    $\lambda \leftarrow \lambda(\frac{1}{2(1-\epsilon_m)})$
  else
    $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda$
    $\epsilon_m \leftarrow \frac{\lambda_m^{sw}+1}{\lambda_m^{sc}+\lambda_m^{sw}+1}$
    $\lambda \leftarrow \lambda(\frac{1}{2\epsilon_m})$
  end if
end for

PHASE II: ONSBoost modification
Move window along datastream
if $k = K$ **then**
  H_P = all classifiers which have trained on at least P examples
  $h' = \arg\max_{h \in H_P} \epsilon(H - h)$
  if $\epsilon(H - h') < \epsilon(H)$ **then**
    Remove h' from H, and add a new h to the end of H
    Set $\lambda_h^{sc} = 0, \lambda_h^{sw} = 0, \epsilon_h = 0$
  end if
  $k \leftarrow 0$
end if

return $H(x) \leftarrow \arg\max_{c \in Y} \sum_{m:h_m(x)=c} \log \frac{1-\epsilon_m}{\epsilon_m}$

4 Experimental Results

An obvious question is "how should we fix the parameters K, W and P?" In this section we empirically evaluate the proposed algorithm with particular emphasis on characterising the parameter space.

4.1 Datasets

We use 4 datasets in the presented experiments, STAGGER and 3 datasets taken from an automatic parallelisation problem which have concept drift and thousands or millions of examples.

The parallelisation datasets all use a set of features derived from an offline inspection of Java bytecode. The Java applications are taken from the DaCapo benchmarks suite [1]. The datasets are described in detail in [11]. Each dataset comprises a set of method features, and if the method was successfully executed in parallel with its parent method. The class concept is subject to concept drift as the underlying virtual machine state is not fully captured in the feature set. This is a *hidden context* [4] which can be subject to gradual, cyclical, and abrupt forms of concept drift. Abrupt and cyclical behaviour can be generated as the feature set does not include information on all currently executing methods, and these can cause conflicts which cause the parallel execution to fail. Gradual behaviour can be generated as variables in the application change over time and cause different memory access patterns, which affect the parallel execution.

The STAGGER dataset is described in [12]. A standard methodology with this dataset is to generate 100 test examples sampled from the current distribution which are used to test the performance of any given algorithm. This does not reflect a true online learning scenario as there is not generally an opportunity to sample a testing set from the same distribution As a result dataset size was increased to 600 to improve the measure of accuracy, with 1/3 still taken from each of the three concepts. The format and features of the datasets are described in Table 2.

Table 2. Dataset Properties

Dataset	Examples	Features	Feature Type	Class Skew
bloat	1,273,359	38	Binary	7% +ve
antlr	169,578	38	Binary	22% +ve
pmd	38,994	38	Binary	48% +ve
STAGGER	600	3	Ternary	29% +ve

4.2 Comparison with Online Boosting

Due to the problems Learn^{++}.NSE and KBS-Stream have with data which is not i.i.d. at the example level, we compare the new algorithm with Online Boosting (without priming). In all cases the base learner used is a categorical Naive Bayes with pseudocounts. The number of classifiers was kept constant at 30 when using both ONSBoost and Online Boosting. Each experiment was repeated 10 times, to eliminate some of the randomness inherent in the Poisson distribution used in both algorithms. Results on STAGGER are presented, with additional results using datasets collected by our automatic parallelisation

Table 3. Online accuracy comparison between ONSBoosting and Online Boosting

Dataset	Online Boosting	ONSBoost	ONSBoost Parameters
antlr	80.89% ± 0.23	**81.92% ± 0.05**	$K = 200, W = 50$
bloat	89.32% ± 0.85	**91.93% ± 0.03**	$K = 200, W = 50$
pmd	76.19% ± 0.18	**78.10% ± 0.18**	$K = 200, W = 50$
STAGGER	95.97% ± 0.36	**96.10% ± 0.36**	$K = 10, W = 10$

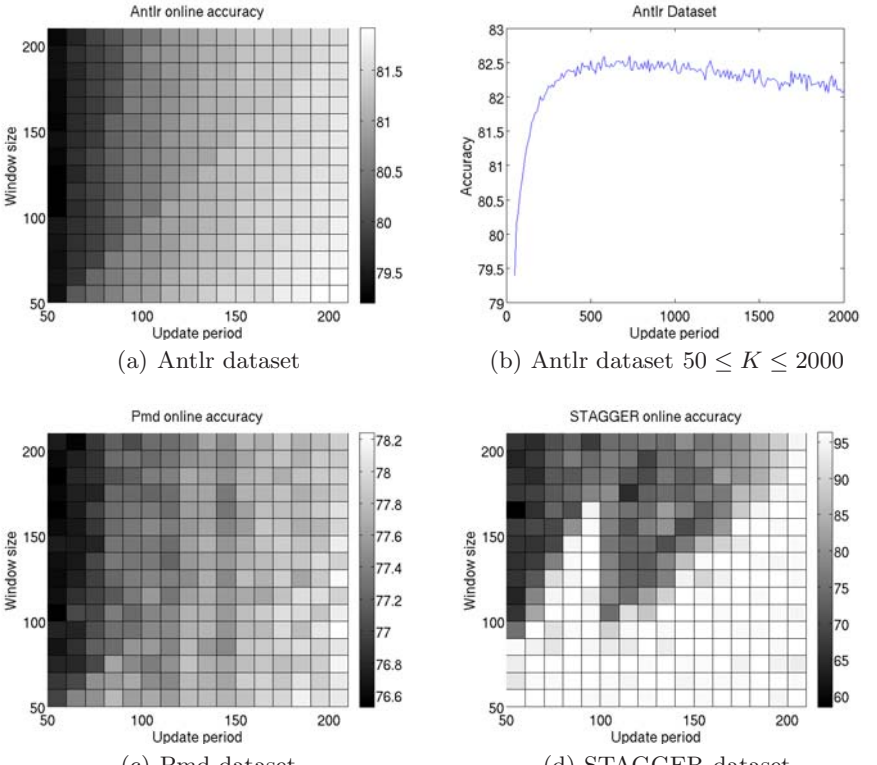
framework. Each experiment tests the online accuracy of a classifier. Online accuracy is the percentage of the training data that the classifier predicts correctly, before it has been trained on that example. The value are given with 95% confidence intervals.

The parameters for the comparison with Online Boosting were chosen as these give the best result across the most data. This can be seen in the parameter exploration in Section 4.3. With the STAGGER dataset an Update Period of 200 examples means that the algorithm considers classifiers for removal based upon a concept which changes with a hard boundary at the example immediately afterwards. For this reason a small update period was chosen, with a consequently smaller window size. In these experiments we perform better than Online Boosting in the parallelisation datasets, and equivalently in the STAGGER dataset. Even small improvements in accuracy are important for our task, as each incorrect prediction has associated costs, and a 1% improvement in accuracy results in an extra 12,000 correctly classified examples in the bloat dataset.

4.3 Exploration of Parameter Space

We now vary the parameters of ONSBoost to see how sensitive the performance is to parameter choice. Results are presented in Figure 2 varying the update period (K) and window size (W) parameters of ONSBoost. The number of examples to protect a new classifier (P) was fixed at 100 for all experiments. Figure 2(b) shows the accuracy for one line of the heat maps, fixing the window size to 50, and varying the update period from 50 to 2000 in steps of 10. This shows how the performance increases as this parameter increases before decreasing again once past a data dependent threshold, and the performance will converge to the performance of Online Boosting when the update period is equal to the size of the dataset.

From the exploration of the parameter space it appears that small window sizes and a relatively high number of steps before a backward search are the best parameters, as these consistently provide the highest accuracies. Using a window size which is much greater than the number of steps before a backwards search causes a decrease in performance, as new classifiers are penalised by the window as they have not been trained upon the examples it contains.



(a) Antlr dataset (b) Antlr dataset $50 \leq K \leq 2000$

(c) Pmd dataset (d) STAGGER dataset

Fig. 2. Online accuracy results for ONSBoosting

5 Conclusion and Future Work

We presented an algorithm capable of dealing with continuous concept drift in a resource constrained environment over millions of examples. The algorithm is motivated by the need to develop a fast online learning algorithm for an automatic parallelisation problem, which imposes constraints on the types of algorithms which can be used. The algorithm replaces classifiers based upon their impact on the ensemble's performance rather than simply removing the oldest classifier in the ensemble, so it can cope with some measure of cyclic behaviour in the concept drift. We compared this new algorithm to Online Boosting, and found an improvement in performance on our automatic parallelisation problem, and comparable performance on a standard problem. The key idea is to maintain a fixed number of classifiers which are updated online, and to replace classifiers if they are negatively impacting the performance of the ensemble.

Our algorithm is based upon Online Boosting but will not converge to offline AdaBoost given the limit of infinite training examples. This is because the data is assumed to be i.i.d. in AdaBoost. Concept drift data is not i.i.d. and thus Online Boosting is a sub-optimal choice of learning algorithm in a concept drift

environment. ONSBoost provides a way to cope with concept drift in streaming data while maintaining the useful properties of Online Boosting, namely the ability to deal with incremental learning of streaming data, and the fixed number of classifiers, and thus fixed memory usage.

An area of further research is to develop a system to enable ONSBoost to cope better with cyclical drift. Learn^{++}.NSE can turn off classifiers based upon their current performance, but keep them for reuse later. A way of mimicking this ability in the ONSBoost system would be to select new ensemble members from a pool which included all previously removed classifiers, and a new ensemble member with no prior knowledge of the system. Other possible extensions include using a variable number of classifiers to decrease the ensemble size when the data is simple to classify and to increase the ensemble size when the data is difficult to classify.

References

1. Blackburn, S.M., et al.: The DaCapo benchmarks: Java benchmarking development and analysis. In: OOPSLA 2006: Proc. of the 21st annual ACM SIGPLAN conference on Object-Oriented Programing, Systems, Languages, and Applications, pp. 169–190. ACM Press, New York (2006)
2. Elwell, R., Polikar, R.: Incremental Learning of Variable Rate Concept Drift. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 142–151. Springer, Heidelberg (2009)
3. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proc. of the Thirteenth International Conference on Machine Learning, pp. 148–156 (1996)
4. Kuncheva, L.: Classifier Ensembles for Changing Environments. In: Roli, F., Kittler, J., Windeatt, T. (eds.) MCS 2004. LNCS, vol. 3077, pp. 1–15. Springer, Heidelberg (2004)
5. Kuncheva, L.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives. In: Proc. of the 2nd Workshop SUEMA 2008, ECAI 2008 (2008)
6. Li, S., Zhang, Z., Shum, H., Zhang, H.: FloatBoost learning for classification. In: Advances in Neural Information Processing Systems, pp. 1017–1024 (2003)
7. Muhlbaier, M., Polikar, R.: An ensemble approach for incremental learning in nonstationary environments. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 490–500. Springer, Heidelberg (2007)
8. Oza, N.C.: Online Ensemble Learning. PhD thesis, The University of California, Berkeley, CA (September 2001)
9. Pudil, P., Novoviová, J., Kittler, J.: Floating search methods in feature selection. Pattern recognition letters 15, 1119–1125 (1994)
10. Scholz, M., Klinkenberg, R.: Boosting classifiers for drifting concepts. Intelligent Data Analysis 11, 3–28 (2007)
11. Singer, J., Pocock, A., Yiapanis, P., Brown, G., Luján, M.: Fundamental Nano-Patterns to Characterize and Classify Java Methods. In: Proc. Workshop on Language Descriptions, Tools and Applications (2009)
12. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine learning 23, 69–101 (1996)

Combining Neural Networks to Improve Performance of Handwritten Keyword Spotting

Volkmar Frinken, Andreas Fischer, and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{frinken,afischer,bunke}@iam.unibe.ch

Abstract. Keyword spotting refers to the process of retrieving all instances of a given word in a document. It has received significant amounts of attention recently as an attractive alternative to full text transcription, and is particularly suited for tasks such as document searching and browsing. In the present paper we propose a combination of several keyword spotting systems for unconstrained handwritten text. The individual systems are based on a novel type of neural network. Due to their random initialization, a great variety in performance is observed among the neural networks. We demonstrate that by using a combination of several networks the best individual system can be outperformed.

1 Introduction

The automatic recognition of handwritten text – such as letters, manuscripts or entire books – has been a focus of intensive research for several decades [1,2]. Yet the problem is far from being solved. Particularly in the field of unconstrained, writer independent handwriting recognition where the writing styles of various writers must be dealt with, severe difficulties are encountered.

Making handwritten texts available for searching and browsing is of tremendous value. For example, one might be interested in finding all occurrences of the word "complain" in the letters sent to a company. As another example, libraries all over the world store huge numbers of handwritten books that are of crucial importance for preserving the world's cultural heritage. Making these books available for searching and browsing would greatly help researchers and the public alike. Finally, it is worth mentioning that Google and Yahoo have announced to make handwritten books accessible through their search engines [3].

Transcribing the entire text of a handwritten document for searching is not only inefficient as far as computational costs are concerned, but it may also result in poor performance, since misrecognized words cannot be found. Therefore, techniques especially designed for the task of keyword spotting have been developed.

Current approaches to word spotting can be split into two categories, viz. query-by-example (QBE) and query-by string (QBS). With the former approach, all instances of the search word in the training set are compared with all word images in the test set. Among the most popular approaches in this category

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 215–224, 2010.

are dynamic time warping (DTW) [4,5,6] and classification using global features [7,8]. Word shape methods using Gradient, Structural and Concavity features (GSC) have been shown to outperform DTW in [9,10]. Algorithms based on QBE suffer from the drawback that they can only find words appearing in the training set. The latter approach of QBS models the key words according to single characters in the training set and searches for sequences of these characters in the test set [11,12]. Recently, keyword spotting systems that are modified versions of handwriting recognition systems have received increasing attention. In [11,13,14], hidden Markov models are used to find the words to be searched.

To the knowledge of the authors, only single stand-alone keyword spotting systems have been proposed, but no attempt for a combination has been published yet. In this paper we propose to combine several systems using well-known multiple classifier combination techniques. We demonstrate that combining several keyword spotting systems is a convenient method to substantially increase the performance.

In [15] a novel neural network based keyword spotting system is proposed that performs very well. However, a potential problem with this system is the fact that different neural networks vary in their performance due to their random initialization. But an ensemble of well performing keyword spotting systems can be created easily. This renders this technique ideal as a basis to investigate the combination of keyword spotting systems.

The rest of the paper is structured as follows. In Section 2, the underlying neural network based keyword spotting system is introduced. In Section 3, we present the combination methods used for experimental evaluation. The experiments and their results are given in Section 4 and conclusions are drawn in Section 5.

2 The Base Keyword Spotting System

Keyword spotting refers to the process of retrieving all instances of a given word in a document. In this paper, we focus on handwritten documents, such as letters, memos, or manuscripts. Without transcribing the data, a user should still be able to search for any possible word, just like using a search engine. How the results of such a search may look like can be seen in Fig. 1. Note that the base system just returns a likelihood of the word being found. Afterwards, this likelihood can be compared to a threshold to decide whether or not this is a true match.

2.1 Preprocessing

We consider complete text lines as input units for our keyword spotting system. The texts used in the experiments come from the IAM database [16]. They are extracted from pages of handwritten texts, which were scanned and segmented into individual text lines. After binarizing the image with a threshold on the grey scale value, the slant and skew of each textline are corrected and the width and

(a) Returned log Likelihood: -1.7125

(b) Returned log Likelihood: -8.4097

(c) Returned log Likelihood: -11.0900

(d) Returned log Likelihood: -11.6082

Fig. 1. Search results for the word "found"

height are normalized. Then features are extracted using a horizontally sliding window. A window with a width of one pixel is used to extract nine geometric features at each position, three global and six local ones. The global features are the 0^{th}, 1^{st} and 2^{nd} moment of the black pixels' distribution within the window. The local features are the position of the top-most and that of the bottom-most black pixel, the inclination of the top and bottom contour of the word at the actual window position, the number of vertical black/white transitions, and the average grey scale value between the top-most and bottom-most black pixel. For details on these steps, we refer to [17].

2.2 BLSTM Neural Networks

The recognizer used in this paper is a recently developed recurrent neural network, termed *bidirectional long short-term memory* (BLSTM) neural network [18]. Instead of simple nodes, the hidden layers are made up of so-called *long short-term memory* blocks, specifically designed to address the the problem of exponential increase or decay of information in recurrent neural networks.

The output layer contains one node for each possible character in the sequence plus a special ε node, to indicate "no character". At each position, the output activations of the nodes are normalized so that they sum up to 1, and are treated as a probability vector for each letter at this position. For more details about BLSTM networks, we refer to [18,19].

The sequence of probability vectors returned by the neural network can be efficiently used for word and text line recognition as well as for word spotting [15], where the Connectionist Temporal Classification (CTC) Token Passing algorithm [18] is utilized for the latter task. In short, the probability sequence is extended by an additional entry representing an *any* character ($'*'$) and having always the value 1. By adding a symbol, representing the *any* character, to the

beginning and to the end of the word w to be spotted, the CTC algorithm finds the best path that passes through the *any* character, then through the word w and then again through the *any* character. This means that the path traverses through the letters of the word w where it fits best while the rest of the text line has no influence. Then, the product of all probability values along this path is computed and divided by the keyword's length (the number of letters in the word). The result can be interpreted as the likelihood that this word is contained in the considered text line. For more details about the keyword spotting algorithm we refer to [15].

3 Combination Methods

The final decision whether or not a word appears in a given textline is made by comparing the likelihood returned by the neural network to a global threshold. In this paper, we propose to first combine all likelihoods from the different systems and then compare the resulting value to a threshold. This way, the combination method can make use of information returned by all individual systems. Additionally, it is possible to combine all kind of recognition methods, as long as they return a likelihood value.

We investigated five different combination methods, *Min*, *Max*, *Average*, *Median*, and *Product* [20]. The *Min* combination method returns the minimum of all values and reflects thereby a group of experts that always listen to their most skeptical member. The *Max* combination method returns the maximum value and acts like experts that are as confident as their most confident member. The *Average* combination methods returns the average of the different values. Under this combination rule, every expert is listened to equally well. The *Median* combination method returns the median of the values and is therefore insensitive to outliers. Finally, the *Product* combination method returns the product of the values. Due to the nature of multiplication and the range of the values (between 0 and 1), this combination method is more biased towards outliers having a low value. In the group of experts analogy this would represent a group where skeptical experts have a higher impact on the final decision than confident experts.

Finally, we also implemented an *Oracle* combination method that returns the *Min* value if the word is not contained in the text line and the *Max* value if the word is contained. It serves as a theoretical upper limit on what could be achieved given the values of the underlying systems. We further investigated the diversity of the ensemble by means of the *correlation* coefficient. To formally describe this measures, consider two systems S_1 and S_2 and the four possible word spotting outputs: (a) both systems are correct, (b) S_1 is wrong and S_2 is correct, (c) S_1 is correct and S_2 is wrong, and (d) both S_1 and S_2 are wrong. With these values, *correlation* can be defined:

$$\text{Correlation}_{S_1,S_2} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}$$

Note that *correlation* is a pairwise measure, i.e. it can be applied to two systems only. To extend it to ensembles consisting of more than two members, the mean of all pairwise values is taken.

4 Experimental Evaluation

4.1 Setup

For testing the proposed keyword spotting method, we used the IAM offline database[1]. This database consists of 1,539 pages of handwritten English text, written by 657 writers. From this database, we used 6,161 lines as a training set, 920 lines as a writer independent validation set, and an additional 920 lines as a test set. Using the training set, we trained 50 randomly initialized neural networks and used the validation set to stop the back propagation iterations in the training process; see [18] for details on the neural network training algorithm. Then we selected the all 3,421 non-stop words[2] among the 4,000 most frequent words from all three sets and performed keyword spotting using these words. (Note that by far not all keywords occur in every set.)

The applied threshold of a system regulates it sensitivity. With a lower threshold, the results are noisier but the system misses less true positives. For various thresholds, we computed the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP), and *false negatives* (FN). These number were then used to plot a *precision-recall* scatter plot for each neural network. *Precision* is defined as number of relevant objects found by the algorithm divided by the number of all objects found $\frac{TP}{TP+FP}$, and *recall* is defined as the number of relevant objects found divided by the number of all relevant objects in the test set $\frac{TP}{TP+FN}$. A precision-recall plot therefore gives us an idea about the noise in the returned results, given the percentage of how many true elements are found. We considered for each system every single likelihood it returned as a possible threshold. Note that such a threshold is used as a global threshold to accept or reject lines (threshold based). Due to the high number of tested thresholds, about 3.1 millions, the points in the scatter plot can be considered as continuous curves.

Another evaluation commonly used for information retrieval tasks are *precision-recall* values considering the first n ranks of all queries[3] (rank based). Thus keywords that don't appear in the test set lower the precision value, while the systems ability to reject all lines when no keyword is found as it would be using a global threshold is not taken into account. Therefore, this is not an optimal measure for our task, but we report the averaged interpolated precision values for the sake of completeness.

[1] `http://www.iam.unibe.ch/fki/databases/iam-handwriting-database`

[2] We used the stop word list from the SMART project [21], which can be found at `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop`

[3] As done using the *trec_eval* program, `http://trec.nist.gov/trec_eval/`

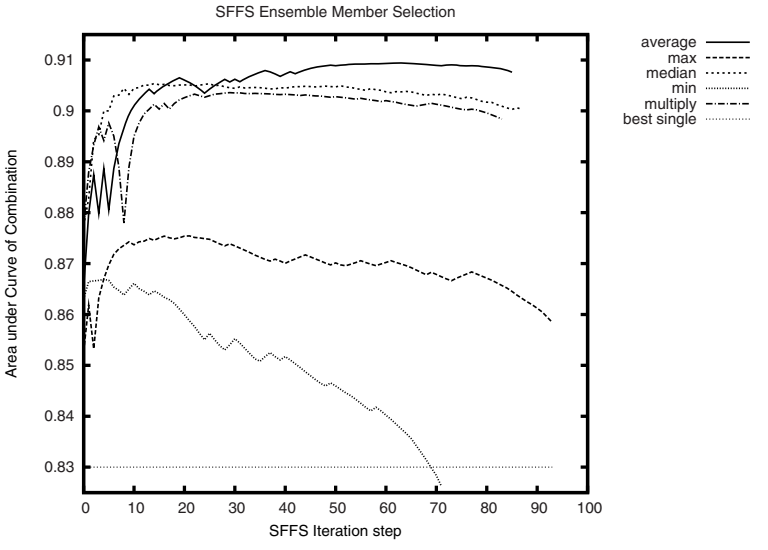


Fig. 2. The performance of the combination methods on the validation set during the SFFS iterations. The selected ensemble sizes are *average*: 33, *max*: 13, *median*: 14, *min*: 4, and *multiply*: 15.

There exist several ways to compare two curves. The most popular ones are the precision at 50% recall, the point where precision and recall values are equal, and the average precision over all recall values, which is the area under the curve. We chose to consider the average precision over all recall values since it includes information about the entire curve. Nevertheless, the improvement resulting from combining several classifiers can be observed with any of these measures.

4.2 Results

The network combination has been optimized using the sequential floating forward search (SFFS) [22] on the validation set to select the ensemble members. The fitness function used in the SFFS algorithm is the average precision of the *precision-recall* plot. In Fig. 2 the performance on the validation set can be seen for all presented combination methods during the course of the SFFS iterations. The average precision of the best single network can be seen as the dotted line at 0.83. Clearly, all combination techniques can outperform the best single network on the validation set.

The ensembles that performed best on the validation set according to the SFFS algorithm, using different combination rules, were then compared on the test set with each other as well as with the network that performed best on the validation set and the one that performed best on the test set. The following table lists the average precision of all mentioned systems.

system	av. precision (threshold based)	interpolated av. precision (rank based)
Ø single systems	0.7576	0.2926
system performing best on val set	0.7823	0.2950
system performing best on test set	0.7902	0.2961
Min Combination	0.8222	0.3009
Max Combination	0.8307	0.3055
Product Combination	0.8615	0.3070
Median Combination	0.8637	0.3071
Average Combination	0.8751	0.3082
Oracle Combination	0.9951	0.3213

In Fig. 3 selected *precision-recall* curves for the global threshold approach are shown, namely the curves of the best (*Average*) and worst (*Min*) combination as well as the best underlying system and the *Oracle*. (We do not show all of the curves to improve readability.) An improvement of the average precision can observed using any of the presented combination methods. The greatest improvement is achieved using the *Average* Combination. It increases the average precision by 0.1175 compared to the mean of all single systems and 0.0849 compared to the best single system considering the threshold based evaluation and 0.0156 resp. 0.0121 considered the rank based averaged interpolated precision.

The *Oracle* curve in Fig. 3 is close to that of a perfect keyword spotting system, which would have a *precision* of 1 for all *recall* values in the treshold based and 0.3218 in the rank based framework. We conclude from this observation that the ensemble has a tremendous potential, in principal, which is not yet fully exploited by the combination rules applied in our current system.

4.3 Diversity Analysis

Comparing two systems in order to determine the values a, b, c, and d (see Section 3) cannot be done by using the same threshold for both systems, since the performance is independent of the absolute likelihood values the CTC algorithm returns (only their relative distribution is important). Instead we compare two systems at equal *recall* values. Given such a value, the corresponding threshold was computed for each system separately. For a combination of more than two systems, the mean of all pairwise diversity values is used. Using this approach, the *correlation* of the underlying base systems as well as the *precision* increase of a combination compared to the best underlying system can be plotted as a function of the *recall* value. In Fig. 4 these two functions resulting from the *Median* combination, representative of all combination rules, are given. The curves for the other combination schemes look very similar. One can see that using an adequate combination method, the *precision* can be more than tripled for high *recall* values. A clear symmetry between the *correlation* and *improvement* plots can be observed in Fig. 4. The lower the *correlation* between the underlying keyword spotting systems is, the higher is the relative improvement of the best combination's *precision* compared to the best individuals system's *precision*. In

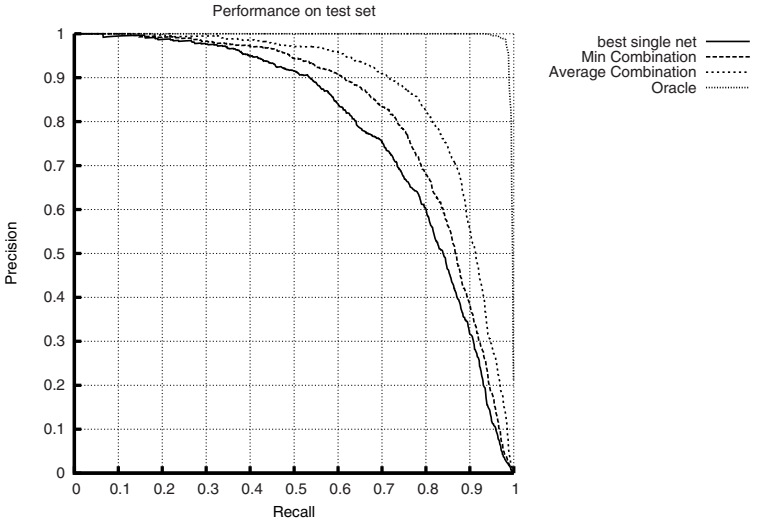


Fig. 3. Threshold based precision-recall plot of the best and worst combinations, the best underlying base systems and the oracle

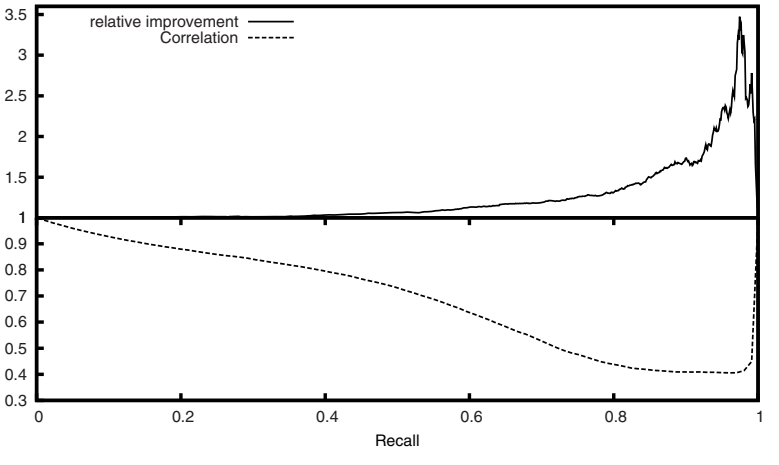


Fig. 4. The *precision* improvement factor of the *Median* combination method relative to the best underlying base system (upper part of the plot) and relative to the *correlation* of the underlying systems (lower part), as a function of the combined system's *recall* rate

other words, our expectation that the less the underlying systems correlate with each other, the higher is the improvement gained by combining them has been clearly experimentally verified.

5 Conclusion

A keyword spotting system returns, for a given keyword, all text segments (text lines in our case) in which the keyword occurs. In this paper, we proposed to combine different keyword spotting systems. To the knowledge of the authors this is a novel application in the field of multiple classifier systems that has not been proposed before. We used several neural networks as the base keyword spotting systems. In our approach, the likelihood values for the considered keyword to appear in the given text segment are combined prior to the comparison against a threshold.

We demonstrated that, using an adequate combination method, it is possible to construct a new system that is better at the task of keyword spotting than any of their their underlying base systems. The increase in performance correlates with the diversity of the base systems. The lower the base systems' *correlation* coefficient, the greater is the increase of the *precision*.

In the future, we will investigate the combination of structurally diverse keyword spotting systems, based on DTW, hidden Markov models, and different sets of features. Weighted combinations and trainable combination rules, such as neural networks, are also along this line of research.

Acknowledgments

This work has been supported by the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM2) and the the Swiss National Science Foundation (Project CRSI22_125220/1). We thank Alex Graves for kindly providing us with the BLSTM Neural Network source code.

References

1. Vinciarelli, A.: A Survey On Off-Line Cursive Word Recognition. Pattern Recognition 35(7), 1433–1446 (2002)
2. Plamondon, R., Srihari, S.N.: On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Transaction on Pattern Analysis and Machine Intelligence 22(1), 63–84 (2000)
3. Levy, S.: Google's two revolutions. Newsweek, December 27-January 3 (2004)
4. Kołcz, A., Alspector, J., Augusteijn, M.F., Carlson, R., Popescu, G.V.: A Line-Oriented Approach to Word Spotting in Handwritten Documents. Pattern Analysis and Applications 3, 153–168 (2000)
5. Manmatha, R., Rath, T.M.: Indexing of Handwritten Historical Documents - Recent Progress. In: Symposium on Document Image Understanding Technology, pp. 77–85 (2003)

6. Rath, T.M., Manmatha, R.: Word Image Matching Using Dynamic Time Warping. In: Computer Vision and Pattern Recognition, vol. 2, pp. 521–527 (2003)
7. Ataer, E., Duygulu, P.: Matching Ottoman Words: An Image Retrieval Approach to Historical Document Indexing. In: 6th Int'l. Conf. on Image and Video Retrieval, pp. 341–347 (2007)
8. Leydier, Y., Lebourgeois, F., Emptoz, H.: Text Search for Medieval Manuscript Images. Pattern Recognition 40, 3552–3567 (2007)
9. Srihari, S.N., Srinivasan, H., Huang, C., Shetty, S.: Spotting Words in Latin, Devanagari and Arabic Scripts. Indian Jounal of Artificial Intelligence 16(3), 2–9 (2006)
10. Zhang, B., Srihari, S.N., Huang, C.: Word Image Retrieval Using Binary Features. In: Proceedings of the SPIE, vol. 5296, pp. 45–53 (2004)
11. Edwards, J., Whye, Y., David, T., Roger, F., Maire, B.M., Vesom, G.: Making Latin Manuscripts Searchable using gHMM's. In: Advances in Neural Information Processing Systems (NIPS) 17, pp. 385–392. MIT Press, Cambridge (2004)
12. Cao, H., Govindaraju, V.: Template-free Word Spotting in Low-Quality Manuscripts. In: 6th Int'l. Conf. on Advances in Pattern Recognition (2007)
13. Perronnin, F., Rodriguez-Serrano, J.: Fisher Kernels for Handwritten Wordspotting. In: 10th Int'l Conf. on Document Analysis and Recognition, vol. 1, pp. 106–110 (2009)
14. Jiang, H., Li, X.: Incorporating training errors for large margin hmms under semidefinite programming framework. In: Int'l. Conf. on Acoustics, Speech and Signal Processing, April 2007, vol. 4, pp. 629–632 (2007)
15. Frinken, V., Fischer, A., Bunke, H.: A Novel Word Spotting Algorithm Using Bidirectional Long Short-Term Memory Neural Networks. In: 4th Workshop on Artificial Neural Networks in Pattern Recognition (2010)
16. Marti, U.V., Bunke, H.: The IAM-Database: An English Sentence Database for Offline Handwriting Recognition. Int'l Journal on Document Analysis and Recognition 5, 39–46 (2002)
17. Marti, U.V., Bunke, H.: Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. Int'l Journal of Pattern Recognition and Artificial Intelligence 15, 65–90 (2001)
18. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence 31(5), 855–868 (2009)
19. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist Temporal Classification: Labelling Unsegmented Sequential Data with Recurrent Neural Networks. In: 23rd Int'l Conf. on Machine Learning, pp. 369–376 (2006)
20. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
21. Salton, G.: The SMART Retrieval System—Experiments in Automatic Document Processing. Prentice-Hall, Inc., Upper Saddle River (1971)
22. Pudil, P., Novovicova, J., Kittler, J.: Floating Search Methods in Feature Selection. Pattern Recognition Letters 15(11), 1119–1125 (1994)

Combining Committee-Based Semi-supervised and Active Learning and Its Application to Handwritten Digits Recognition

Mohamed Farouk Abdel Hady and Friedhelm Schwenker

Institute of Neural Information Processing
University of Ulm
D-89069 Ulm, Germany
{mohamed.abdel-hady,friedhelm.schwenker}@uni-ulm.de

Abstract. *Semi-supervised learning* reduces the cost of labeling the training data of a supervised learning algorithm through using unlabeled data together with labeled data to improve the performance. *Co-Training* is a popular *semi-supervised learning* algorithm, that requires multiple redundant and independent sets of features (views). In many real-world application domains, this requirement can not be satisfied. In this paper, a single-view variant of *Co-Training*, *CoBC* (Co-Training by Committee), is proposed, which requires an ensemble of diverse classifiers instead of the redundant and independent views. Then we introduce two new learning algorithms, *QBC-then-CoBC* and *QBC-with-CoBC*, which combines the merits of committee-based *semi-supervised learning* and committee-based *active learning*. An empirical study on handwritten digit recognition is conducted where the random subspace method (*RSM*) is used to create ensembles of diverse C4.5 decision trees. Experiments show that these two combinations outperform the other non committee-based ones.

1 Introduction

Supervised learning algorithms require a large amount of labeled data in order to achieve high accuracy. However, labeling data is often tedious, time consuming and expensive, as it requires the efforts of experienced human annotators such as biologists, radiologists or physicians. In many real-world data mining applications, there is often an extremely large pool of data available. *Semi-supervised learning* (SSL) and active learning (AL) both attempt to exploit the unlabeled data to improve the recognition performance of supervised learning algorithms and to minimize the cost of data labeling.

Self-Training (ST) [1] is a single-view SSL algorithm, in which a single classifier is initially trained using a small amount of labeled data. Then it iteratively classifies the unlabeled examples, rank the examples by confidence in their prediction and adds *permenantly* the *most confident* examples with their predicted labels into the training set. Then, the underlying classifier is retrained with the augmented training set. A corresponding active learning algorithm is *Uncertainty Sampling* (US), in which the *least confident* examples are selected to be classified by a human expert before they are added into the training set.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 225–234, 2010.

Co-Training (*CT*) is a widely applied multi-view *SSL* paradigm first introduced by Blum and Mitchell [2], in which two classifiers are initially trained using two redundant and independent sets of features (views). Then at each further iteration, each classifier classifies the unlabeled examples, adds *permenantly* the examples about which it is *most confident* into the training set. The aim is that the *most confident* examples with respect to one classifier can be informative with respect to the other and help to improve its accuracy. Although Nigam and Ghani [1] showed that *Co-Training* is sensitive to the multi-view requirement, it can not be fulfilled in most real-world application. In [3], a corresponding multi-view active learning algorithm, *Co-Testing*, is proposed.

Query by Committee (*QBC*) [4] is a committee-based active learning algorithm, in which an ensemble of diverse classifiers is constructed. Then the ensemble members are applied to unlabeled examples. The most informative examples are the *least confident* ones on which the ensemble members are mostly disagree. Then an expert is asked to assign labels to these examples and then the committee is re-trained using the augmented training set.

Now, we can see that *semi-supervised learning* and *active learning* tackle the same problem but from different directions. *Semi-supervised learning* exploits what the underlying classifiers are *most confident* from the unlabeled data, and *active learning* exploits the *least confident* ones. Although, there are some approaches that combine *semi-supervised learning* and *active learning* to integrate the benefits of the two, such as [5,6,7], there is no work done to investigate the combination of *committee-based* semi-supervised learning with *active learning*. The main objective of this study is to relax the hard impractical requirement of *Co-Training* through using a set of diverse classifiers instead of using independent views. First, we propose a single-view variant of *Co-Training*, *CoBC* (Co-Training by Committee). Second, two new learning algorithms are introduced for combining *active learning* and *semi-supervised learning* with committees, which are denoted by *QBC-then-CoBC* and *QBC-with-CoBC*.

The rest of the paper is organized as follows: the related work is given in Section 2. Section 3 presents the proposed *CoBC* framework before the new combinations of *committee-based* semi-supervised learning and *committee-based* active learning are introduced in Section 4. The results of our experiments are reported in Section 5. Finally, we conclude our work in Section 6.

2 Related Work

2.1 Single-View Committee-Based Co-Training

A full review of ensemble-based semi-supervised algorithms is out of the scope of this paper; the reader might refer to [8] for a more extensive survey. We now briefly review one of the recent studies [9] that investigated the applicability of *Co-Training* without multiple views. Li and Zhou [9] proposed a *Co-Training* style algorithm, called *Co-Forest*, in which an initial ensemble of random trees is trained on bootstrap subsamples generated from the given labeled data set L.

To select new training examples for each ensemble member h_i ($i = 1, \ldots, N$) from a given unlabeled data set U, a new ensemble H_i, called the concomitant ensemble of h_i, is defined that contains all the other classifiers except h_i. At each iteration t and for each ensemble member h_i, first the error rate $\hat{\epsilon}_{i,t}$ of H_i is estimated. If $\hat{\epsilon}_{i,t}$ is greater than $\hat{\epsilon}_{i,t-1}$ (1^{th} condition), H_i predicts the class label of the unlabeled examples in $U'_{i,t}$ (random subsample of U). A set $L'_{i,t}$ is defined that contains the unlabeled examples in $U'_{i,t}$ where the confidence of H_i about their prediction exceeds a predefined threshold (θ) and $W_{i,t}$ is the sum of the confidences of the examples in $L'_{i,t}$. Finally, if $W_{i,t}$ is greater than $W_{i,t-1}$ (2^{nd} condition) and $\hat{\epsilon}_{i,t}W_{i,t}$ is less than $\hat{\epsilon}_{i,t-1}W_{i,t-1}$ (3^{rd} condition), the i^{th} random tree will be re-trained using the original labeled data set L and $L'_{i,t}$. The algorithm will stop if there is not a classifier h_i where the three conditions are fulfilled. We have two concerns: First, the error rate $\hat{\epsilon}_{i,t}$ is estimated accurately only at the first iteration as it depends on the *out-of-bag error estimation*, afterward the estimation tends to be an under-estimate as it depends on the training set. Therefore, *Co-Forest* will stop when the training error of a classifier reaches *zero*, for instance this is true for the *Nearest Neighbor* classifier. Second, setting the value of θ is not straightforward especially for multi-class problems where the confidence of the concomitant ensemble H_i is distributed among many classes. If θ is high, the 2^{nd} condition will not be fulfilled and the algorithm will stop. If θ is low, the size of $L'_{i,t}$ might be large and even equal to $U'_{i,t}$ which increases the risk that h_i will receive a lot of mislabeled examples.

2.2 Combining Active Learning and Semi-supervised Learning

In [5], two ways to combine *QBC* and semi-supervised EM are introduced and applied for text classification. The results have shown that combining *QBC* and semi-supervised EM outperform both of them. In [6], *Co-Testing* and *Co-EM* are combined, called *Co-EMT*, to produce an active multi-view semi-supervised algorithm. The experimental results on web page classification show that *Co-EMT* outperforms other non-active multi-view algorithms (*Co-Training* and *Co-EM*) without using more labeled data and that it is more robust to the violation of the requirements of two independent and redundant views. In [7], the experiments show that a combination of *Co-Testing* and *Co-Training* like schemes, called *SSAIR*, can exploit the unlabeled images to improve the performance of content-based image retrieval.

3 Co-Training by Committee

CoBC works as follows (see Algorithm 1): initially a committee of N diverse accurate classifiers $H^{(0)}$ is constructed with *EnsembleLearn* and *BaseLearn* using L. Then the following steps are repeated until a maximum number of iterations T is reached or U becomes empty. For each iteration t and for each classifier i, a set $U'_{i,t}$ of u examples drawn randomly from U without replacement. It is computationally more efficient to use $U'_{i,t}$ instead of using the whole set U.

Algorithm 1. The pseudo code of Co-Training by Committee framework

Require: set of labeled training examples (L), set of unlabeled training examples (U), maximum number of iterations (T), ensemble learning algorithm ($EnsembleLearn$), base learning algorithm ($BaseLearn$), number of committee members (N), number of unlabeled examples in the pool (u), number of nearest neighbors (k), sample size (n), number of classes (C) and an initial committee ($H^{(0)}$)

Training Phase

1: Get the class prior probabilities, $\{Pr_c\}_{c=1}^{C}$
2: Set the class growth rate, $n_c = n \times Pr_c$ where $c = 1, \ldots, C$
3: **if** $H^{(0)}$ is not given **then**
4: Construct an initial committee, $H^{(0)} = EnsembleLearn(L, BaseLearn, N)$
5: **end if**
6: **for** $t \in \{1, \ldots, T\}$ **do**
7: $L'_t \leftarrow \phi$
8: **if** U is empty **then** $T \leftarrow t$-1 and abort loop **end if**
9: **for** $i \in \{1, \ldots, N\}$ **do**
10: $U'_{i,t} \leftarrow RandomSubsample(U, u)$
11: $\pi_{i,t} \leftarrow SelectCompetentExamples(i, U'_{i,t}, H_i^{(t-1)}, k, \{n_c\}_{c=1}^{C}, C)$
12: $L'_t \leftarrow L'_t \cup \pi_{i,t}$, $U'_{i,t} \leftarrow U'_{i,t} \setminus \pi_{i,t}$ and $U \leftarrow U \cup U'_{i,t}$
13: **end for**
14: **if** L'_t is empty **then** $T \leftarrow t$-1 and abort loop **end if**
{ Re-train the N classifiers using their augmented training sets}
15: **for** $i \in \{1, \ldots, N\}$ **do** $L_i \leftarrow L_i \cup L'_t$ and $h_i^{(t)} \leftarrow BaseLearn(L_i)$ **end for**
16: **end for**

Prediction Phase

17: **return** $H^{(T)}(x) \leftarrow \frac{1}{N} \sum_{i=1}^{N} h_i^{(T)}(x)$ for a given sample x

The method *SelectCompetentExamples* is applied to estimate the competence of each unlabeled example in $U'_{i,t}$ given the companion committee $H_i^{(t-1)}$. A set $\pi_{i,t}$ is created that contains the n_c most competent examples assigned to class ω_c. Then $\pi_{i,t}$ is removed from $U'_{i,t}$ and inserted into the set L'_t that contains all the examples labeled at iteration t. The remaining examples in $U'_{i,t}$ are returned to U. We have two options: (1) if the underlying ensemble learner depends on training set perturbation to promote diversity, then insert $\pi_{i,t}$ only into L_i. (2) otherwise, insert $\pi_{i,t}$ into the training sets of all classifiers as shown in step 15. Then, the N classifiers are retrained using their updated training set L_i.

3.1 Confidence Estimation Using Local Competence

An important factor that affects the performance of any *Co-Training* style algorithm is how to measure its confidence about the labeling of an example which determines its probability of being selected. An inaccurate confidence measure can lead to adding mislabeled examples to the training set which leads to performance degradation during the SSL process.

Algorithm 2. The pseudo code of *SelectCompetentExamples* method

Require: pool of unlabeled examples ($U'_{i,t}$), the companion committee of classifier $h_i^{(t-1)}$ ($H_i^{(t-1)}$), number of nearest neighbors k and growth rate ($\{n_c\}_{c=1}^{C}$)

$\pi_{i,t} \leftarrow \phi$
for each class $\omega_c \in \{\omega_1, \ldots, \omega_C\}$ **do** $count_c \leftarrow 0$ **end for**
for each $x_u \in U'_{i,t}$ **do**
  $H_i^{(t-1)}(x_u) \leftarrow \frac{1}{N-1} \sum_{j=1,\ldots,N, j \neq i} h_j^{(t-1)}(x_u)$
  Get the class label assigned to x_u by committee $H_i^{(t-1)}$,
  $\omega_{pred} \leftarrow arg \max_{1 \leq c \leq C} H_i^{(t-1)}(x_u, \omega_c)$
  Get the neighborhood of x_u, $N(x_u) \leftarrow \{x_n | x_n \in Neighbors(x_u, k, L)\}$
  Calculate the local competence, $Comp(x_u, H_i^{(t-1)})$ as in eq. 2.
end for
Rank the examples in $U'_{i,t}$ by competence
{Select the n_c examples with the maximum competence for class ω_c}
for each $x_u \in U'_{i,t}$ **do**
  if $Comp(x_u, H_i^{(t-1)}) > 0$ and $count_{pred} < n_c$ **then**
    $\pi_{i,t} \leftarrow \pi_{i,t} \cup \{(x_u, \omega_{pred})\}$ and $count_{pred} \leftarrow count_{pred} + 1$
  end if
end for
return $\pi_{i,t}$

A labeling confidence can be assigned to each unlabeled example using its class probability estimate (*CPE*) provided by companion committee.

$$Confidence(x_u, H_i^{(t-1)}) = \max_{1 \leq c \leq C} H_i^{(t-1)}(x_u, \omega_c) \tag{1}$$

Unfortunately, in many cases the classifier does not provide an accurate *CPE*. For instance, a decision tree provides piecewise constant probability estimates. That is, all unlabeled examples x_u which lie into a particular leaf, will have the same *CPE*s because the exact value of x_u is not used in determining its *CPE*.

To measure confidence (see Algorithm 2), we estimate the companion committee accuracy in the neighborhood of an unlabeled example x_u, that is defined with respect to the initially labeled training set. The neighborhood could also be determined using a separate validation set, but it may be impractical to spend a part from the small-sized labeled data for validation. To avoid the inaccurate estimation of local accuracy that may result due to overfitting problem, the newly-labeled training examples $\pi_{i,t}$ will not be involved in the estimation. The local competency of an unlabeled example x_u given $H_i^{(t-1)}$ is defined as follows:

$$Comp(x_u, H_i^{(t-1)}) = \sum_{x_n \in N(x_u), x_n \in \omega_{pred}} \frac{W_n . H_i^{(t-1)}(x_n, \omega_{pred})}{||x_n - x_u||_2 + \epsilon} \tag{2}$$

where ω_{pred} is the class label assigned to x_u by $H_i^{(t-1)}$ and ϵ is a constant added to avoid zero denominator.

4 Combining QBC and CoBC

4.1 QBC Then CoBC

The most straightforward method of combining *QBC* and *CoBC* is to run *CoBC* after *QBC*. The objective is that active learning can help *CoBC* through providing it with a better starting point instead of randomly selecting examples to label for the starting point. *QBC* selects the training examples that *CoBC* cannot reliably label on its own. Hence, we expect that *QBC-then-CoBC* will outperform both stand-alone *QBC* and stand-alone *CoBC*. In addition, we expect that *QBC-then-CoBC* will outperform other possible combinations of non committee-based *active learning* and *semi-supervised learning* algorithms.

4.2 QBC with CoBC

A more interesting approach is to interleave *CoBC* with *QBC*, so that *CoBC* not only runs on the results of active learning, but *CoBC* also helps active learning. To do this, at each *QBC* round, we run *CoBC* for a predefined number of iterations (T_{CoBC}). The objective is improve the performance of the committee members through updating them with the most competent examples selected by *CoBC*. With more accurate committee members, *QBC* should select more informative examples to label. Hence, we expect that *QBC-with-CoBC* will outperform both stand-alone *QBC* and stand-alone *CoBC*. In addition, we expect that *QBC-with-CoBC* will outperform *QBC-then-CoBC*. The reason is that in *QBC-then-CoBC*, *QBC* does not benefit from *CoBC*. On the other hand, in *QBC-with-CoBC*, both algorithms are benefiting from each other.

5 Experimental Evaluation

The experiments are conducted on four feature sets describing handwritten digits and publicly available at UCI Repository [10]. The digits were extracted from a collection of Dutch utility maps. A total 2,000 patterns (200 patterns per class) have been digitized in binary images (see Table 1 and Figure 1).

5.1 Methodology

In all experiments, the pruned C4.5 decision tree with Laplace Correction (*J48*) was used as the base learning algorithm and the *RSM* [11] was used to construct

Table 1. Description of the four sets of features used for handwritten digits recognition

Name	Description
mfeat-pix	240 pixel averages in 2 x 3 windows
mfeat-kar	64 Karhunen-Love coefficients
mfeat-fac	216 profile correlations
mfeat-fou	76 Fourier coefficients of the character shapes

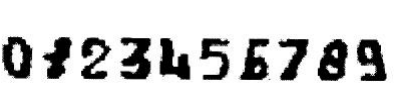


Fig. 1. Sample of the handwritten digits

decision forests of size ten ($N = 10$) where each tree uses only half of the available features that were randomly selected. For classification, *average* is employed to combine the decisions of the individual trees. All algorithms are implemented using WEKA library [12]. All the features are normalized to have zero mean and unit variance. For each experiment, 4 runs of 10-fold cross-validation have been performed. That is, for each data set, 10% (200 patterns) are used as test set, while the remaining 90% (1800 patterns) are used as training examples. For comparsion, paired t-test with 0.05 significance level is used as significance test where significance is marked with bullet(•). For SSL algorithms, 10% of the training examples (180 patterns) are randomly selected as the initial labeled data set L while the remaining 90% are used as unlabeled data set U. The number of iterations T is chosen such that the algorithm stops when the number of labeled examples in L reaches 60% (1080 patterns). For AL algorithms, only 5% of the training examples ($|L| = 90$ patterns) are randomly selected as L and the algorithms stops when the number of labeled examples in L reaches 10% ($|L| = 180$ patterns). We set the pool size u to 100, the sample size n to one and the number of nearest neighbors used to estimate local competence k is 10.

5.2 Results

Comparison between forests and trees. The *RSM* forests significantly outperform the single C4.5 decision trees for all datasets using 5%, 10% and 100% of training data set (see Table 2(a)). These results are considered as a basic requirement to continue our experiments.

Comparison between *CoBC* **and** *Self-Training*. For fair comparison, both algorithms are given the same L and U and allowed to label the same amount of unlabeled data. That is, both are initialized with 10% of the training data that are randomly selected and work until the size of L reaches 60% of the training data ($|L| = 1080$ patterns) which implies 90 iterations for Self-Training and only 9 iterations for *CoBC*. Table 2(b) presents the average error at iteration 0 (*initial*) trained on the initially labeled data set L, after the final SSL iteration of exploiting the unlabeled data set U (*final*) and the relative improvement percentage ($improv = \frac{initial-final}{initial} \times 100$). The final test error of *CoBC* is significantly better than its initial error on all the data sets except for *mfear-fac* where the difference is not significant. In addition, the final test error after *CoBC* is significantly better than the final error after *Self-Training* on all the data sets.

Comparison between *CPE* **and local competence confidence measures**. Table 2(b) and 2(c) show that the average error of *random-then-ST* and *random-then-CoBC* using *CPE* increases by 0.23% and 6.84% respectively, while

Table 2. Average test error rates of the compared algorithms applied to the handwritten digits (Part 1)

(a) Passive Supervised Learning (random sampling)						
Data set	$\|L\|$	*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
J48	5%	30.99(4.60)	42.04(3.62)	32.63(4.80)	41.74(4.74)	36.85
RSM(J48)	5%	22.78(4.17)$^\bullet$	31.53(4.41)$^\bullet$	23.23(4.82)$^\bullet$	35.10(3.95)$^\bullet$	28.16
J48	10%	25.48(3.45)	34.51(4.01)	25.03(2.79)	36.80(4.82)	30.45
RSM(J48)	10%	16.69(2.82)$^\bullet$	21.30(2.96)$^\bullet$	15.71(2.52)$^\bullet$	29.29(3.50)$^\bullet$	20.75
J48	100%	11.23(1.96)	17.54(2.10)	11.66(2.53)	23.71(2.63)	16.03
RSM(J48)	100%	5.10(1.61)$^\bullet$	7.90(1.52)$^\bullet$	5.33(1.71)$^\bullet$	17.69(2.42)$^\bullet$	9.00
(b) Passive *SSL* using *Competence* (Starting with 10% random sampling and Until 60%)						
Data set		*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
random-then-ST	*initial*	25.48(3.45)	34.51(4.01)	25.03(2.79)	36.80(4.82)	30.45
	final	22.50(2.81)$^\bullet$	29.20(4.16)$^\bullet$	21.50(3.30)$^\bullet$	34.15(3.76)$^\bullet$	26.84
	improv	11.70	15.39	14.10	7.20	11.86
random-then-CoBC	*initial*	16.69(2.82)	21.30(2.96)	15.71(2.52)	29.29(3.50)	20.75
	final	13.18(3.58)$^\bullet$	15.10(2.20)$^\bullet$	14.24(3.33)	25.65(3.03)$^\bullet$	17.04
	improv	21.03	29.11	9.36	12.43	17.88
(c) Passive *SSL* using *CPE* (Starting with 10% random sampling)						
Data set		*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
random-then-ST	*initial*	25.48(3.45)	34.51(4.01)	25.03(2.79)	36.80(4.82)	30.45
	final	25.01(3.50)	34.40(3.76)	25.14(3.52)	37.54(4.73)	30.52
	improv	1.84	0.32	-0.44	-2.01	-0.23
random-then-CoBC	*initial*	16.69(2.82)	21.30(2.96)	15.71(2.52)	29.29(3.50)	20.75
	final	19.24(3.02)	21.13(3.63)	19.00(3.91)	29.31(3.80)	22.17
	improv	-15.28	0.80	-20.94	-0.07	-6.84
Co-Forest ($\theta = 0.75$)	*initial*	14.51(3.05)	21.83(3.39)	14.11(2.59)$^\bullet$	29.35(3.18)	19.95
	final	16.01(3.59)	21.83(3.39)	17.18(3.01)	29.24(3.42)	21.06
	improv	-10.33	0.0	-21.75	0.37	-5.56

it decreases by 11.86% and 17.88% using local competence estimates. This emphasizes that inaccurate confidence measure leads to performance degradation.

Comparison between *CoBC* **and** *Co-Forest*. For fair comparison with *CoBC*, *Co-Forest* will be applied using the random subspace method and C4.5 decision trees as *CoBC*. However, the initial test error of *Co-Forest* is different from the initial error of *CoBC* because the *Co-Forest*'s initial C4.5 decision trees are not only trained using different random feature subsets but also trained using different bootstrap samples from L and *majority voting* is employed to produce the final decision of a forest. From Table 2(c), one can see that *Co-Forest* failed to improve the classification accuracy using unlabeled data. We can attribute the poor performance of *Co-Forest* to the irrelevant setting of θ (we get similar results for $\theta = 0.75$ and 0.6).

Comparison between *QBC* **and** *Uncertainty Sampling*. For fair comparison, both algorithms are given the same L and U and allowed to label the same amount of unlabeled data. That is, both are initialized with 5% of the training data ($|L| = 90$ patterns) that are randomly selected and work until the size of L reaches 10% of the training data ($|L| = 180$ patterns) which implies 9 iterations for both. Table 3(a) indicates that the final test error after *QBC* is significantly better than the final error after *US* on all the data sets.

QBC-then-CoBC and QBC-with-CoBC. Table 3(b) and Table 3(c) show that both *QBC-then-CoBC* and *QBC-with-CoBC* significantly outperform

Table 3. Average test error rates of the compared algorithms applied to the handwritten digits (Part 2)

(a) Active Learning (Starting with 5% random sampling and Until 10%)						
Data set		*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
UncertaintySamping	*initial*	30.99(4.60)	42.04(3.62)	32.63(4.80)	41.74(4.74)	36.85
	final	25.08(3.73)$^\bullet$	34.53(4.36)$^\bullet$	24.53(4.06)$^\bullet$	36.10(4.22)$^\bullet$	30.06
	improv	19.07	17.86	24.82	13.51	18.43
QBC	*initial*	22.78(4.17)	31.53(4.41)	23.23(4.82)	35.10(3.95)	28.16
	final	13.48(2.61)$^\bullet$	19.13(2.77)$^\bullet$	12.26(3.41)$^\bullet$	28.30(2.59)$^\bullet$	18.29
	improv	40.83	39.33	47.22	19.37	35.05
(b) Active *SSL* (Starting with 5% random sampling plus 5% selective sampling and Until 60%)						
Data set		*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
US-then-ST	*initial*	30.99(4.60)	42.04(3.62)	32.63(4.80)	41.74(4.74)	36.85
	final	20.86(4.31)$^\bullet$	28.04(3.56)$^\bullet$	19.41(3.76)$^\bullet$	33.42(4.09)	25.43
	improv	32.69	33.30	40.51	19.93	30.99
US-then-CoBC	*initial*	30.99(4.60)	42.04(3.62)	32.63(4.80)	41.74(4.74)	36.85
	final	12.21(3.00)	14.49(2.76)	12.88(2.95)	25.99(3.42)	16.39
	improv	60.60	65.53	60.53	37.73	55.52
QBC-then-ST	*initial*	22.78(4.17)	31.53(4.41)	23.23(4.82)	35.10(3.95)	28.16
	final	19.84(3.38)	28.55(3.63)	18.66(3.48)	32.29(2.97)	24.83
	improv	12.91	9.45	19.67	8.01	11.83
QBC-then-CoBC	*initial*	22.78(4.17)	31.53(4.41)	23.23(4.82)	35.10(3.95)	28.16
	final	11.04(2.11)$^\bullet$	14.71(2.26)$^\bullet$	11.95(2.65)	25.10(2.69)$^\bullet$	15.70
	improv	51.54	53.35	48.56	28.49	44.25
(c) Interleaving *AL* and *SSL* (Starting with 5% random sampling and Until 60%)						
Data set		*mfeat-pix*	*mfeat-kar*	*mfeat-fac*	*mfeat-fou*	*ave.*
QBC-with-CoBC	*initial*	22.78(4.17)	31.53(4.41)	23.23(4.82)	35.10(3.95)	28.16
	final	9.11(1.93)$^\bullet$	13.45(2.47)$^\bullet$	10.11(2.45)$^\bullet$	24.38(2.46)$^\bullet$	14.26
	improv	60.01	57.34	56.48	30.54	49.36

stand-alone QBC on all the data sets except for *mfeat-fac* where the improvement is not significant. In addition, *QBC-then-CoBC* insignificantly outperforms *random-then-CoBC* for all data sets while *QBC-with-CoBC* significantly outperforms *random-then-CoBC* for two data sets *mfeat-pix* and *mfeat-fac*. *QBC-with-CoBC* performs better than *QBC-then-CoBC* on all data sets but the improvement is significant only for *mfeat-pix* data set.

For comparison, we implemented three alternative combinations of active and *semi-supervised learning* algorithms: *US-then-ST*, *US-then-CoBC* and *QBC-then-ST*. If we sort all the algorithms based on average of the final test error for all data sets, we get (1) *QBC-with-CoBC* (14.26%), (2) *QBC-then-CoBC* (15.70%), (3) US-then-CoBC (16.39%), (4) *random-then-CoBC* (17.04%), (5) QBC-then-ST (24.83%), (6) *US-then-ST* (25.43%) and (7) random-then-ST (26.84%). This shows that combining committe-based active learning with committee-based SSL algorithm is superior to combining the non-committee ones.

6 Conclusions

In this paper, we introduced a single-view committee-based Co-Training style algorithm for *semi-supervised learning*, *CoBC*, for applications in which the data is not represented by multiple redundant and independent views. In addition, we proposed two new algorithms, *QBC-then-CoBC* and *QBC-with-CoBC*, that

combine the merits of *CoBC* and *QBC*. Experiments were conducted on four data sets for handwritten digits recognition, where the random subspace method and C4.5 decision tree are used as ensemble learner and base learner, respectively. The experiments on handwritten digits recognition show that *CoBC* exploits the unlabel data to improve the recognition accuracy. This improvement can be attribute to the local competence based confidence estimate that compensate the inaccurate class probability estimates of decision trees. In addition, both *QBC-with-CoBC* and *QBC-then-CoBC* outperform *QBC* and *CoBC*.

Acknowledgements

This work was partially supported by the Transregional Collaborative Research Centre SFB/TRR 62 *Companion-Technology for Cognitive Technical Systems* funded by the German Research Foundation (*DFG*). The first author was supported by a scholarship of the German Academic Exchange Service (*DAAD*).

References

1. Nigam, K., Ghani, R.: Analyzing the effectiveness and applicability of co-training. In: Proc. of the 9th Int. Conf. on Information and knowledge management, New York, NY, USA, pp. 86–93 (2000)
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of the 11th Annual Conf. on Computational Learning Theory (COLT 1998), pp. 92–100. Morgan Kaufmann Publishers, San Francisco (1998)
3. Muslea, I., Minton, S., Knoblock, C.A.: Selective sampling with redundant views. In: Proc. of the 17th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 621–626 (2000)
4. Freund, Y., Seung, H., Shamir, E., Tishby, N.: Selective sampling using the Query by Committee algorithm. Machine Learning 28(2-3), 133–168 (1997)
5. McCallum, A.K., Nigam, K.: Employing EM and pool-based active learning for text classification. In: Proc. of the 15th Int. Conf. on Machine Learning (ICML 1998), pp. 350–358. Morgan Kaufmann Publishers Inc., San Francisco (1998)
6. Muslea, I., Minton, S., Knoblock, C.A.: Active + Semi-Supervised learning = robust multi-view learning. In: Proc. of the 19th Int. Conf. on Machine Learning (ICML 2002), pp. 435–442 (2002)
7. Zhou, Z.H., Chen, K.J., Jiang, Y.: Exploiting unlabeled data in content-based image retrieval. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 525–536. Springer, Heidelberg (2004)
8. Zhou, Z.H., Li, M.: Semi-supervised learning by disagreement. Knowledge and Information Systems (in press)
9. Li, M., Zhou, Z.H.: Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. IEEE Trans. on Systems, Man and Cybernetics-Part A: Systems and Humans 37(6), 1088–1098 (2007)
10. Blake, C., Merz, C.: UCI repository of machine learning databases. University of California (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
11. Ho, T.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
12. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)

Using Diversity in Classifier Set Selection for Arabic Handwritten Recognition

Nabiha Azizi[1], Nadir Farah[1], Mokhtar Sellami[2], and Abdel Ennaji[3]

[1] Labged, Department of Computer Science, Badji Mokhtar University, Annaba Algeria
{azizi,farah}@labged.net
[2] Lri, Department of Computer Science, Badji Mokhtar University of Annaba, Algeria
sellami@lri-annaba.net
[3] Litis, Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes - EA 4108
Abdel.Ennaji@univ-rouen.fr

Abstract. The first observation concerning Arabian manuscript reveals the complexity of the task, especially for the used classifiers ensemble. One of the most important steps in the design of a multi-classifier system (MCS), is the its components choice (classifiers). This step is very important to the overall MCS performance since the combination of a set of identical classifiers will not outperform the individual members. To select the best classifier set from a pool of classifiers, the classifier diversity is the most important property to be considered. The aim of this paper is to study Arabic handwriting recognition using MCS optimization based on diversity measures. The first approach selects the best classifier subset from large classifiers set taking into account different diversity measures. The second one chooses among the classifier set the one with the best performance and adds it to the selected classifiers subset. The performance in our approach is calculated using three diversity measures based on correlation between errors. On two database sets using 9 different classifiers, we then test the effect of using the criterion to be optimized (diversity measures,), and fusion methods (voting, weighted voting and Behavior Knowledge Space). The experimental results presented are encouraging and open other perspectives in the classifiers selection field especially speaking for Arabic Handwritten word recognition.

Keywords: Combining classifiers, Arabic Handwritten word, classifier subset selection, diversity measures, voting, weighted voting, BKS.

1 Introduction

For almost any real world pattern recognition problem a series of approaches and procedures may be used to solve it. After more than 20 years of continuous and intensive effort devoted to solving the challenges of handwriting recognition, progress in recent years has been very promising [1]. Research on Arabic handwritten word and text recognition is still of great interest during the past few years [2], [3]. The multiple classifier system has been shown to be useful for improving recognition rates [4]. In the literature, the use of MCS has been widely used for several pattern recognition

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 235–244, 2010.

tasks [5], [6], [7], [8], [9]. One of the most important task in optimizing a multiple classifier system is to select a group of adequate classifiers from a pool of classifiers. These methods choose a small subset from a large set of candidate models. Since there are 2^L-1 possible subsets of L models [10], it is not possible to try all the possibilities unless the subset *L* is small [11]. Subset classifier selection methods also differ in the criterion they optimize. Additional to methods which directly optimize ensemble accuracy, *diversity* measures play an important role in selecting and explaining this classifiers subset choice.

Diversity should therefore be regarded in a more general context than as a way of finding the best classifiers combination for a specific combination method. Optimally it should produce a classifiers set member that are different from each other in a way that is beneficial for classifier combining in general, regardless of the actual combination method [12], [13]. We can resume that there are three major topics associated with subset classifier selection: set creation, set selection and classifier combination.

In this paper we propose two new approaches for Arabic handwritten recognition based on diversity measures for selecting the best classifier subset from a proposed classifier ensemble. The first investigates the effect of several diversity measures and fusion methods in handwritten recognition system. The second proposes a progressive algorithm combining accuracy and diversity in classifier selection. Three combination methods are tested in the two proposed approaches (voting, weighted voting, and BKS (Behavior Knowledge Space)).

This paper is organized as follows: In the Section 2, we illustrate the main Arabic handwriting properties. Pre processing and the different families of the used features are presented in Section 3. In the section 4, we debate adopted classifier combinations methods. The Section 5 retails the two proposed approaches after presentation of the used diversity measures. The databases used for the validation of the approaches and the experimental results are summarized in section 6.

2 Arabic Handwritten Properties

More than 300 million people speak the language, and over 1 billion people use it in several religion-related activities. The Arabic alphabet consists of 28 characters. Ten of them have one dot, three have two dots, and two have three dots. Dots can be above or below. The shape of the character is context sensitive, depending on its location within a word [14], [15], [16].The following image (Fig. 1) illustrates the diversity of Handwritten Arab properties causing problems during recognition.

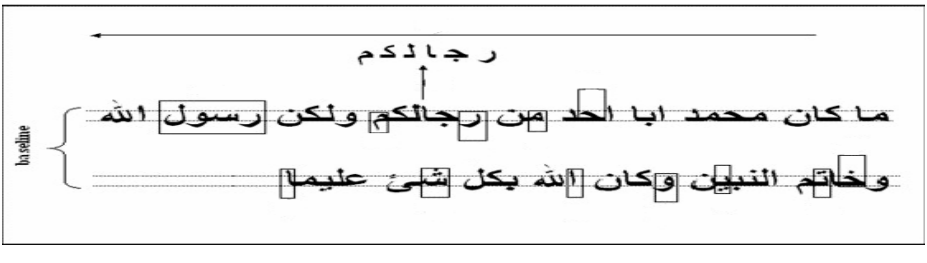


Fig. 1. Arabic Text Characteristics

3 Features Extraction

The choice of features to represent the patterns affects several aspects of the pattern recognition problem such as: accuracy required, learning time, and the necessary number of samples. In our case, the processing of Arabic handwritten writing, with its quoted morphology problems detailed in section 2, a cooperation of several types of primitives could be imposed according to the wide variability of the word shapes. Among the different feature types, we adopted two classes:

3.1 The Structural Features

In this system, we considered the following global structural features (Fig. 2) which are detailed in [8], [18], [19]:

- The number of : Connected components (using contour tracing), Descendants, Ascendants, unique dot below the baseline, unique dot above the baseline, Two dots below the baseline, Two dots bound above the baseline, Three bound dots, "Hamzas" (zigzags), Stroke (Loop), "Tasnine" (by calculation of the middle intersection number in the median zone) , and Concavity features with the four configurations [17].

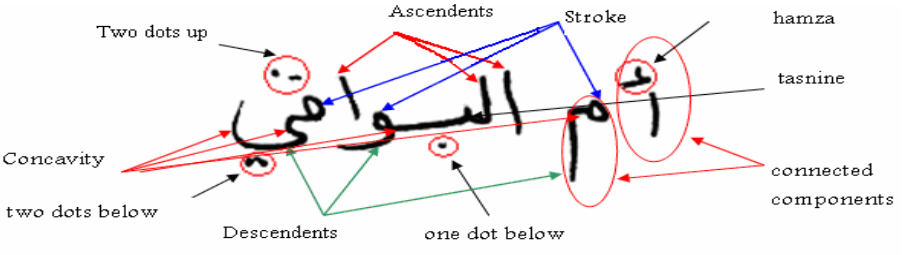


Fig. 2. Structural Features

3.2 The Statistical Features

The statistical features, in our approach are the density measures or Zoning [34]. For this study, the following subdivisions are used **(Fig. 3)**: For every zone, two statistical measures are calculated that are the black pixels density and the variance (to localize the position of the black pixels in every zone selected). The statistical features number is then : 35*2+36*2 = 142.

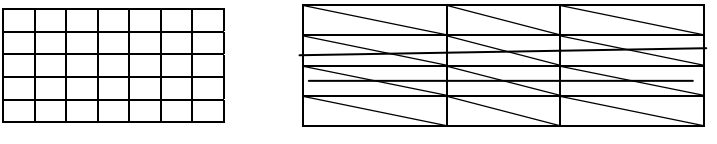


Fig. 3. Example of the used Zoning

4 Combination of Classifiers

Given a classifiers set, the easiest way to calculate the overall output is by taking a sum, which corresponds to taking a vote. There are other fixed rules, i.e., median, product, minimum, or maximum [2]. Alkoot and Kittler [21] investigated fixed rules and concluded that the sum and the median rules are more robust to noise. Concerning the MCS combination method which has heterogeneous outputs, we must adopt methods based on the class label and not on the score of all output classes. Among these methods, we have used the voting, the weighted voting and the BKS (Behaviour-Knowledge Space Method) [22], [23], [24], [25], [26].

Behaviour-Knowledge Space Method. It provides a knowledge space by collecting the decisions of all classifiers for each learned sample.

If the decision fusion problem is defined as a mapping of K classifiers: $e_1, ..., e_K$ into M classes: $c_1, ..., c_M$, the method operates on the K – dimensional space. Each dimension corresponds to an individual classifier, which can produce $M+1$ crisp decisions, M class labels and one rejection decision. A unit of BKS is an intersection of decisions of every single classifier. Each BKS unit contains three types of data: the total number of incoming samples: $T_{e1,...,eK}$, the best representative class: $R_{e1,...,eK}$, and the total number of incoming samples for each class: $n_{e1,...,eK}(m)$ [26],[27] .

5 Architecture of the Two Proposed Set Classifier Selection Approaches

Different works have been done in the field of AOCR (Arabic Optical Character Recognition) [27], [28], [29], [8], [9], [15].

To select the set of classifiers having the best individual performances doesn't imply a better recognition rate in any case in the global system. It is justified by the classifiers nature [29], [30]. In this section, we present the adopted diversity measures followed by the two proposed approaches details.

5.1 Measure of Diversity between Classifiers

The diversity measure is calculated in term of the output value through all classifiers [30]. In this work, we used six well known diversity measures to construct the best classifier subset:

Correlation Between Errors*:* It is interesting to examine that the independence between the committed errors is beneficial for the MCS; the correlation between the classifiers errors is a natural choice to compare the classifier subsets [31]:

$$\rho_{a,b} = \frac{Cov[v_e^a, v_e^b]}{\sqrt{Var[v_e^a]Var[v_e^b]}} \tag{1}$$

v_e^a and v_e^b are binary vectors of the a and b classifiers errors. The best set is selected while choosing the one having the minimum average of these pairs of measures.

Q Average: The Q average Or Q statistic aims to calculate the similarity between two classifiers [12]. It is defined for two classifiers *a, b* as:

$$Q_{a,b} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{2}$$

Where N^{11} is the number of time where the two classifiers are correct, N^{00} the number of time where the two classifiers are incorrect, N^{01} and N^{10} represent the number of time where just the first or the second are either correct.

Disagreement Measure: This measure represents the ratio between the number of observations where one classifier is correct and the other is incorrect with respect to the total number of observations [31]:

$$D_{a,b} = \frac{N^{10} + N^{01}}{N} \tag{3}$$

Report Between Different and Same errors*:* The worst case of a classifiers set in a MCS is when these last present the same incorrect results; that will be less degradable if the labels of classes are different [3]:

$$r_{a,b}^{DME} = \frac{N_{Different}^{00}}{N_{same}^{00}} \tag{4}$$

Kohavi-Wolpert (KW) Variance: Let $l(x_j)$ be the number of classifiers that correctly recognize the features vector x_j. From the formula for the variance [32] the diversity measure becomes:

$$kw = \frac{1}{NL^2}\sum_{j=1}^{N}(L - l(x_j)) \tag{5}$$

Exponential of the errors numbers: It is remarkable that if the classifiers agree on the bad result, the performance of the system decreases meaningfully [3]. These errors can be calculated and weighted by the number of classifiers detecting the errors (bad result) in an exponential manner.

$$r_{c_1,\ldots.\,c_k}^{\exp} = \sum_{i=1}^{k} \frac{(N_{i\ same}^{0})^{i}}{N_{all}^{1}} \tag{6}$$

5.2 Our Approach Based on Diversity Measure for Set Classifier Selection

Being given a set of *L* classifiers already designed and tested, and being given a set of diversity measures; we want through this approach to study the following parameters which are necessary to the classifier subsets selection:

- Which size will have the selected subset?
- What is the diversity measurement that is the most adapted for the S.E.S?
- Once the subset selected, which fusion functions can be applied?

To achieve these goals, the approach steps suggested are summarized as follows:

1. The subset size to be selected (M) starting from (L) the existing classifiers must be fixed at the beginning.
2. During the training, and for each entry sample, for each classifier we safeguard his good and bad responses.
3. After the six diversity measurements application discussed above, for each one of these measurements we can retain, the most powerful subset taking into account on the diversity and not on recognition rate. Let us note that the diversity calculation of a subset of M classifiers for pairwise measure is the average of all the combinations of the set couples.
4. For the combination function and as the classifier outputs are heterogeneous, we can apply the three combination methods noted in section 4, using the class label not on the class probability generated.

5.3 Approach Based on a Progressive Algorithm for Classifiers Set Selection

Although the first approach results published in [9], it suffers from some limits which we can summaries in the following points:

• Cost in time and memory capacity during the diversity measurements calculation for all the possible m classifiers combinations.
• As the majority of the subset selection approaches of classifiers based on diversity, neglect the individual criterion to classify accuracy. This last is a very important factor in the Multi classifiers systems design.
• Indeed, a subset selected by the diversity measurement application can not contain *el* the most powerful classifier (with great recognition rate) or even more serious than that, can contain that the M weak classifiers which represent the most diversified ones. What inevitably degrades the total system recognition rate.
• From this we used the idea which tries to combine two criteria ACCURACY and DIVERSITY for a classifier subset, selection while avoiding the lasting test of all calculation of the possible combinations and of measurement diversity:

Our proposed approach chooses a fixed m out of all L base classifiers.

1) It starts with a set containing 1 classifier which is the best classifier (based on accuracy) during the test phase;
2) At each iteration, it chooses among all possible classifiers the one that best improves the global system performance when added to the current ensemble. The performance is calculated using evaluation criterion (the three diversity measure as we will discuss next).

Methods based on output labels classes as, voting, weight voting, and BKS detailed in section 4 will be used in our study.

6 Experimental Results and Discussion

6.1 Used Data Base

We used two different databases in order to validate the two proposed approaches: the first base is containing the 48 wilayas (Town) of Algeria, containing 10000 words, The second used database is The IFN/ENIT [33], database for Arabic handwritten words containing 26459 Arabic words handwritten consisting of the 946 Tunisian town/village names.

6.2 Used Classifier

The single classifier members used for the selection are:

- 02 SVM (Support Vector Machine), with the strategy "one against all ", elaborated under the library lisb SVM, version 2.7 [3]. The inputs on this SVM system are the structural features. We have used polynomial and Gaussian kernel function.
- 03 KPPV (k - Nearer Neighbor with K=2, 3, 5) [8].
- 03 NN (Neuronal Network with different number of the hidden layer neurons and different inputs corresponding in features families detailed in section 3 [8], [18].
- 02 HMM (discret and Continuous with modified viterbi algorithm) [19], [28].

6.3 Approaches Result

Classifiers are indexed from 01 to10. Their individual performances using the two databases are resumed in (Table 1).

Table 1. Individual classifier accuracy

Classifier index	Member classifier	Accuracy (database 01)	Accuracy (database 02)
01	SVM(1)	86.88	87.03
02	SVM(2)	87.12	87.69
03	KNN(1)	82.45	82.78
04	KNN(2)	83.41	83.42
05	KNN(3)	85.02	84.96
06	NN(1)	86.69	87.12
07	NN(2)	87.08	87.46
08	NN(3)	86.23	87.05
09	HMM(1)	88.23	88.78
10	HMM(2)	89.15	89.23

Approach Results: In this study, we have fixed *M* at 4 for set classifier size. Six diversity measures are applied for training, and the best combinations of subset classifiers of each measure is given in Table 2.

The obtained results have shown that diversity is very important in selecting member classifiers. We noted that “exponential” and “correlation” measures generate the most powerful set, better than combining best individual classifiers. The results using BKS are better than the one using a weighted vote as a combination method. This encourages taking this research path for average size databases.

Table 2. Best subset classifiers with the obtained performances (global accuracy)

		Data base 01			Data base 02		
Diversity measures	**Subset classifier**	**Voting**	**W/ vote**	**BKS**	**Voting**	**W/vote**	**BKS**
Correlation / errors	1,7,9,10	**90.12**	**92.41**	**92.86**	**90.45**	**93.14**	**93.64**
Q Average	2,9,5,8	85.23	86.25	87.33	85.89	86.63	86.41
Ratio / errors	2,8,9,10	89.36	90.02	91.12	90.74	90.85	91.74
Disagreement measure	4,5,6,8	85.45	86.99	86.06	85.82	87.65	86.36
Kohavi-Wolpert (KW) variance	1,6,7,10	89.36	90.32	90.49	90.12	91.10	91.14
Exponential numbers of the errors	2,7,9,10	**91.56**	**93.25**	**93.75**	**91.74**	**93.94**	**93.79**
Best Individual Classif	**1,2,9,10**	**89.12**	**90.68**	**91.16**	**90.45**	**91.68**	**91.56**

Progressive Algorithm results: Experimental results after execution of our progressive algorithm are resumed in table 3.

Table 3. Best subset classifiers with the obtained performances (global accuracy)

		Data base 01			Data base 02		
Diversity measures	**Subset classifier**	**Voting**	**Weight voting**	**BKS**	**Voting**	**Weight voting**	**BKS**
Correlation / errors	10,1,8,9	**91.18**	**92.56**	**92.91**	**91.74**	**93.26**	**94.89**
Q Average	10,2,7,8	91.15	91.78	92.25	91.74	92.14	93.08
Disagreement measure	10,1,7,9	91.56	93.16	93.96	91.66	93.45	93.81
Best individual classifiers	**1,2,9,10**	**89.12**	**90.68**	**91.16**	**90.45**	**91.68**	**91.56**

It is noticed that the obtained results using the progressive algorithm is better than the first approach; specially by applying the diversity measurement "QStatistic"; indeed the subset generated by the latter does not contain the best individual classifier (classifier # 10).

For the disagreement measurement, it can be well noted that it generates a subset of the more or less weak classifiers in the first approach; what justifies the weak performance of system MCS; but as soon as we introduced the strong classifiers (classifier #10), the subset generated by this measurement was completely changed, and gives a powerful results of the MCS.

In any case, it is now clear that MCS performance strongly depends on careful classifiers selection to be combined. The effectiveness of various classifier fusion methods depends again on the selections made within classifiers.

7 Conclusion

Two newly applied methods based on MCS for AOCR have been investigated in this paper. The Diversity notion may be obtained either by finding the most diversified subset of classifiers among the existing combinations. In order to reach a decent objective, two independent systems were designed. In the first proposed approach, six well known diversity measures were used in order to select the 04 most diversified

classifiers. The "exponential" measure offers the best MCS subset in terms of performances. For combination methods, BKS is very reliable and outperform weight voting and vote approaches for the IFN/INIT base.

With the optimizing aim, the first approach which tests all classifiers combinations in order to generate the best diversified subset, and which does not take into account individual classifier accuracy during the selection, the progressive algorithm was proposed; its goal is creating a subset containing at the beginning the best individual classifier (based on accuracy) and adds after each iteration the best classifier which returns the subset more diversified.

The obtained results are encouraging especially in the second approach when compared to prior works realized on Arabic handwritten recognition.

We must note that several classification errors are due to a bad detection of the baseline, the diacritics and the presence of ligatures.

References

1. Cheriet, M., Bunke, H., Hu, J.: NewFrontiers in Handwriting Recognition. Pattern Recognition, Editorial (2009)
2. Giacinto, G., Roli, F.: An approach to the automatic design of multiple classifier systems. Pattern Recognition Letters 22(1), 25–33 (2001)
3. Aksela, M.: Comparison of Classifier Selection Methods for Improving Committee Performance. In: Multiple Classifier Systems, UK, pp. 84–93 (2003)
4. Xu, l., Krzyzak, A., Suen, C.Y.: Methodes of combining multiple classifiers and their applications to handwritten recognition. IEEE trans. Syst., Man, Cybern. SMC-22(3), 418–435 (1992)
5. Czyz, J., Sadeghi, M., Kittler, J., Vandendorpe, L.: Decision fusion for face authentication. In: Proc. First Internat. Conf. on Biometric Authentication, pp. 686–693 (2004)
6. Zhou, D., Zhang, J.: Face recognition by combining several algorithms. Pattern Recognition 3(3), 497–500 (2002)
7. Canuto, A.M.P., Abreu, M.C.C., Oliveira, L.M., Xavier, J.C.: Using weighted dynamic classifier selection methods in ensembles with different levels of diversity. Int. J. Hybrid Intell. Syst. 3(3), 147–158 (2006)
8. Farah, N., Souici, L., Sellami, M.: Classifiers combination and syntax analysis for arabic literal amount recognition. Engineering Applications of Artificial Intelligence 19(1) (2006)
9. Azizi, N., Farah, N., Khadir, M., Sellami, M.: Arabic Handwritten Word RecognitionvUsing Classifiers Selection and features Extraction / Selection. In: 17th IEEE Conference in Intelligent Information System, IIS 2009, Proceedings of Recent Advances in Intelligent Information Systems, pp. 735–742. Academic Publishing House, Warsaw (2009)
10. Santos, A.M.: Investigating the influence of the choice of the ensemblemembers in accuracy and diversity of selection-based and fusion-based methods for ensembles. Pattern Recognition Letters 28, 472–486 (2007)
11. Ulas, A., Semerci, M., Yıldız, O.T., Alpaydın, E.: Incremental construction of classifier and discriminant ensembles. Information Sciences 179, 1298–1318 (2009)
12. Aksela, M., Laaksonen, J.: Using diversity of errors for selecting members of a committee classifier. Pattern Recognition 39, 608–623 (2006)
13. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley–Interscience, Chichester (2004)
14. Amin, A.: Off-line Arabic character recognition: The state of the art. Pattern Recognition 31(5), 517–530 (1998)

15. Ben Amara, N., Belaid, A.: Printed PAW Recognition based on planar Hidden Marcov Model. In: Proceedings of the 13th International Conference on Pattern Recognition, ICPR, vol. 2, pp. 220–226 (1998)
16. Al-Badr, B., Mohmoud, S.A.: Survey and bibliography of Arabic optical text recognition. Signal Processing 41, 49–77 (1995)
17. Benouareth, A., Ennaji, A., Sellami, M.: Arabic Handwritten Word Recognition Using HMMs with Explicit State Duration. EURASIP, Journal on Advances in Signal Processing 2008, Article ID 247354 (2008)
18. Azizi, N., Sari, T., Souici, L., Sellami, M.: Une architecture de combinaison floue de classifieurs neuronaux pour la reconnaissance de mots arabes manuscrits. In: CIFED 2002, Hammamet, Tunisie, pp. 89–96 (2002)
19. Azizi, N., Khadir, M.T., Sellami, M.: Un système HMM pour le tri postal algérien. In: 9th Maghrebian Conference on Information Technologies MCSEAI, pp. 256–262 (2006)
20. Alkoot, F.M., Kittler, J.: Experimental evaluation of expert fusion strategies. Pattern Recognition Letters 20, 1361–1369 (1999)
21. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE transactions on pattern analysis and machine intelligence 20(3) (1998)
22. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. 51(2), 181–207 (2003)
23. Ruta, D., Gabrys, B.: Application of the Evolutionary Algorithms for Classifier Selection in Multiple Classifier Systems with Majority Voting. In: Multiple Classifier Systems, pp. 399–408 (2001)
24. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 4–37 (2000)
25. Ko, A.H.R., Sabourina, R., Britto, A., Oliveira, L.: Pairwise fusion matrix for combining classifiers. Pattern Recognition 40, 2198–2210 (2007)
26. Raudys, S., Roli, F.: The Behavior Knowledge Space Fusion Method: Analysis of Generalization Error and Strategies for Performance Improvement (2006)
27. Dara, R.A.: Cooperative Training in Multiple Classifier Systems; thesis: Doctor of Philosophy; Waterloo, Ontario, Canada (2007)
28. Benouareth, A., Ennaji, A., Sellami, M.: Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition. Pattern Recognition Letters 29(12), 1742–1752 (2008)
29. Ruta, D., Gabrys, B.: A theoretical analysis of the limits of majority voting errors for multiple classifier systems. Pattern Anal. Appl. 5(4), 333–350 (2002)
30. Aksela, M.: Adaptive combinations of classifiers with application to on-line handwritten.character recognition, dissertations in computer and information science report d19, espoo (2007)
31. Kapp, M.N., Sabourin, R., Maupin, P.: An Empirical Study on Diversity Measures and Margin Theory for Ensembles of Classifiers. In: 10th International Conference on Information Fusion, pp. 1–8 (2007)
32. Kohavi, H., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In: Saitta, L. (ed.) Proceedings of the Thirteenth International Conference on Machine Learning, pp. 275–283 (1996)
33. Pechwitz, M., Maergner, V.: HMM Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT Database. In: ICDAR, Scotland, pp. 890–894 (2003)
34. Suen, C.Y., Guo, J., Li, Z.C.: Analysis and recognition of alphanumeric handprints by parts. IEEE Trans. on Systems, Man, and Cybernetics 24(4), 614–631 (1994)

Forecast Combination Strategies for Handling Structural Breaks for Time Series Forecasting

Waleed M. Azmy[1], Amir F. Atiya[2], and Hisham El-Shishiny[3]

[1] Faculty of Computers and Information, Cairo University, 12613 Giza, Egypt
w.azmy@fci-cu.edu.eg
[2] Faculty of Engineering, Cairo University, Giza, Egypt
amir@alumni.caltech.edu
[3] IBM Center of Advanced Studies in Cairo, IBM Cairo Technology Development Center, Giza, Egypt
shishiny@eg.ibm.com

Abstract. Time-series forecasting is an important research and application area. Much effort has been devoted over the past decades to develop and improve the time series forecasting models based on statistical and machine learning techniques. Forecast combination is a well-established and well-tested approach for improving forecasting accuracy. Many time series may contain some structural breaks that may affect the performance of forecasting due to the varying nature of the dynamics with time. In this study we investigate the performance of using forecast combination in handling these breaks, and in mitigating the effects of discontinuities in time series.

Keywords: Time series forecasting, MLP, GPR, Exponential smoothing, Forecast combination, Structural breaks.

1 Introduction

Machine learning models have been prevalently used for time series forecasting. Examples of successful models are neural networks, support vector machines, Gaussian process-regression, regression trees [1]. The prevalent approach has been to use all the history of the time series as a training set for designing the forecasting model [2] [3] [4]. The problem, however, is that the dynamics of the time series often vary over time. The variations of the dynamics are usually caused by a gradual change in the effects of external factors, or by sudden events. Events or breaks in the time series are characterized by periods that differ significantly from some underlying baseline [5].

Structural changes or "breaks" appear to affect many different types of time series, such as economic and financial time series. The occurrence of breaks could reflect legislative, institutional or technological changes, shifts in economic policy, or could even be due to large macroeconomic shocks such as the doubling or quadrupling of oil prices experienced over the past decades [6]. For example, a structural break occurring due to the September 11, 2001 terrorist incident affected airline passenger travel time series data. A key question that arises in the context of time-series forecasting is how future values of the variables of interest might be affected by breaks.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 245–253, 2010.

In traditional time series forecasting models an important step before applying a forecasting model is to identify the break positions, isolate the relevant part of the time series, and use this part only for training. For this purpose, many tests for structural breaks have been proposed. Davis et al. (2006) proposed the Auto-PARM procedure which adopted the minimum description length (MDL) principle [7]. F-tests for structural breaks with a known break point are widely used in the literature. Brown, Durbin and Evans (1975) derived Cusum and Cusum Squared tests that are also applicable when the time of the break is unknown. More recently, contributions by Ploberger, Kramer and Kontrus (1989), Hansen (1992), Andrews (1993), Inclan and Tiao (1994), Andrews and Ploberger (1996) and Chu, Stinchcombe and White (1996) have extended tests for the presence of breaks to account for heteroskedasticity and dynamics. Methods for estimating the size and timing of multiple break points have also been developed, see Bai and Perron (1998, 2003) and Altissimo and Corradi (2003) [6].

The problem with these approaches is that finding the number and the timing of the breaks and the duration of each is a very difficult problem, and often false alarms indicate spurious breaks, while missing the real breaks. The ramifications of a poorly estimated break can be grave, for example leading to wrongly dispensing of a large portion of the training set. In this study we attempted to overcome the difficulties of finding and testing for structural breaks in time series by making use of the power of the forecast combination concept. Forecast combination, where a number of models are designed and their forecasts are combined, has established itself as a potent concept that mostly leads to improved forecast accuracy [9]. In this work we propose to train forecasting models on chunks of the time series with varying training set sizes and then combine the forecasts. This means that we apply the training and the forecasting on multiple historical portions of the time series with varying length. Subsequently we use well-tested strategies for selecting the best few and combining their forecasts.

The outline of the paper is as follows. Section 2 briefly describes the model used and a quick overview on some combination functions used and introduces the proposed new training setups. Section 3 explains the experimental and implementation setup – Data sets, benchmarks and results. Section 4 states the conclusion from the study.

2 Proposed Approach

2.1 The Proposed Method

Our proposed method is based on dividing the time series into multiple training periods. For each training period we train a forecasting model, and at the end combine the results. The main motivation behind this idea is that older parts of the training set might not be very relevant, or some time series could have some structural breaks. It might therefore be better to choose a training set size suitable to each time series so that we avoid using too large a training set (which will tend to use older irrelevant data) and too small a training set data (which results in the forecasts having large statistical error).

Our proposed training setup is based on training the model on the time series starting from T to the last actual point then going back starting from T-k, T-2k ... till the

starting point of the time series (As shown in Fig. 1). Thus, there will be about T/k training sets, each starting from T-ik and ending at the last point of the series, where i is some integer.

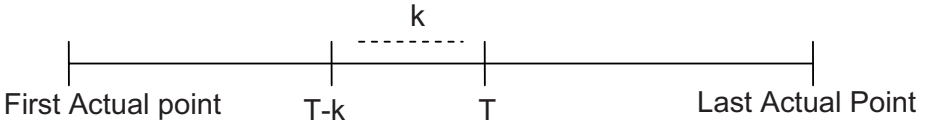


Fig. 1. Proposed Training Setups

We apply five new different strategies combining the forecasts after the training. In the first strategy the final forecast is the combination of ALL the forecasts for all models using average rule (using simple average, see Section 2.3). But in the second strategy, the final forecasts are determined by taking the first two models based on the validation data set over the training periods and combine them only if the error difference is within a given threshold else select the best model. And the third, the final forecasts are determined by taking the first three models and combine them only if the error difference between them all is within a threshold else select the best model. The fourth strategy is the same as the third but we always combine the best three models using the rank based weighting (see Section 2.3). Finally the fifth one is select the best model over all the models.

2.2 Time Series Forecasting Models

In a recent study [12] a large scale empirical comparison of eight different computational intelligence models for time series prediction was conducted. The study revealed that the standard multilayer perceptron neural network (MLP) and the Gaussian process regression (GPR) are the best models to consider. In addition the combination of MLP/GPR led to top ranks in two recent time series forecasting competitions [13], [9]. So we used these models as forecasting methods. In addition, we used exponential smoothing, as it has established itself as a leading method in the statistical literature. The description of the models is given below, with more emphasis on GPR because it is a relatively new and unknown method in the time series forecasting field.

2.2.1 Multi-layer Perceptron

The multilayer perceptron is based on a layered architecture whereby a weighted sum operation is followed by application of a nonlinear squashing function. Specifically, the MLP output is given as follows:

$$\hat{y} = v_0 + \sum_{j=1}^{NH} v_j g(w_j^T x') \tag{1}$$

Where x' is the input vector x augmented with 1, i.e. $x' = (1; x^T)^T$, w_i is the weight vector for j^{th} hidden node, v^0; v^1; v^2:::v^{NH} are the weights for the output node, and $\hat{y}$ is

the network output. The function ***g*** represents the hidden node output, and it is given in terms of a squashing function, in our study for example we use the logistic function: $g(u) = \frac{1}{1+\exp(-u)}$.

2.2.2 Gaussian Processes Regression

Gaussian process regression (GPR) has many desirable properties, such as ease of obtaining and expressing uncertainty in predictions, the ability to capture a wide variety of behavior through a simple parameterization, and a natural Bayesian interpretation. Because of this, GPR has been suggested as replacements for supervised neural networks in non-linear regression, and have been extended to handle classification tasks[8]. .The ability in GPR to fit arbitrary functions or surfaces is its nonparametric flexibility. Gaussian process regression models are constructed from classical statistical models by replacing latent functions of parametric form (e.g. linear functions, truncated Fourier or Wavelet expansions, multi-layer perceptrons) by random processes with Gaussian prior. The posterior distribution of a to-be-predicted function value can then be obtained using the assumed prior distribution –normal distribution- by applying simple probability manipulations.

Let x_i be the input vector for training data point i, with corresponding observed response (or target output) y_i. Arrange the vectors x_i in a matrix X, and arrange the y_i's in a vector y. The observed response y_i equals the underlying function value f_i plus some random error term ϱ_i (assume it is zero-mean normal with variance equal to σ^2_n). The function values f_i are smoothed versions of the observed responses y_i and are the inherent function values to be estimated. Also arrange the f_i's in a vector f.

Some multivariate normal prior is attached to the vector f to enforce smoothness:

$$f \sim N(0, V(X,X)) \tag{2}$$

Where V (X,X) denotes the covariance matrix between the function values of the different training data points, with the $(i, j)^{th}$ element of V (X,X) being V (x_i, x_j), the covariance between the function values of training data points i and j. To guarantee smoothness, the covariance function is taken as a monotonically decreasing function g of the distance between x_i and x_j:

$$V(x_i, x_j) = g(\|x_i - x_j\|) \tag{3}$$

The role of the prior is to impose smoothness of the solution (very smooth solutions are favored by the selected prior). Then, the observed responses are factored in, to obtain the posterior estimate of the function values. Given an input vector x_*, the prediction y_* can be derived using some Bayes rule manipulations as:

$$\dot{y}_* = V(x_*,X)[V(X,X) + \sigma^2_n I]^{-1} y \tag{4}$$

For Gaussian process regression we have three key parameters that control the function smoothness: σ_n (the standard deviation of the error terms ϱ_i), σ_f and the α (the parameters that pertain to the covariance function)[9].

2.2.3 Exponential Smoothing Regression

Exponential smoothing is a procedure for continually revising a forecast in the light of more recent experience. The Holt's exponential smoothing model is an exponential smoothing model with a trend forecasting. It is based on estimating smoothed versions of the level and the trend of the time series. Then, the level plus the trend is extrapolated forward to obtain the forecast. We apply the version of Holt's exponential smoothing based on maximum likelihood that is proposed by Andrawis and Atiya [10]. In this approach, both level and trend smoothing constants, and initial level and initial trend are obtained using the maximum likelihood approach [11].

2.3 Combination Functions

Suppose we have N models and x_i is the forecast output of model i. There are several possible forecast combination methods. We use the following two methods: the simple average or equal weights, defined such that the final combined output is given by:

$$\hat{y} = 1/N \sum_{i=1}^{N} x_i \tag{5}$$

The second one is the rank based weighting. In the rank based weighting the weights are proportional to the inverse of its performance rank (e.g. the forecast performance in some validation time period). This method is expected to be more robust and less sensitive to outliers [11]. The output of each model is weighted by w_i such that:

$$w_i = \frac{1/R_i}{\sum_{j=1}^{N} 1/R_j} \tag{6}$$

R_i is the rank of model(i).

3 Implementation

3.1 Data Sets

We considered the problem of forecasting inbound tourism demand for Egypt. Specifically, we consider the monthly tourist numbers originating from 36 major source countries and the number of nights the tourists spent in Egypt during the period from 1993 to 2007, in addition to the aggregated tourism numbers/nights (aggregated over all source countries). So, we have 74 time series spanning the period starting from 1993. Tourism is one of major economic sectors in Egypt, and it is the fastest growing sector. Decision makers need to have an accurate estimate of future tourism demand.

The forecast horizon is 6 months, and we used a two year validation period. The validation period is used for model selection and some parameter tuning. In addition, the validation data set is used as a basis for selecting the best two or three models as

used in the proposed forecast combination strategies. We used two different snapshots of the data sets. In one of them the last actual data point is November 2008. In the other snapshot the last data point is June 2009. We applied some preprocessing, such as a deseasonalization step and a log transformation. After the forecasting is performed, we unwind these preprocessing steps to obtain the final forecast.

The rationale for applying the proposed strategies on these data sets is that tourism numbers can suffer from structural breaks. For example, the economic downturn of late 2008 affected tourism considerably. Terrorism attacks produce a significant break in the time series. Changing preferences, such as novel tourist source countries change the dynamics of the time series. For example, Russia and eastern block countries increased their contribution to inbound tourism for Egypt in the last 5 to 10 years.

3.2 Benchmark

We compare the new five strategies with a benchmark system that has only two training periods – From Jan 1993 till the end and from Jan 1998 till the end, whereby the best is selected (among the model trained on 1993 till end and 1998 till end). The same forecasting models are used for the new strategy, as well as for the benchmark. The models are the following four: MLP, GPR, MLP/GPR ensemble and exponential smoothing. The same preprocessing and postprocessing has been applied on the two as well. After training and validation, a selection and forecast combination step is applied for the benchmark. The selection and combination rule states "select the best two models according to the validation set and combine them using the simple average if the error difference between them is within a threshold else select the best model." The rationale for using this model as a benchmark for comparison is that in previous work this is the combination that yielded best results for the tourism forecasting problem. Our goal is now to improve upon this previously optimized strategy.

3.3 Results

We use the Symmetric Mean Absolute Percentage Error, SMAPE as the error measure, as described by the equation

$$SMAPE = \frac{1}{M}\sum_{m=1}^{M}\frac{\left|\hat{y}_m - y_m\right|}{(\left|\hat{y}_m\right| + \left|y_m\right|)/2} \tag{7}$$

(y_m is the target output and $\hat{y}_m$ is the prediction). SMAPE has several advantages in that it is a relative measure (its range is from 0 to 200%). We used 2 years (24 points) of validation data set. We compute the average SMAPE for each time series using a six-month rolling forecast covering the period of one year (12 points), as an out of sample test set. Then the average SMAPE is taken over the all 74 time series. We compare the performance of the five strategies over the performance of the benchmark system. Table 1 compares the error percentage between the new strategies and the benchmark.

Table 1. SMAPE (%) for different strategies compared to Benchmark

	November 2008	June 2009
First Strategy	23.04	22.44
Second Strategy	19.86	20.46
Third Strategy	19.96	20.53
Fourth Strategy	19.86	20.46
Fifth Strategy	19.85	20.50
Benchmark	20.23	20.79

3.4 Discussion

By observing Table 1, one can see that the second strategy, the fourth strategy and the fifth strategy are the best strategies, giving almost identical performance. Following these comes the third strategy, with minor differences from the top three strategies. All four strategies considerably beat the benchmark and the first strategy. One can summarize the findings in the following:

1. Selecting and combining the top one, two, or three models is considerable better than combining all models. Thus "selective combination" works better than "combining all".
2. Selecting models based on best training set sizes is considerably better than taking the whole historical data as training set. This suggests that at least for these type of data sets the time series dynamics changes with time or they possess some breaks.
3. For many time series applications, especially economic time series, it is recommended to carefully analyze the data set and use selective subset(s) for training.

By observing the best models that Strategies 2, 4, and 5 selected, we can have an idea about whether and where structural breaks occur (if any). We observed the following: The best training set periods started for most countries during the times 1993-1999. There is a large number that starts in 1998-1999. This can be explained as follows; in November 1997 a major brutal terrorist attack occurred in Luxor, Egypt (a number of scattered other attacks occurred also during the period from 1993 to 1997), This severe terrorist attack in 1997 had a significant dampening effect on tourism to Egypt. Only, one or two years later tourism somewhat recovered. It is not surprising that the model selected the training set to start past this break in 1997.

As an example of how the selection of the training set period affects the performance, refer to Figure 2. It shows the SMAPE error measure as a function of the start year of the training set for one of the time series (France, number of tourist nights). The SMAPE here is measured on the two-year validation period. One can see that the best starting year is 1998. This is consistent with the break that occurred in 1997 due to the Luxor terrorist attack

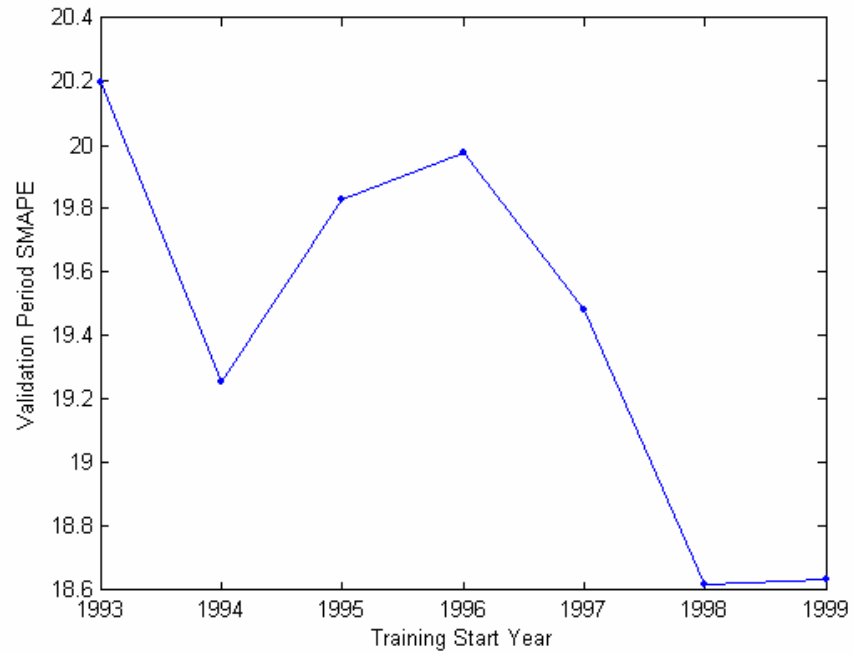


Fig. 2. The validation period SMAPE error measure as a function of the start year of the training set for the France number of tourist nights time series

4 Conclusion

In this paper we presented ideas of using forecast combination to handle time varying conditions and breaks in time series. This is to overcome the difficulties of detecting structural breaks for time series forecasting application. The application we focused on is forecasting tourist demands in Egypt. We observed that this strategy has merit and leads to improved results compared to the benchmark.

Acknowledgement

This work is part of the Cross-Industry Data Mining track within the Egyptian Data Mining and Computer Modeling Center of Excellence. We would like to acknowledge the cooperation of the Egyptian Ministry of Tourism MoT.

References

1. Gheyas, I.A., Smith, L.S.: A Neural Network Approach to Time Series Forecasting. Presented at the 2009 International Conference of Computational Statistics and Data Engineering (ICCSDE), part of the World Congress in Engineering 2009, London (2009)

2. Lai, K.K., Yu, L., Wang, S., Wei, H.: A novel nonlinear neural network ensemble model for financial time series forecasting. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J., et al. (eds.) ICCS 2006. LNCS, vol. 3991, pp. 790–793. Springer, Heidelberg (2006)
3. Yao, J.T., Tan, C.L.: A study on training criteria for financial time series forecasting. In: Proceedings of International Conference on Neural Information Processing, Shanghai, China, November 2001, pp. 772–777 (2001)
4. Liu, Z., Ren, G., Shi, X.: Research of chaotic SVM with Incorporated Intelligence Algorithm forecasting model in SCM. International Journal of Computer Science and Network Security, IJCSNS 6(10), 136–141 (2006)
5. Preston, D., Protopapas, P., Brodley, C.E.: Event Discovery in Time Series. In: SDM 2009, pp. 61–72 (2009)
6. Pesaran, H., Pettenuzzo, D., Timmermann, A.: Forecasting Time Series Subject To Multiple Structural Breaks. Review of Economic Studies 73(4), 1057–1084 (2006)
7. Cho, H., Fryzlewicz, P.: Multiscale breakpoint detection in piecewise stationary AR models. In: Proceedings of IASC 2008, Yokohama, Japan, December 5-8 (2008)
8. Boyle, P., Frean, M.: Dependent gaussian processes. In: Advances in Neural Information Processing Systems, vol. 17, pp. 217–224. MIT Press, Cambridge
9. Andrawis, R.R., Atiya, A., El-Shishiny, H.: Forecast combination model using computational intelligence/linear models for the nn5 time series forecasting competition (2008), http://www.neural-forecasting-competition.com/results.htm
10. Andrawis, R.R., Atiya, A.F.: A new Bayesian formulation for Holts exponential smoothing. Journal of Forecasting (April 2009)
11. Andrawis, R., Atiya, A.F., El-Shishiny, H.: Combination of long Term and short term forecasts, with application to tourism demand forecasting, conditionally accepted. International Journal of Forecasting (2009)
12. Ahmed, N.K., Atiya, A., Gayar, N.E., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. Accepted for publication in Econometric Reviews (2009)
13. Ahmed, N., Atiya, A., Gayar, N.E., El-Shishiny, H.: A combined neural network/gaussian process regression time series forecasting system for the nn3 competition (2007), http://www.neural-forecasting-competition.com/NN3/results.htm

A Multiple Classifier System for Classification of LIDAR Remote Sensing Data Using Multi-class SVM

Farhad Samadzadegan, Behnaz Bigdeli, and Pouria Ramzi

Depertment Of Geomatics Engineering, Faculy of Engineering, University of Tehran, Tehran, Iran
{samadz,bigdeli,pramzi}@ut.ac.ir

Abstract. Rapid advances in remote sensing sensor technology have made it recently possible to collect new dense 3D data like Light Detection And Ranging (LIDAR). One of the challenging issues about LIDAR data is classification of these data for identification of different objects in urban area like building, road, and tree. Regarding to complexities of objects in urban area and disability of LIDAR data to collect the radiometric information of surface, traditional classifiers have low level of performance in classification of LIDAR data. Combining classifiers is an established concept that it used for improvement of classification results. In this paper we propose a classifier fusion system scheme based on Support Vector Machine (SVM) for classification of LIDAR data. Different SVMs are trained on the best different subset of features that are proper for object extraction in LIDAR data and chosen by RANSAC as feature selection method. In this article, two multiclass SVM methods known as one-against-one and one-against-all are investigated for classification of LIDAR data and then final decision is achieved by Majority Voting method. The results confirm that established method on LIDAR data has improved accuracy of classification. It is also demonstrated that one-against-all results better accuracy comparing to one-against-one although it is much more time consuming.

Keywords: LIDAR data, SVM, Classification, Classifier Fusion, Urban Area.

1 Introduction

Remotely sensed data has been widely used to land cover classification and object extraction [1, 2]. Light Detection And Ranging (LIDAR) is one of the recent remote sensing technologies that is widely used for Digital Terrain Model (DTM) data collection and also for other studies including 3D extraction, urban management, atmospheric studies, and so on [2, 3]. Comparing to other remote sensing data sources, LIDAR has its advantages such as acquisition of very dense data in a short period of time. LIDAR data contains plenty of scene information, from which most ground features such as roads, buildings and trees are discernible. More recently, advancements in LIDAR enabled the acquisition of dense point clouds. Major benefits of this technique are its high level of automation during data capturing and its spatial resolution. With point densities of up to several points per square meter, LIDAR data has become a valuable additional source for the reconstruction of different urban objects [2].

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 254–263, 2010.

Automatic extraction of objects such as building, tree and road from LIDAR in complex area is a challenging research topic in pattern recognition and remote sensing studies [2, 3]. This topic is still very limited in the extraction of urban objects due to the complexity of urbane scene and due to producing similar information by LIDAR for different objects.

Several urban classification methods have been proposed for object extraction from LIDAR data [4]. These methods are usually based on a feature extraction step followed by a classification algorithm. Because of high complexity of LIDAR data, single classification methods couldn't achieve more accurate results. As a consequence, the newer concept proposed to use several approaches and try to take advantages of the strengths of each classifier. This concept is known as Decision Fusion. Decision fusion can be defined as the process of fusing information from several individual data sources after each data source has undergone a preliminary classification.

Multiple Classifier System (MCS) is an established method for fusion of different decisions that making by a group of classifiers. Nowadays, classifier fusion is used in pattern recognition and object extraction in the hope of increasing the overall accuracy [5]. The performance of classifier fusion essentially depends on the accuracy and power of single classifiers. One of the state-of-the-art classification methods which has been widely used in remote sensing is Support Vector Machine (SVM) [6]. Recently, SVM is used in groups of research in pattern recognition like object extraction, medical research like cancer diagnosis and handwriting recognition [8, 9].

SVM by itself is a binary classification but in some researches like remote sensing or pattern recognition, we usually have more than two classes. Multi-class SVM is the solution for this problem which is has been utilized in some researches [7, 10, and 12].

In this paper, we introduce a classifier fusion method for classification of LIDAR data in an urban area. The key idea in this paper relies on multi-class SVM classifiers that are trained on different subsets of features selected by RANSAC as feature selection method. In the last step, final decision is obtained by fusing the results of SVM classifiers using weighted majority voting.

2 Multi-class SVM Based Classifier Fusion

The ultimate goal of most traditional methods in pattern recognition is to achieve the best possible classification performance for recognition of different objects. This objective traditionally led to the development of newer concept. The idea is not to rely on a single decision making scheme. Instead, all the designs, or their subset, are used for decision making by combining their individual opinions to derive a consensus decision [5]. Combining classifiers is an established research area shared between statistical pattern recognition and machine learning. It is variously known as committees of learners, mixtures of experts, classifier ensembles, multiple classifier systems, consensus theory, etc [5, 11]. The performance of classifier fusion essentially depends on the accuracy and power of single classifiers. As in this paper SVM is used as base classifier in our classifier fusion scheme, we provide a short introduction about this concept here.

SVM belong to the general category of kernel methods. A kernel method is an algorithm that depends on the data only through dot-products. SVM finds an optimal separating hyperplane in the feature space and uses a regularization parameter to control its model complexity and training error. SVMs are an example of a linear two-class classifier and it can only take two values: ± 1. For a remote sensing application, several classes are usually of interest. One solution for this difficulty is to split the problem into a set of binary classification before combining them. The one-against-one and the one-against-all are the two most popular strategies in this category [10, 11].

One-against-one is the method that calculates each possible pair of classes of a binary classifier. Each classifier is trained on a subset of training examples of the two involved classes. In this method, all N (N-1)/2 binary classifications are combined to estimate the final output. Final output is then created by a majority voting scheme. This approach is suitable for problem with larg amount of data [10, 12].

The most important problem caused by this method is the existence of unclassifiable regions which is able to be solved using one-against-all technique. For an N-class problem, the one-against-all method constructs N SVM models (one SVM per class), which is trained to distinguish the samples of one class from samples of all remaining classes [12]. In this method, the i^{th} SVM is trained using all the learning examples in the i^{th} class with positive labels and the others with negative labels and finally, N hyperplanes are obtained. If ith conventional decision function that classify class i and the remaining class be

$$D_i(X) = w_i^t X + b_i \tag{1}$$

The vector w is known as the weight vector, b is called the bias and D is rhe hyperplane that divids the space into two regions. The hyperplane $D_i(x) = 0$ forms the optimal separating hyperplane and if the training data are linearly separable, the support vectors belonging to class i, satisfy $D_i(x) = 1$ and those belonging to the remaining classes satisfy $D_i(x) = -1$. For the conventional SVM, if for the input vector X, the below equation

$$D_i(X) \succ 0 \tag{2}$$

is satisfied for one i, X is then classified into class i. Since only the sign of $D_i(x)$ is used, this decision is discrete. If (2) is satisfied for plural i, or there will be no i to satisfy this relation, X will be unclassifiable region. To avoid this, datum X is classified into a class which is determined by equation (3):

$$\arg\max_i D_i(X) \tag{3}$$

In this method, the continuous value of the decision function determines classification and as is seen, this decision is continuous.

The use of SVMs as component classifiers in classifier fusion or ensemble is reported to improve the results of classification in some researches. In the field of classifier ensemble, AdaBoost with SVM as component classifiers is introduced to

achieve better results by Li and his colleagues who have utilized SVMs with RBF kernel as a component classifier in AdaBoost algorithm [13]. Lienemann has introduced an ensemble classification system based on SVM and used this algorithm in pharmacology. In his report, final classification results are obtained by voting [14].

Kotez and Morsdort have used SVM based classifier fusion on LIDAR and spectrometer images. Single SVM are applied on two types of data and then the outputs of SVMs are fused. This method is used in land cover classification in the context of forest fire management [15]. Reiter and Rigoll have implemented SVM as base classifier in classifier fusion for segmentation and classification of meeting events [16]. Min and Hong have discussed potential of SVM based classifier fusion for improvement of fingerprint classification. The results in this research exhibit the feasibility and validity of fingerprints classification are increased. This method is also reported to overcome the problem of ambiguous fingerprints. They have used one-against-all SVMs as base classifiers and results are then fused by Decision Template [17].

3 Proposed Method

In this paper a Multi-class SVM based classifier fusion is applied on LIDAR data. The classification process is initiated by feature extraction operation and then two one-against-one and one-against-all SVM methods are implemented on the dataset. Fig1, illustrates the general structure of proposed methodology which contains four main steps.

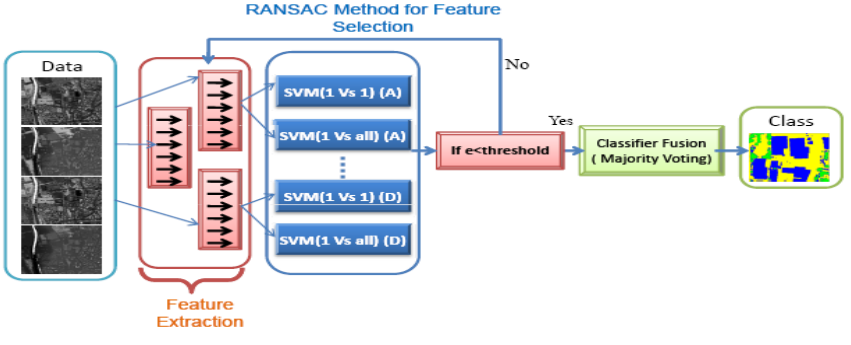


Fig. 1. Proposed method for SVM-based classifier fusion

Feature Extraction. The first step in every classification process is to extract proper features from data set. These features must contain useful information to discriminate between different regions of the surface. In our experiment, we have used different features extracted from two types of LIDAR data (Range data and Intensity data). The normalized difference of the first and last pulse range images (NDI) is usually used as a major feature band for discrimination of the vegetation pixels from the others [18].

The morphological Opening operator is also utilized to filter elevation space. This operator with a flat structuring element eliminates the trend surface of the terrain [19]. The Opening operation is defined by:

$$A \circ B = (A \Theta B) \oplus B \tag{4}$$

where $A \oplus B$ is the morphological Dilation and $A \ominus B$ is the morphological Erosion of set A with structure element B. All types of features used in this research are introduced in Table 1.

Table 1. Different features that used on LIDAR data

Name	Formulation
First Pulse Intensity	FPI
Last Pulse Intensity	LPI
First Pulse Range	FPR
Last Pulse Range	LPR
NDI	$\frac{FPR - LPR}{FPR + LPR}$
Opening	$A \circ B = (A \Theta B) \oplus B$
Mean	$Mean\_i = \sum_{i,j=0}^{N-1} i \times P(i,j)$
Entropy	$\sum_{i,j=0}^{N-1} P_{i,j} \times (-\ln P_{i,j})$
Standard Deviation	$\text{var}\, iance\_i = \sum_{i,j=0}^{N-1} P(i,j) \times (i - Mean\_i)^2$
Homogeneity	$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1 + (i+j)^2}$

In features of Table 1, N is the number of grey levels and P is the normalized symmetric co-occurrence matrix of dimension N x N. V is the normalized grey level difference vector of dimension N.

SVM classification. In our proposed method, we choose four subsets of these features to produce different classifiers. We have applied both one-against-one and one-against-all methods on these feature subsets. Consequently 8 SVM classifiers are prepared to be used in classifier fusion step. All of these classifiers are trained on the same training dataset.

Feature subset Selection. We have used RANSAC (RANdom SAmple Consensus) method for our subset feature selection. It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. We have used this random selection technique for choosing best subsets of features [20].

Classifier fusion. The final decision in our research is achieved by weighted Majority Voting that it applied on a collection of single SVM classification results. Single SVMs

are then compared to the results of fusion in terms of accuracy achieved by them. Overall accuracy of each classifier is used as its weight in weighted Voting method.

4 Accuracy Assessment

We have used error matrices of classification results as main evaluation method of interpretation the quality of each classification method. Each column in this matrix indicates the instances in a predicted class and each row represents the instances in an actual class. All the diagonal variants refer to the correct interpreted numbers of different classes found in reality. Some measures can be derived from the error matrix, such as producer accuracy, user accuracy and overall accuracy [21]. Producer Accuracy (PA) is the probability that a sampled unit in the image is in that particular class. User Accuracy (UA) is the probability that a certain reference class has also been labeled that class. Producer accuracy and user accuracy measures of each class indicate the interpretability of each feature class. We can see the producer accuracy and user accuracy of all the classes in the measures of "producer overall accuracy" and "user overall accuracy".

$$PA_i = \left[\frac{N_{i,i}}{N_{.i}}\right] * 100\% \quad , \quad \mathrm{UA_i} = \left[\frac{N_{i,i}}{N_{i.}}\right] * 100\% \tag{5}$$

Where

$N_{i,j}$: (i, j)th entry in confusion matrix

$N_{i..}$: the sum of all columns for row i

$N_{.j}$: the sum of all rows for column i

"Overall accuracy" considers all the producer accuracy and user accuracy of all the feature classes. Overall accuracy yields one number of the whole error matrix. It's the sum of correctly classified samples divided by the total sample number from user set and reference set [21].

$$OA = \frac{\sum_{i=1}^{k} N_{i,i}}{\left[\sum_{i=1}^{k} N_{.i} + \sum_{i=1}^{k} N_{i.}\right]} * 100\% \tag{6}$$

5 Experiment and Results

5.1 Data Set

The LIDAR remote sensing data with four popular bands, first pulse intensity, last pulse intensity, first pulse range and last pulse range is classified by our proposed method based on SVMs. This sample of LIDAR data is an urban area recorded from city of Castrop-Rauxel which is located in the west of Germany.

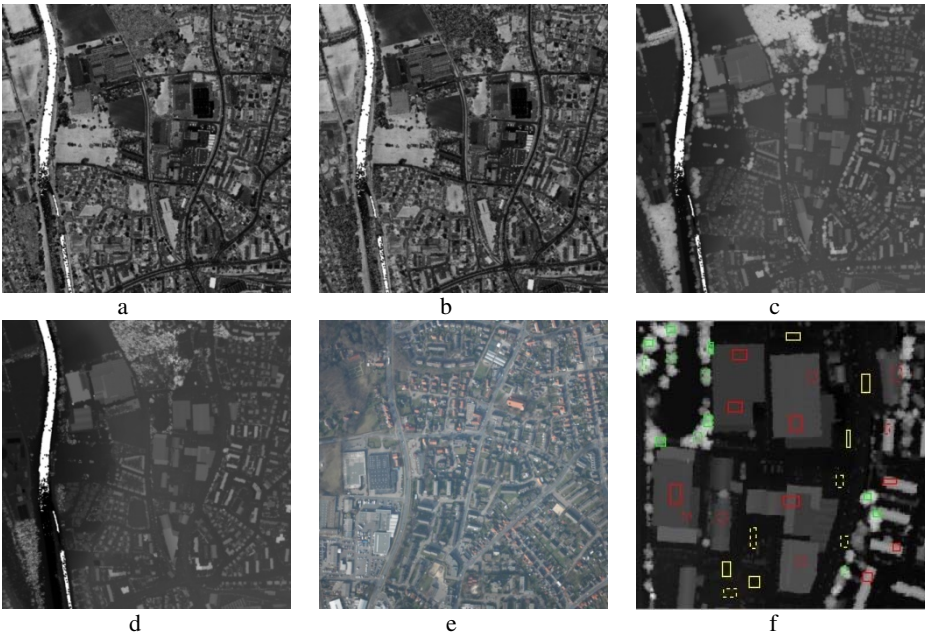


Fig. 2. Data set consist of a) First pulse intensity, b) Last pulse intensity, c) First pulse range, d) Last Pulse range, e) Image, f) Train and test data from selected area for Tree (green), Building (red) and Ground (yellow). Dashed lines demonstrate test data and continuous ones demonstrate training data selected from dataset.

This dataset has enough complexity in urban area for evaluating our proposed method. The LIDAR data is classified into three main classes in urban areas: building, tree and ground. Table.2 illustrates the number of training and test samples selected for each class.

Table 2. Information of training and test sample of each class

Class	Number of training sample	Number of test sample
Tree	510	420
Building	1426	672
Ground	672	564

5.2 Results

In the first step of experiment, we need to produce different SVMs for fusion. For this purpose, we apply one-against-one and one-against-all SVM methods on the four different subsets of features selected by RANSAC. Consequently, we apply two SVM methods on our four subsets of features to produce 8 classifiers. Error matrix and overall accuracy of each classifier (SVM1-1(A), SVM1-1(B), SVM1-1(C),

SVM1-1(D), SVM1-all(A), SVM1-all(B), SVM1-all(C), SVM1-all(D)) are provided in table 3. In this table, A, B, C and D are the best subsets of selected features selected by RANSAC method from all the subsets of features.

Table 3. Confusion matrix and overall accuracy of 8 single SVM

	Reference Data			
SVM1-1(A)		Tree	Building	Ground
	Tree	408	42	4
	Building	11	587	488
	Ground	1	43	72
	Overall accuracy=0.6443			
	FPI, LPI, NDI			
SVM1-1(B)	Tree	78	23	32
	Building	137	648	8
	Ground	205	1	524
	Overall accuracy=0.7548			
	LPR, Opening, Mean, FPI			
SVM1-1(C)	Tree	177	23	46
	Building	40	648	0
	Ground	203	1	518
	Overall accuracy=0.8109			
	Opening, LPR, Mean, Homogeneity			
SVM1-1(D)	Tree	259	0	0
	Building	157	672	1
	Ground	4	0	563
	Overall accuracy=0.9021			
	FPR, LPR, Entropy			

	Reference Data			
SVM1-all(A)		Tree	Building	Ground
	Tree	406	41	5
	Building	11	587	488
	Ground	3	44	71
	Overall accuracy=0.6425			
	FPI, LPI, NDI			
SVM1-all(B)	Tree	57	17	4
	Building	147	654	8
	Ground	216	1	552
	Overall accuracy=0.7627			
	LPR, Opening, Mean, FPI			
SVM1-all(C)	Tree	153	17	14
	Building	53	654	0
	Ground	214	1	550
	Overall accuracy=0.8194			
	Opening, LPR, Mean, Homogeneity			
SVM1-all(D)	Tree	264	0	0
	Building	152	672	2
	Ground	4	0	562
	Overall accuracy=0.9046			
	FPR, LPR, Entropy			

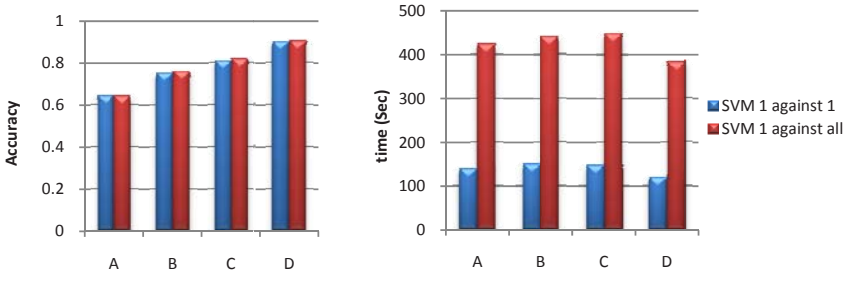


Fig. 3. Comparison of overall accuracy and time between one-against-one and one-against-all

In addition, table 3 indicates that in our utilized dataset, one-against-all method yields better overall accuracy comparing to one-against-one. Figure 3 simply demonstrates our mentioned conclusion. The overall accuracies of eight classifiers are compared here. In addition, in this figure, the times needed for each of the methods are

compared. In this chart, we can observe one-against-one is much faster than one-against all method. Considering this fact, in applications using large amount of data like remote sensing, one-against-one method is preferred. In the last step of our experiment, we fuse the results of 8 SVM classifiers to obtain better classification results. Table 4 shows confusion matrix and overall accuracy of our utilized classifier fusion technique.

Table 4. Confusion matrix and overall accuracy of classifier fusion result

		Reference Data		
Fusion Results		Tree	Building	Ground
	Tree	292	0	0
	Building	124	672	1
	Ground	4	15	563
	Overall accuracy=**0.9166**			

In table 4 it can be seen our utilized fusion algorithm outperforms each of the single classifiers mentioned previously.

6 Conclusion

In this paper we have proposed Multi-class SVM based classifier fusion for classification of LIDAR data in an urban area. We have extracted some standard features from this dataset such as morphological opening, NDDI, Homogeneity, Entropy, … and then selected the best subsets of these features. This selection is done using RANdom SAmple Consensus (RANSAC) which is a random selection technique. Different subsets of features are used in both one-against-one and one-against-all SVM methods, which are two multiclass SVM techniques. We then evaluated and compared these two SVM methods in terms of accuracy and time spent for each one. Experimental results demonstrate one-against-all outperforms one-against-one in accuracy but on the other hand one-against-all is much more time consuming than one-against-all. Based on the mentioned results, one-against-one is preferred to other one for handling larger datasets. In the last step of our research, we tried to improve the classification results by using a classifier fusion scheme. We applied Weighted Majority Voting for the purpose of fusing single SVMs results. It was observed that higher accuracy and performance is achieved by the utilized fusion algorithm comparing to each of the single SVM classifiers.

References

1. Wehr, A., Lohr, U.: Airborne Laser Scanning – An Introduction and Overview. ISPRS Journal of Photogrammetry and Remote Sensing 54, 68–82 (1999)
2. Clode, S., Kootsookos, P., Rottensteiner, F.: The Automatic Extraction of Roads from LIDAR Data. IAPRSIS XXXV-B3, 231–236 (2004)

3. Alharthy, A., Bethel, J.: Automated Road Extraction From LIDAR Data. In: Proceedings of ASPRS, Anchorage, Alaska, unpaginated, CD-ROM (2003)
4. Zhuang, X., Engel, B.A., Xiong, X., Johannsen, C.J.: Analysis of classification results of remotely sensed data and evaluation of classification algorithms. Photogrammetric Engineering and Remote Sensing 61(4), 427–433 (1995)
5. Kuncheva, L.: Combining Pattern Classifiers methods and algorithms. John Wiley&Sons, Inc. Publication, Hoboken (2004)
6. Fauvel, M., Chanussot, J., Benediktsson, J.A.: Decision fusion for the classification of urban remote sensing images. IEEE Trans. Geosci. Remote Sensing (2008) (accepted for publication)
7. Weston, J., Watkins, C.: Multi-class support vector machines, Technical report CSD-TR-98-04 (1998)
8. Naotosi, S.: A comparison of Multi-calss Support Vector Machine Methods for Face Recognition. Department of Electrical and Computer Engineering, The University of Maryland (2007)
9. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning 46(1-3), 389–422 (2002)
10. Weston, J., Watkins, C.: Multi-class support vector machines, Technical report CSD-TR-98-04 (1998)
11. Ruta, D., Gabrys, B.: An overview of classifier fusion methods, computing and information system. University of Paisley (2000)
12. Hsu, C., Lin, C.: A comparison of methods for multi-class support vector machines, Technical report, Department of Computer Science and Information Engineering, National Taiwan University (2001)
13. Liu, Y., Wang, R., Zeng, Y.-S.: An Improvement of one-against-one Method for Multiclass Support Vector Machine. In: 4th International Conference Sciences of Electronic, Technologies of Information and Telecommunications (2007)
14. Li, X., Wang, L., Sung, E.: AdaBoost with SVM-based Component Classifiers. Engineering Application of Artificial Intelligence, 785–791 (2008)
15. Kotez, B., Morsdort, F., Curt, T.: Fusion of Imaging Spectrometer and Lidar Data Using Support Vector Machine for Land Cover Classification in the context of Forest Fire Management. Dept. of remote sensing, University of Zurich, Switzerland (2008)
16. Rieter, S., Rigoll, G.: Segmentation and Classification of Meeting Events Using Muliple Classifier Fusion and Dynamic Programming. Institute of Human-Machine-Communication, Technische Universität München, Germany (2003)
17. Min, J., Hong, J., Cho, S.: Effective Fingerprint Classification for Localized Models of SVM. In: Zhang, D., Jain, A.K. (eds.) ICB 2005. LNCS, vol. 3832, pp. 287–293. Springer, Heidelberg (2005)
18. Katzenbeisser, R.: Technical note on Echo Detection of Laser Data (2003), `http://www.topscan.de/` (accessed: March 2005)
19. Gonzalez, R.C., Woods, R.: Digital Image Processing. Addison-Wesley Publishing, Reading (1993)
20. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model ¯fitting with application to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)
21. Liu, Y., Jin, R., Jain, A.K.: BoostCluster: boosting clustering by pairwise constraints. In: KDD 2007, the 13th International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA (2007)

A Multi-Classifier System for Off-Line Signature Verification Based on Dissimilarity Representation*

Luana Batista, Eric Granger, and Robert Sabourin

Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de technologie supérieure
1100, rue Notre-Dame Ouest, Montreál, QC, H3C 1K3, Canada
lbatista@livia.etsmtl.ca, {eric.granger,robert.sabourin}@etsmtl.ca

Abstract. Although widely used to reduce error rates of difficult pattern recognition problems, multiple classifier systems are not in widespread use in off-line signature verification. In this paper, a two-stage off-line signature verification system based on dissimilarity representation is proposed. In the first stage, a set of discrete HMMs trained with different number of states and/or different codebook sizes is used to calculate similarity measures that populate new feature vectors. In the second stage, these vectors are employed to train a SVM (or an ensemble of SVMs) that provides the final classification. Experiments performed by using a real-world signature verification database (with random, simple and skilled forgeries) indicate that the proposed system can significantly reduce the overall error rates, when compared to a traditional feature-based system using HMMs. Moreover, the use of ensemble of SVMs in the second stage can reduce individual error rates in up to 10%.

1 Introduction

Signature verification (SV) systems seek to authenticate the identity of an individual, based on the analysis of his or her signature, through a process that discriminates a genuine signature from a forgery [13]. SV systems are relevant in many situations where handwritten signatures are currently used, such as cashing checks, transactions with credit cards, and authenticating documents. In off-line SV, signatures are available on sheets of paper, which are later scanned in order to obtain a digital representation. Given a digitized signature, an off-line SV system will perform preprocessing, feature extraction and classification (also called verification) [3].

The utilization of multiple classifier systems (MCS) has been shown to reduce error rates of many challenging pattern recognition problems. However, MCS have received relatively little attention in the off-line SV community [1,4,6,15].

* This research has been supported by the *Fonds Québécois de la Recherche sur la Nature et les Technologies* (FQRNT).

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 264–273, 2010.

By using just the best classifier, it is possible to loose valuable information contained in the other suboptimal classifiers. Moreover, it has been shown that, when a set of R classifiers is averaged, the variance contribution in the bias-variance decomposition decreases by 1/R, resulting in a smaller expected classification error [16]. Classifiers may be combined in parallel by changing (i) the training set, (ii) the input features and (iii) the parameters/architecture of the classifier. Multi-stage approaches, where each classification level receives the results of the previous one, is another way to use multiple classifiers to reduce the complexity of the problem.

Among several well-known classification methods used in off-line SV, the discrete Hidden Markov Model (HMM) [14] – a finite stochastic automata used to model sequences of observations – is known to adapt easily to the dynamic characteristics of the western handwriting [10]. The traditional approach consists in training a HMM only with genuine signatures. Therefore, the decision boundary between the impostor and genuine classes is defined later, by using a validation set that contains samples from both classes. Hence, an input pattern is assigned to the genuine class if its likelihood is greater than the decision threshold.

In contrast to this traditional system, Bicego [5] proposed a classification strategy (applied to the problem of 2D shape recognition) where both the genuine subspace, w_1, and the impostor's subspace, w_2, are modeled. Based on the dissimilarity representation (DR) approach – in which an input pattern is described by its distances with respect to a predetermined set of prototypes [12] –, the strategy consists in using a set of HMMs not as classifiers, but as a way to calculate similarity measures that define a new input feature space. The fact that two sequences O_i and O_j present similar degrees of similarity with respect to several HMMs enforces the hypothesis that O_i and O_j belong to the same class [5].

In this paper, a two-stage off-line SV system inspired by Bicego's DR concept [5] is proposed. Given a set of discrete HMMs trained with different number of states and/or different codebook[1] sizes, a greedy algorithm is employed to select the most representative ones that will be part of the first stage of the system. These HMMs can be viewed as feature extractors used to obtain the vectors of similarities. In the second stage, the vectors of similarities are used to train a SVM (or an ensemble of SVMs) whose objective is to provide the final decision. To analyze the system's performance, an overall ROC curve that takes into account user-specific thresholds is constructed. This curve also allows the system to dynamically select the most suitable solution for a given input pattern. This property can be useful in banking applications, for example, where the decision to use a specific operating point (threshold) may be associated with the value of a check.

Experiments performed with the Brazilian SV database [4] (with random, simple and skilled forgeries), indicate that the proposed system can significantly

[1] A codebook contains a set of symbols, each one associated with a cluster of feature vectors, used to generate sequences of discrete observations in discrete HMM-based systems.

reduce the overall error rates, when compared to a traditional feature-based system that uses a single HMM per writer. The paper is organized as follows. The next section presents the proposed approach. Then, Section 3 describes the experimental methodology and Section 4 presents and discusses the experiments.

2 Proposed System

In this section, a two-stage off-line SV system inspired by Bicego's DR concept [5] is proposed. In the first stage, a set of representative HMMs are used as feature extractors in order to obtain similarity measures (likelihoods) that populate new feature vectors. This idea is formally defined as follows.

Let $w_1 = \{\lambda_1^{(C_1)}, ..., \lambda_R^{(C_1)}\}$ be the set of R representative models of the genuine class C_1; $w_2 = \{\lambda_1^{(C_2)}, ..., \lambda_S^{(C_2)}\}$ be the set of S representative models of the impostor's class C_2; and $\mathcal{M}$ be the vector containing the representative models of both classes, that is, $\mathcal{M} = [w_1 \cup w_2]$. Given a training sequence $O_{trn} \in \{C_1|C_2\}$, its feature vector $\mathcal{D}(O_{trn}, \mathcal{M})$ is composed of the likelihoods computed between O_{trn} and every model in $\mathcal{M}$, that is,

$$\mathcal{D}(O_{trn}, \mathcal{M}) = \begin{bmatrix} P(O_{trn}/\lambda_1^{(C_1)}) \\ \cdots \\ P(O_{trn}/\lambda_R^{(C_1)}) \\ \\ P(O_{trn}/\lambda_1^{(C_2)}) \\ \cdots \\ P(O_{trn}/\lambda_S^{(C_2)}) \end{bmatrix}$$

After applying the same process to all training signatures from C_1 and C_2, the obtained feature vectors are used to train an ensemble of user-specific classifiers[2] (SVMs) in the second stage. During the test phase, the feature vector $\mathcal{D}(O_{tst}, \mathcal{M})$ is calculated for a given input sequence O_{tst}, and then sent to the ensemble of SVMs, which takes the final decision by majority vote. Figure 1 illustrates the proposed system, where three HMMs per subspace are used.

Observe that, if O_{tst} belongs to class C_1, the feature vector $\mathcal{D}(O_{tst}, \mathcal{M})$ should contain bigger values in the first R positions and smaller values in the remaining S positions (the inverse if O_{tst} belongs to class C_2), which allows to discriminate between the classes C_1 and C_2. In a feature-based approach, O_{tst} would be assigned to the class of the most similar model. However, this approach does not use all the information contained in a space of dissimilarities [5].

In order to obtain the most representative models to compose the subspaces w_1 and w_2, a greedy algorithm is used. Starting with an empty subspace, the models are incrementally added until a convergence criterion is reached. Basically, a model λ is chosen if its addition to the subspace minimizes the average error rate (AER) provided by a 1-NN classifier (with Euclidean distance) on the validation set. Algorithm 1 presents more details of this strategy.

[2] In this paper, the term *user-specific classifier* is used to differentiate from systems where a same *global classifier* is shared by all users.

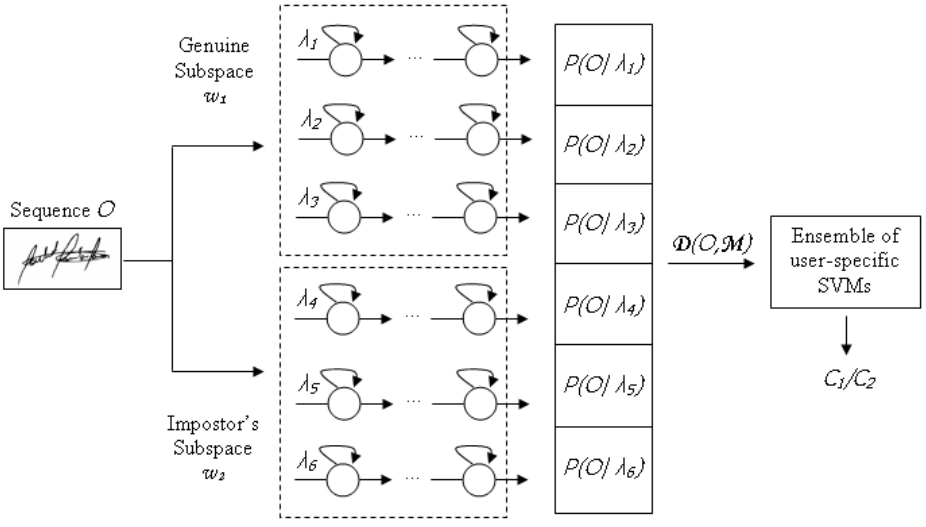


Fig. 1. Diagram of the proposed system

3 Experimental Methodology

The Brazilian SV database [4] is used for proof-of-concept computer simulations. It contains 7920 samples of signatures that were digitized as 8-bit greyscale images over 400X1000 pixels, at resolution of 600 dpi. The signatures were provided by 168 writers and are organized in two sets: the development database (DB_{dev}) and the exploitation database (DB_{exp}). DB_{dev} is composed of 4320 genuine samples supplied by 108 individuals, and it is used for designing codebooks and to train the HMMs that will compose the impostor's subspace, w_2.

DB_{exp} contains 60 writers, each one with 40 samples of genuine signatures, 10 samples of simple forgery and 10 samples of skilled forgery. 20 genuine samples are used for training, 10 genuine samples for validation, and 30 samples for test (10 genuine samples, 10 simple forgeries and 10 skilled forgeries). Moreover, 10 genuine samples are randomly selected from the other 59 writers and used as random forgeries to test the current user-specific classifier. Each writer in DB_{exp} will, therefore, be associated to a genuine subspace, w_1.

The signature images are represented by means of density of pixels, extracted through a grid composed of cells of 16x40 pixels [2]. In order to generate the sequences of observations, a codebook with 35 symbols, denoted as CB_{35}, is employed (for more details regarding CB_{35} construction, see ref. [2]). For each writer from both DB_{dev} and DB_{exp}, a set of HMMs is trained by using the *left-to-right* topology [14] with 20 genuine samples and different number of states; where the maximum number of states is given by the smallest sequence of observations used for training. Therefore, to compose w_1, there are a variable number of available HMMs that depends on the writer's signature size. On the other hand,

Algorithm 1. Selection of representative models.

Inputs:
(i) the validation set $\mathcal{V}$ composed of genuine signatures (C_1) and random forgeries (C_2)
(ii) the sets of available models Φ_1 and Φ_2, representing, respectively, C_1 and C_2
Outputs: the vector of representative models, $\mathcal{M}$

for each class C_i, $i = 1, 2$ **do**
  set $w_i \leftarrow [\,]$;
  set $j \leftarrow 0$; //j represents the number of positions of vector w_i
  repeat
    $j \leftarrow j + 1$;
    find the model λ_k in Φ_i that, when added to $w_i(j)$, provides the smallest AER of a 1-NN classifier using the validation set $\mathcal{V}$;
    remove λ_k from the list of available models Φ_i;
  until AER reaches a minimum value
end for
append, vertically, each $w_i(1..j)$; that is, $\mathcal{M} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

to compose w_2, there are always 3296 available HMMs, taken from the 108 writers in DB_{dev}.

It is assumed that the overall system's performance is measured by an averaged ROC curve obtained from a set of user-specific ROC curves. The chosen averaging method [9] generates an overall ROC curve taking into account user-specific thresholds. At first, the cumulative histogram of random forgery scores of each user i is computed. Then, the similarity scores (thresholds) providing a same value of cumulative frequency, γ, are used to compute the operating points $\{TPR_i(\gamma), FPR_i(\gamma)\}$. Finally, the operating points associated with a same γ are averaged. Note that γ can be viewed as the true negative rate (TNR) and

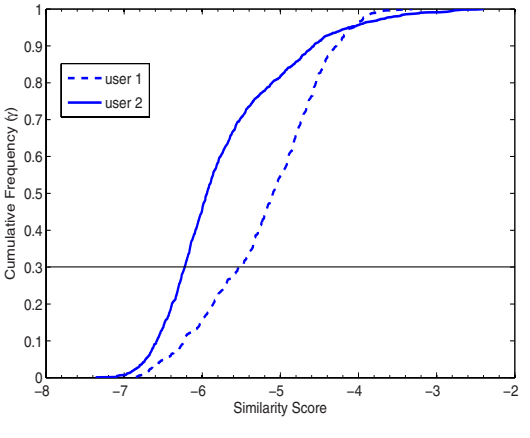


Fig. 2. Cumulative histogram of random forgery scores regarding two different users

that it may be associated with different thresholds. Figure 2 shows an example where the thresholds associated with $\gamma = 0.3$ are different for users 1 and 2, that is $t_{user1}(0.3) \cong -5.6$ and $t_{user2}(0.3) \cong -6.4$.

To measure the system's performance during test, false negative rates (FNR) and false positive rates (FPR) are calculated by using the user-specific thresholds associated to different operating points γ of the averaged ROC curve. The average error rate (AER), also computed for different γ, indicates the total error of the system, where FNR and FPR are averaged taking into account the *a priori* probabilities.

4 Simulation Results

This section presents two sets of experiments performed with the Brazilian SV database [4]. In the first one, the proposed DR-based system is compared to a traditional feature-based system that uses a single HMM per writer. In this case, only one SVM per writer is employed in the second stage of the proposed system. In the second set of experiments, the impact of using ensembles of SVMs in the second stage is investigated.

4.1 DR-Based Approach *vs.* Feature-Based Approach

Given the HMMs trained such as explained in Section 3 and the validation set – which contains 10 genuine signatures taken from DB_{exp} *versus* 1080 (108x10) random forgeries taken from DB_{dev}–, Algorithm 1 was applied to select the most representative models. This process was performed individually for each writer in DB_{exp}, and, on average, 2 models were selected for representing w_1, and 3 models, for representing w_2. Then, still using the validation set, the gridsearch technique with 10-fold cross-validation was employed in order to find the best parameters $\{c, \gamma\}$ of the two-class SVMs ($CSVC$) with RBF kernel [8]. Finally, the selected parameters were used to train a single SVM per writer, with 20 genuine signatures taken from DB_{exp} *versus* 2160 (108x20) random forgeries taken from DB_{dev}.

The averaged ROC curve representing the proposed system (obtained from the validation set) is indicated by the square-dashed line in Figure 3. The circle-dashed curve corresponds to a traditional (feature-based) HMM-based system designed such as described in Introduction. This baseline system uses only density of pixels as features and a single HMM per writer as classifier, where the number of states is selected through the cross-validation procedure described in [10]. Table 1 (a) and (b) present the error rates on test for both systems regarding different operating points (γ). Note that the proposed system provided a reduction in AER from 2.5%, for $\gamma = 1$, up to 9.87%, for $\gamma = 0.95$.

Table 2 shows the results provided by other systems that use the Brazilian SV database [4] and pixel density features. Except in [4], where DR is employed to

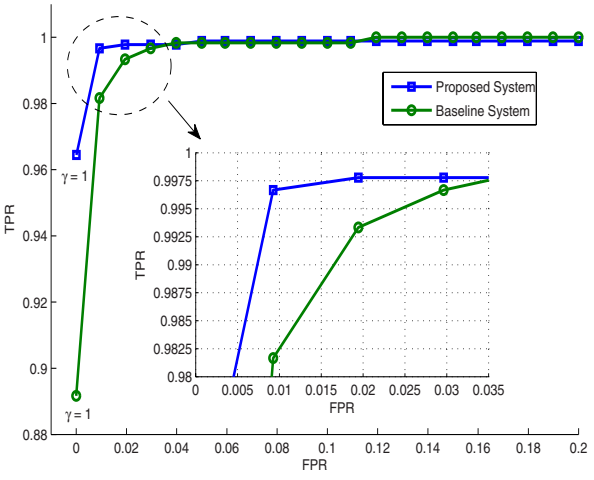


Fig. 3. Averaged ROC curves of baseline and proposed systems

design a global classifier, the refered articles propose feature-based approaches. Note that our system provides the smallest AER (see Table 1 (b), $\gamma = 1$).

4.2 DR-Based Approach Using an Ensemble of SVMs

The experiment presented in this section consisted in analyzing the impact of employing and combining different SVMs per writer in the proposed DR-based system. In order to generate the candidate classifiers, a different user-specific SVM is trained each time that a new model is selected by Algorithm 1. For example, given that w_1 is represented by models (a, b, c) and w_2, by (d, e), six candidate SVMs can be produced by using $w_1 \cup w_2$, that is, (a, d); (a, d, e); (a, b, d); (a, b, d, e); (a, b, c, d) and (a, b, c, d, e).

Once the set of candidate classifiers is obtained, the algorithm ICON [17] is applied in order to incrementally construct the ensemble of SVMs. Like as Algorithm 1, ICON consists in a greedy process that, at each iteration, chooses the classifier that best improves system's performance (on validation data) when added to the current ensemble.

A measure called CI (from Chebishev's inequality) [7,11] is employed to evaluate the ensemble. It is computed as $CI = \sigma(\tau)/\mu(\tau)^2$, where σ and μ denote the variance and the average of the set of margins τ provided by the samples in the validation set, respectively. Given a sample x_i from class C_1, its margin τ_i is given by the difference between the number of votes assigned to the true class C_1, minus the number of votes assigned to class C_2. The ensemble providing the smallest CI value contains the strongest and less correlated classifiers [11].

The overall error rates obtained on test by using majority vote are shown by Table 1 (c). Note that, except for $\gamma = 1$, the improvements were mostly related

Table 1. Overall error rates (%) of baseline and proposed systems on the test data

(a) Baseline system

γ	FNR	FPR_{random}	FPR_{simple}	$FPR_{skilled}$	AER
0.95	0.50	6.83	12.83	68.00	22.04
0.96	0.50	6.00	10.83	64.83	20.54
0.97	0.83	5.67	9.00	60.17	18.92
0.98	1.17	4.00	5.67	52.50	15.83
0.99	2.33	2.67	4.00	42.67	12.92
1	12.67	0.33	1.17	19.83	8.50

(b) Proposed DR-based system with single SVMs

γ	FNR	FPR_{random}	FPR_{simple}	$FPR_{skilled}$	AER
0.95	2.17	3.83	5.17	37.50	12.17
0.96	2.17	3.17	4.83	36.67	11.71
0.97	2.17	2.50	4.50	36.50	11.42
0.98	2.33	2.00	4.00	36.33	11.17
0.99	2.50	1.33	3.33	34.67	10.46
1	16.17	0.00	0.17	7.67	6.00

(c) Proposed DR-based system with ensembles of SVMs

γ	FNR	FPR_{random}	FPR_{simple}	$FPR_{skilled}$	AER
0.95	2.83	2.33	4.50	33.33	10.75
0.96	3.00	1.67	3.67	33.67	10.50
0.97	2.83	1.33	3.67	33.67	10.37
0.98	2.83	1.00	3.50	34.17	10.37
0.99	3.00	1.00	3.50	32.67	10.04
1	13.50	0.00	0.17	8.33	5.50

to $FPRs$. Figure 4 (a) presents the 60 individual $AERs$ for $\gamma = 0.95$. According to this graph, 48.33% of the writers had their $AERs$ on test reduced (in up to 10%) with the use of ensembles – which may indicate a considerable amount of users in a real world application –, while 15% performed better with single SVMs. For the remaining 36.67%, both versions of the system performed equally.

Regarding $\gamma = 1$, only 34 writers were associated to ensembles by algorithm ICON. The remaining 26 writers kept using a single SVM with all models selected by Algorithm 1. In Figure 4 (b), observe that the AER was reduced in 7.5% for writers 4 and 10, in 10% for writer 9, and in 2.5% for writers 21, 26 and 38. Whereas for writer 20, the use of ensembles increased the AER in 2.5%.

Table 2. Error rates (%) provided by other off-line SV systems

Reference	FNR	FPR_{random}	FPR_{simple}	$FPR_{skilled}$	AER
Batista et al. [2]	9.83	0	1.00	20.33	7.79
Bertolini et al. [4]	25.32	3.8	4.48	7.8	10.35
Justino et al. [10]	2.17	1.23	3.17	36.57	7.87

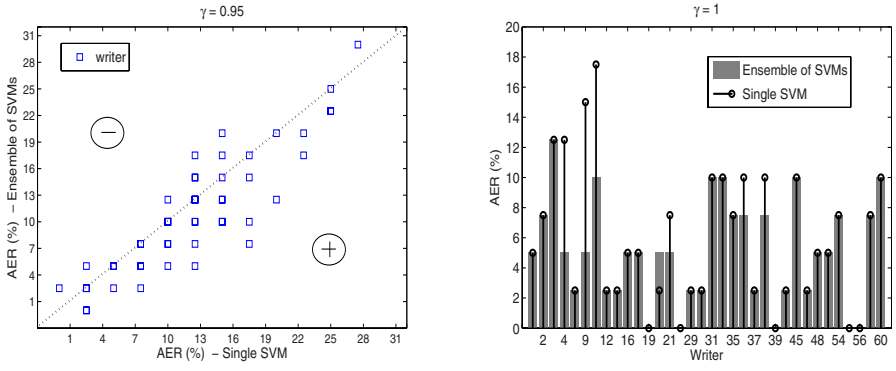


Fig. 4. Individual *AERs* obtained on test for $\gamma = 0.95$ (a) and $\gamma = 1$ (b), before and after using ensemble of SVMs. In (a), note that the writers that had their *AERs* reduced by using ensemble of SVMs are located below the dotted line.

5 Conclusions

In this paper, a two-stage off-line SV system based on DR [5] was proposed. In the first stage, a set of representative HMMs – trained with different number of states – was used to produce similarity measures to form new feature vectors. In the second stage, these vectors were input to one or more SVMs in order to provide the final classification.

When compared to a baseline system that uses a single HMM per writer, the proposed system provides a reduction in *AER* from 2.5%, for $\gamma = 1$, up to 9.87%, for $\gamma = 0.95$. One of the reasons for this improvement is the fact that both genuine and forger subspaces are modeled, which does not occur with traditional (feature-based) systems using HMMs. Moreover, feature-based approaches do not use all the information contained in a similarity space [5]. Finally, the use of ensemble of SVMs can reduce individual error rates in up to 10%.

The proposed approach may require greater computational complexity (training time and memory consumption) than a traditional approach due to the generation of the candidate HMMs and to the selection of the most representative ones. However, once the most representative HMMs are obtained (about 5 per writer in the experiments), all sub-optimal solutions can be discarded. As future work, we intend to employ different codebook sizes and different number of states to generate the set of candidate HMMs.

References

1. Bajaj, R., Chaudhury, S.: Signature verification using multiple neural classifiers. Pattern Recognition 30, 1–7 (1997)
2. Batista, L., Granger, E., Sabourin, R.: Improving performance of hmm-based off-line signature verification systems through a multi-hypothesis approach. International Journal on Document Analysis and Recognition, IJDAR (2009)

3. Batista, L., Rivard, D., Sabourin, R., Granger, E., Maupin, P.: State of the art in off-line signature verification. In: Verma, B., Blumenstein, M. (eds.) Pattern Recognition Technologies and Applications: Recent Advances, 1st edn., IGI Global (2007)
4. Bertolini, D., Oliveira, L., Justino, E., Sabourin, R.: Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. Pattern Recognition 43, 387–396 (2010)
5. Bicego, M., Murino, V., Figueiredo, M.: Similarity-based clustering of sequences using hidden markov models. Pattern Recognition 37(12), 2281–2291 (2004)
6. Blatzakis, H., Papamarkos, N.: A new signature verification technique based on a two-stage neural network classifier. Engineering Applications of Artificial Intelligence 14, 95–103 (2001)
7. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
8. Chang, C., Lin, C.: Libsvm: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
9. Jain, A., Ross, A.: Learning user-specific parameters in a multibiometric system. In: International Conference on Image Processing (ICIP), pp. 57–60 (2002)
10. Justino, E., Bortolozzi, F., Sabourin, R.: Off-line signature verification using hmm for random, simple and skilled forgeries. In: International Conference on Document Analysis and Recognition, pp. 105–110 (2001)
11. Kapp, M., Sabourin, R., Maupin, P.: An empirical study on diversity measures and margin theory for ensembles of classifiers. In: 10th International Conference on Information Fusion (Fusion 2007), pp. 1–8 (2007)
12. Pekalska, E., Paclik, P., Duin, R.: A generalized kernel approach to dissimilarity based classification. Journal of Machine Learning Research 2, 2001 (2002)
13. Plamondon, R.: Progress in Automatic Signature Verification. Word Scientific, Singapore (1994)
14. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. IEEE 77(2), 257–286 (1989)
15. Sansone, C., Vento, M.: Signature verification: Increasing performance by a multi-stage system. Pattern Analysis and Applications 3, 169–181 (2000)
16. Tax, D.: One-class Classification. PhD thesis, TU Delft (2001)
17. Ulas, A., Semerci, M., Yildiz, O., Alpaydin, E.: Incremental construction of classifier and discriminant ensembles. Information Sciences 179(9), 1298–1318 (2009)

A Multi-objective Sequential Ensemble for Cluster Structure Analysis and Visualization and Application to Gene Expression

Noha A. Yousri

Computers and Systems Engineering Department, Faculty of Engineering,
Alexandria University, Egypt
nyousri@alexeng.edu.eg, nyousri@pami.uwaterloo.ca

Abstract. In the presence of huge high dimensional datasets, it is important to investigate and visualize the connectivity of patterns in huge arbitrary shaped clusters. While density or distance-relatedness based clustering algorithms are used to efficiently discover clusters of arbitrary shapes and densities, classical (yet less efficient) clustering algorithms can be used to analyze the internal cluster structure and visualize it. In this work, a sequential ensemble, that uses an efficient distance-relatedness based clustering, "Mitosis", followed by the centre-based K-means algorithm, is proposed. K-means is used to segment the clusters obtained by Mitosis into a number of subclusters. The ensemble is used to reveal the gradual change of patterns when applied to gene expression sets.

Keywords: Clustering, Sequential Ensemble, Multi-Objective Clustering, Density based, Distance-relatedness based, Arbitrary Shaped Clusters, Gene Expression Analysis.

1 Introduction

Ensembles of clustering algorithms have been repeatedly introduced in the literature (a subset of the work done is [1],[2],[3],[4],[5],[6] and others). The aim of combining clustering solutions obtained by different algorithms is to arrive at a consensus clustering using an appropriate voting scheme. However, cluster ensembles' models take several shapes as combining solutions obtained at different parameter settings for the same algorithm, combining solutions obtained from subsets of data, or combining clustering algorithms that use different approaches. The model is selected according to the problem in hand, as handling different parameter settings, large data sets or exploiting variant clustering objectives to arrive at a better clustering solution.

Previous attempts used voting to combine solutions that can be obtained in parallel (independent from each other). These can be termed as *parallel* ensembles. It is important to distinguish such methods from *sequential* ensembles, as presented here, where clustering obtained by one algorithm is further analyzed by another algorithm in a sequential fashion to arrive at a better cluster analysis. These are much simpler than parallel ones and the presented work does not pose into the consensus problem

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 274–283, 2010.

where weighting and voting schemes are the main challenges. However, their importance lies in presenting a multilevel analysis to the same dataset.

The main motivation of the proposed sequential ensemble is to target huge high dimensional datasets that contain large clusters which need further interpretation and analysis. While hierarchical clustering algorithms were previously used to cluster data at different levels of abstraction, yet, they cannot uncover the natural shapes and densities of clusters compared to more recent density and distance-relatedness based algorithms. Besides, using a sequential ensemble would be more efficient for huge datasets, where hierarchical algorithms (at least quadratic in the number of patterns) are prohibitive to use.

Another motivation for the presented work is the issue of the curse of dimensionality, which poses a great challenge in the data analysis process. Visualization of the clustered data is hindered in the presence of very large dimensions. Even the use of PCA (Principal Component Analysis) or SVD (Singular Value Decomposition) [7] to visualize the data in lower dimensions would fail in very high dimensions.

Multiple clustering objectives/criteria can be used in parallel ensembles as an attempt to obtain more valid clustering solutions. Sequential use of different clustering objectives, however, aims at providing a more detailed structure of the clusters obtained, in order to interpret the gradual pattern change in a cluster. These are important especially in the presence of arbitrary shaped clusters, and in general to compare different clusters' structures and outputs of different algorithms.

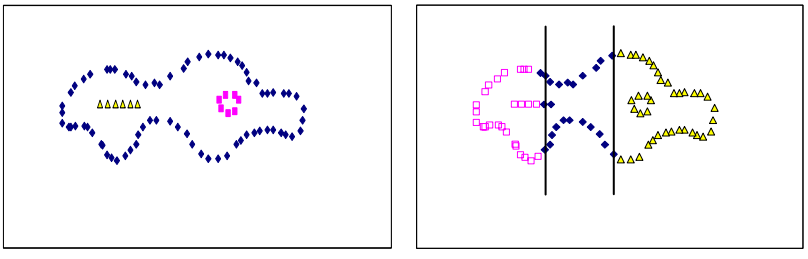
The work presented here proposes to use a sequential ensemble of density-based or distance-relatedness based clustering algorithm as Mitosis [8],[9] followed by a partitioning algorithm as K-means [10] to segment the obtained clusters into a number of subclusters/segments. This proposes the use of two different clustering objectives/criteria: the first aims at tracking natural shapes of clusters by utilizing a density (can be measured by distance characteristics) criterion, and the second aims at forcing segmentation/partitioning of each cluster using a center-based objective. Challenging issues include the selection of the number of segments/subclusters and obtaining an ordering of the segments/subclusters together with the use of an appropriate visualization to arrive at the required objective.

The paper is organized as follows: section 2 gives the background, section 3 presents the proposed work, section 4 gives the experimental results and section5 concludes the paper.

2 Background

Clustering algorithms are based on diverse approaches e.g. partitioning as K-means [10], hierarchical as single, average and complete linkage [10], density-based as DBScan [11], grid-based as DenClue [12], graph-based as Chameleon [13] and ROCK [14] and distance-relatedness based as Mitosis [8], [9]. Of the mentioned algorithms, those which impose a specific cluster structure as globular or spherical shaped clusters and others which use more efficient criteria to uncover the natural cluster shapes, sizes and densities. It is important to focus on the general clustering problem,

assuming the presence of arbitrary shapes, sizes and densities, with the presence of outliers as well. Clustering algorithms as DBScan, Chameleon, and Mitosis, with different clustering criteria, time performance and resulting clustering solutions, have realized the general problem and addressed specific issues of cluster structure variation. Figure 1 shows the difference between the use of density or distance-relatedness based algorithms and center-based or partitioning algorithms as K-means.



(a) (b)

Fig. 1. a) Arbitrary shaped clusters as discovered by density or distance-relatedness based algorithms (as DBScan, Mitosis), (b) Inability of K-means to discover the natural cluster shapes

Huge and high dimensional gene expression data presents a challenging task for unsupervised learning. The target is to discover gene relations and possible gene markers in a gene expression set. A gene expression pattern refers to the behaviour (expression) of gene along a number of tumor samples/conditions (these are the dimensions), or along different time points (as in time series gene expression). Mitosis was previously applied to gene expression data, in [9], [15], [16] and [17], as breast cancer, leukaemia and serum datasets. It has also been combined with outlier and core pattern analysis (as in [16] and [17]) for efficient exploration of gene clusters and gene relations. The importance of using Mitosis with gene expression lies in its ability to discover connectedness of expression patterns which corresponds to the problem of finding arbitrary shaped clusters of different densities.

Visualization of high dimensional data becomes inefficient in many cases, especially in the presence of huge datasets. Visualization can be done using PCA or SVD, where the coefficients of components corresponding to the highest ranked eigen values are selected and plotted, resulting in 2-D and 3-D plots. However, these may produce visually unacceptable results for huge datasets. Thus, the use of efficient clustering analysis followed by segmenting huge clusters, as proposed here, provides an integrated framework for exploring the data.

3 Proposed Methodology

While density-based (e.g. DBScan) or distance relatedness based (e.g. Mitosis) algorithms track patterns connectivity to form clusters, algorithms as K-means partition data around centers, forcing clusters to take globular or hyper-spherical shapes. However, despite the inability of center-based algorithms to discover the true clusters, those algorithms still have their own benefit. The work presented here illustrates how

less efficient partitioning algorithms can be important to investigate the internal structure of arbitrary shaped clusters to understand how connectivity between patterns is formed. Figure 2 shows a generalized sequential clustering ensemble where any algorithm A is used to cluster the data, and any algorithm B is used to segment each cluster into a number of segments/subclusters.

Algorithm ***SequentialClusteringEnsemble***
Input: dataset P
Output: Clusters and segments of each cluster
Begin
  *Execute algorithm A to get a set of clusters S**
  For each cluster i in S, such that size(i)>threshold
    Execute algorithm B to segment the cluster into c segments/subclusters
    Order the segments/subclusters
    Output/visualize the segmented clusters
  EndFor
End

*outliers are properly handled as described later.

Fig. 2. Sequential Clustering Ensemble Method

Mitosis is used to obtain the initial clusters (Algorithm A). It initially retrieves the dynamic range neighbours for all the patterns (a metric tree is used for a logarithmic search time complexity). It then uses a merging criterion that merges patterns based on the consistency of their neighbourhood distances, and merges clusters based on the consistency of their distance characteristics. Mitosis uses two parameters f and k, where f defines a dynamic neighbourhood range for each pattern and k defines a dynamic thresholding criterion for cluster merging. The selection of parameters is based on a distance-relatedness internal validity measure proposed in [8] and [18], where different sequential parameter settings (for both f and k) are used to obtain clusters, and local minima when changing k for a specific f value are used to locate the best k value for each f. The clustering solutions obtained at all best k values are compared using the validity measure in [18] to locate the best f and k values. Clusters above a certain threshold (for example 5% from the total data set) are selected for the segmentation step. This threshold is important to remove the effect of any outlier clusters. Also, patterns that are not allocated to clusters above this threshold are also considered outliers and removed before applying the next step. Those patterns can then be allocated to major clusters later on to have a complete clustering solution.

K-means, used for cluster segmentation (algorithm B) needs the specification of the number of clusters c, and depends on the initial selection of the cluster centers (or initialization of the membership matrix). For the first problem, a number of issues are important to consider, as discussed below. To solve the second problem, multiple random initializations of centers (or memberships) are used and the solution yielding the best objective value is selected, where the objective function minimizes the distances between patterns in a cluster and their cluster center [10].

Number of segments/subclusters
The selection of the number of subclusters (segments) can be done using a proper validity index as Dunn's index [19], where the number of subclusters can be varied between a lower and an upper range values, and K-means run at each setting. The setting that yields the highest Dunn's value is selected. However, this solution might not yield the best visualization quality, and over-clustering might be more appropriate in this case to serve the primary goal of the proposed methodology. Therefore, a proposed method in this case is to increase the number of clusters gradually and locate the number of clusters at which local maxima of Dunn's index are obtained (shown in the experimental results). This can also be user-dependent, however such heuristic can help the user select the number of segments.

Multilevel segmentation
Similar to the concept of hierarchical clustering, a multi-level segmentation of each initial cluster can be obtained, however there is no restriction on the number of subclusters/segments to start with at each level.

Ordering segments and Visualization
In order to be able to visualize the cluster segments, they have to be ordered to reflect the gradual change of patterns inside a cluster, and thus reflect patterns' relations. This is achieved by arranging the patterns of segments on the 2-D plot of the coefficients of highest ranked two SVD components (as will be shown in the results).

Effect of Outliers
Outliers have been known to greatly affect algorithms as K-means, as moving a cluster's center away from its appropriate position, and thus leading to wrong partitioning. The presence of outliers will also affect the visualization process. Fortunately, using Mitosis remove outliers during the clustering process, and it allocates them after clustering to the appropriate clusters according to a merging criteria proposed in [8]. This facilitates excluding outliers when running K-means which can be used before the outlier allocation process.

4 Experimental Results

Two gene expression data sets are used to experiment with the proposed methodology: a small serum dataset with 512 genes and 12 time points [20], and a larger breast cancer dataset of 7129 genes and 49 samples [21],[22]. The SVD based visualization (2-D plot of the coefficients of the two SVD components corresponding to the highest eigenvalues) of the clusters in these sets is shown in figures 3 and 5. (Note that the clustering is done on the original high dimensional data, and the SVD plots are only used for visualization). The connectedness of patterns in the obtained clusters is discovered using the sequential ensemble together with the aid of visualization. In gene expression, connectedness of expression patterns is important to investigate, in order to discover functional relations among genes, where genes affect one another.

In each case, the number of subclusters (for K-means) was decided using local maxima of Dunn's index values obtained when changing the number of clusters. The range of segments considered was 2 to 10 segments. The segments are visualized using both the expression vs. condition/time points plot and the 2-D SVD based visualization. Note that in each case, only one selected cluster obtained by Mitosis is used to illustrate the proposed method (for serum, there is one main cluster, while for breast cancer the cluster investigated is one of two clusters).

Mitosis is executed at parameters f=1.25 and k=1.4 for the serum dataset. The resulting segments are visualized by plotting all time series patterns of the same segment. The use of multilevel segmentation is also shown, where hidden continuity of expression patterns when using 5 segments (figure 4.a) is uncovered when segmenting 3 initial segments into another 8 segments (figure 4.b). The segments in figure 4 show that relatively high expression patterns during the first half of time points seem to be translated to relatively high expression patterns in the second half of the time points, giving a possibility of a regulatory relation between genes expressing in the first half and those expressing in the second half. It can be interpreted as a shift in high expression along the time course of the experiment.

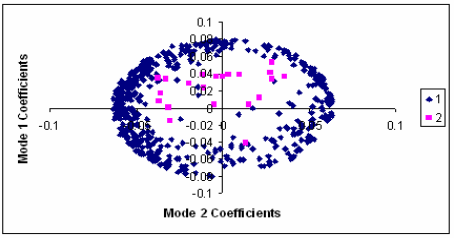


Fig. 3. Clusters of connected patterns obtained by Mitosis for the serum dataset

For the breast cancer dataset, Mitosis parameters were set at f=1.2, and k=1.2. Figure 5.a shows the two clusters obtained, and the higher density cluster (outer periphery) is selected for segmentation. Dunn's local maxima returned 4, 6 and 9 subclusters.

While the choice of 4 subclusters hid the gradual change of expression patterns, the choice of 9 subclusters uncovered the gradual change of patterns, with the repetition of one segment. The segments were presented by their means for avoiding plotting a huge number of patterns. Figure 6 shows six of the segments for the outer cluster (shown in the leftmost figure of figure 5.a), where the major changes involved in the connectedness of patterns is apparent in changing samples/conditions 15 and 18 along the periphery. Sample 15 is related to gene over-expression on the left hemisphere and to gene under-expression on the right hemisphere. Sample 18 is related to gene under-expression in the bottom right corner and to over-expression in the upper left corner. This is not discovered if the main cluster is plotted, as all patterns will intermingle with no specific behaviour. Thus, segmentation using K-means with a

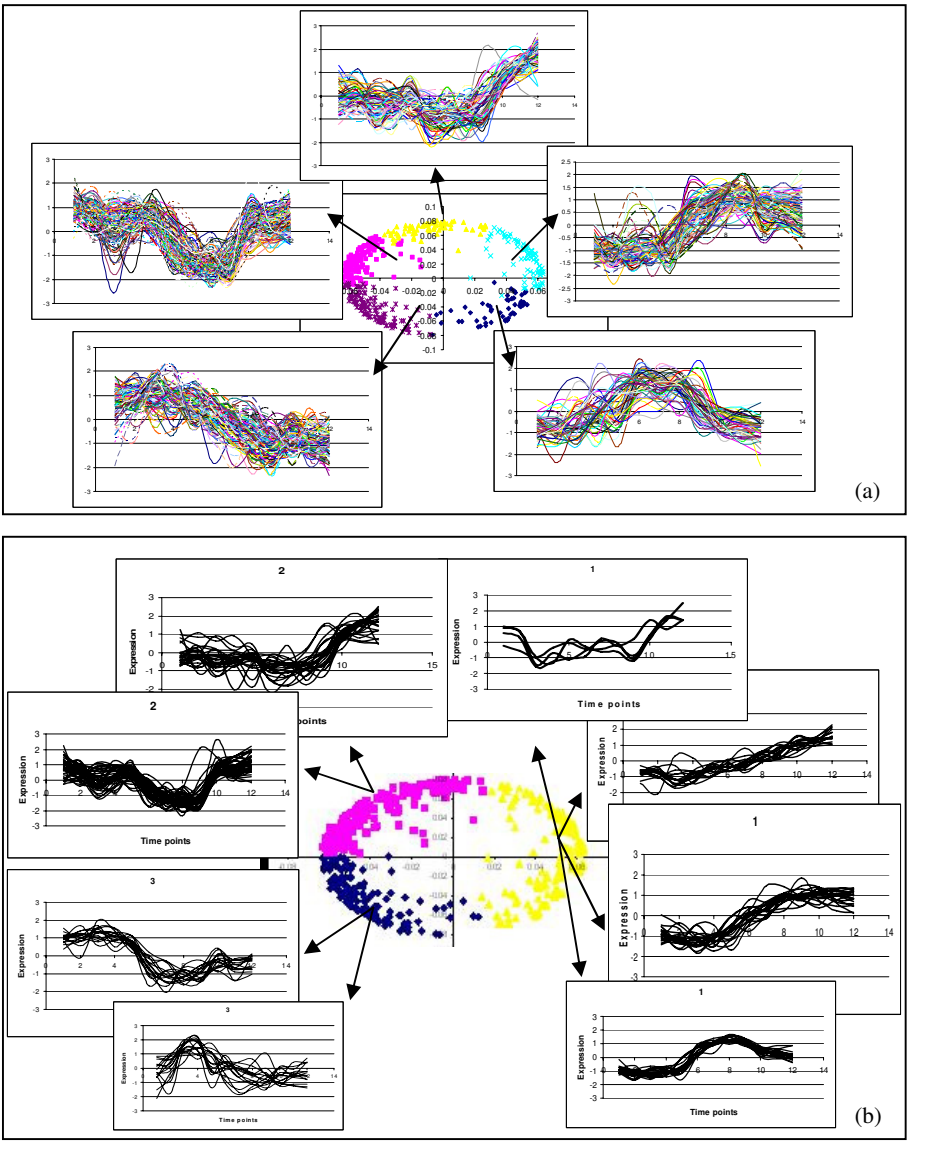


Fig. 4. Segments of the main cluster of serum dataset using pattern visualization and coefficients of SVD components, (a) five segments, (b) eight segments obtained at an extra level of segmentation from an initial set of 3 subclusters

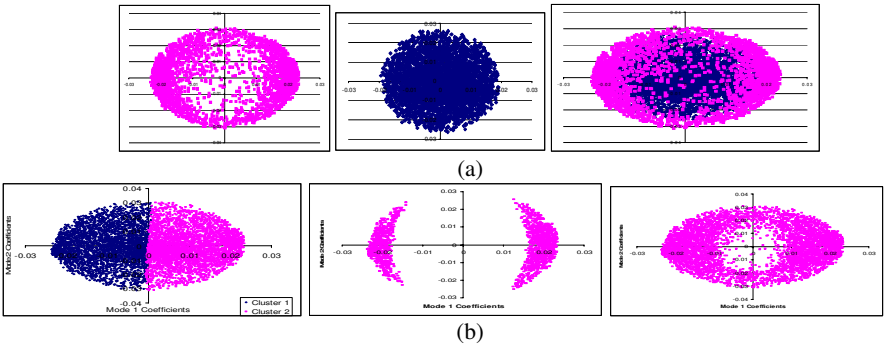


Fig. 5. Clustering results for breast cancer set using a) Mitosis (left most: outer higher density cluster, middle: inner lower density cluster, and rightmost: combined clusters), b) K-means (left most figure, at c=2) and DBScan single cluster at different parameters (middle and rightmost figures)

center-based objective solves this problem for interpreting and understanding pattern connectedness in the cluster.

The connectedness of patterns reveals possible regulatory relations between the genes where over-expression at sample 15 in some genes can result in under-expression in related genes. Also, the high over-expression and under-expression at two specific samples (15 and 18) and the alternation between them reveals information of the presence of possible gene markers that are related to such behavior, and directs attention to further investigate those samples.

Comparing the results obtained from Mitosis to those obtained from other algorithms, figure 5.b shows the clusters resulting from K-means at c=2 and the single cluster resulting from DBScan at two different parameters. Neither of these clustering solutions differentiated the lower and higher density clusters of connected patterns obtained by Mitosis. However, the proposed methodology can be applied to DBScan results to interpret the connectivity of its cluster. It is worth to note that the globular clusters as those obtained by K-means as an initial level of clusters would not capture the connectivity of patterns obtained by Mitosis. It is also important to mention that clusters resulting from hierarchical clustering algorithms as average and complete linkage, tend to be of globular shapes as those obtained by K-means, while single linkage clustering is sensitive to outliers and cannot separate clusters of variable densities. Besides, the high computational complexity of hierarchical clustering hinders its use in huge high dimensional datasets.

In comparison to assessing the significance of clustering in the presence of multiple cluster structure as in [23], validity measures as those developed in [18] for arbitrary shaped clusters, and those developed for center-based ones [19] are used for evaluating the set of clusters obtained at each level. The validity measure proposed in [18] is suitable for evaluating clusters of arbitrary shapes and densities by minimizing standard deviation of distances and maximizing density separateness.

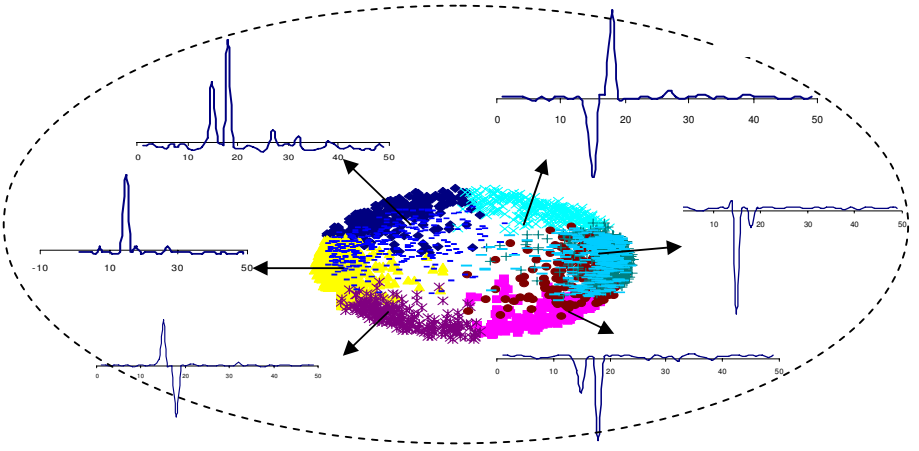


Fig. 6. Segmentation (at 9 segments, showing 6 out of 9 here) of the outer (higher density) cluster of the breast cancer dataset, showing the gradual change (along the dashed circle) of patterns with major changes in samples number 15 and 18

5 Conclusion

A sequential ensemble composed of two clustering algorithms, one with a distance-relatedness based criterion and another with a center-based objective, is introduced. The importance of the ensemble is shown in its ability to analyze huge high dimensional arbitrary shaped clusters of arbitrary densities. The ensemble is used to segment the huge clusters to be able to interpret the connectedness of patterns. The segmentation can be done at multiple levels using classical partitioning algorithms. The results on gene expression sets illustrate the discovery of gradual change among expression patterns.

References

1. Strehl, A., Ghosh, J.: Cluster Ensembles- a knowledge reuse framework for combining multiple partitions. Journal of Machine learning Research 3, 583–617 (2002)
2. Fred, A.L.N.: Finding Consistent Clusters in Data Partitions. In: Kittler, J., Roli, F. (eds.) MCS 2001. LNCS, vol. 2096, pp. 309–318. Springer, Heidelberg (2001)
3. Topchy, A., Jain, A.K., Punch, W.: Combining Multiple Weak Clusterings. In: Proc. IEEE Intl. Conf. on Data Mining, pp. 331–338 (2003)
4. Fred, A.L.N., Jain, A.K.: Data clustering using evidence accumulation. In: Proceedings of International Conference on Pattern Recognition (2002)
5. Hadjitodorov, S.T., Kuncheva, L.I., Todor-ova, L.P.: Moderate diversity for better cluster ensembles. Information Fusion 7(3), 264–275 (2006)
6. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning 52, 91–118 (2003)

7. Alter, O., Patrick, O.B., Botstein, D.: Singular value decomposition for genome-wide expression data processing and modeling. Proc. Natl. Acad. Sci. USA 97(18) (August 2000)
8. Yousri, N.A., Kamel, M.S., Ismail, M.A.: A Distance-Relatedness Dynamic Model for Clustering High Dimensional Data of Arbitrary Shapes and Densities. Journal of Pattern Recognition (July 2009)
9. Yousri, N.A.: Novel Methodologies for Discovering Clusters of Arbitrary Shapes and Densities in High Dimensional Data, with Applications. Ph.D.Thesis, Computers and Systems Engineering, Alexandria University, Egypt (June 2008)
10. Hartigan, J.A.: Clustering Algorithms. Wiley Series in Probability & Mathematical Statistics (1975)
11. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial data sets with noise. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, pp. 226–231
12. Hinneburg, A., Keim, D.: An efficient approach to clustering in large multimedia data sets with noise. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
13. Karypis, G., Han, E.H., Kumar, V.: CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. Computer 32(8), 68–75 (1999)
14. Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical Attributes. In: Proceedings of the IEEE Conference on Data Engineering (1999)
15. Yousri, N.A., Ismail, M.A., Kamel, M.S.: Discovering Connected Patterns in Gene Expression Arrays. In: IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Hawaii, USA (2007)
16. Yousri, N.A., Kamel, M.S., Ismail, M.A.: Pattern Cores and Connectedness in Cancer Gene Expression. In: 7th IEEE International Conference on BioInformatics and BioEngineering (BIBE), Boston, USA (October 2007)
17. Yousri, N.A., Kamel, M.S., Ismail, M.A.: A Fuzzy Approach for Analyzing Outliers in Gene Expression Data. In: BMEI 2008, Haikou, China (2008)
18. Yousri, N.A., Kamel, M.S., Ismail, M.A.: A Novel Validity Measure for Clusters of Arbitrary Shapes and Densities. In: International Conference of Pattern Recognition, ICPR 2008 (2008)
19. Dunn, J.C.: Well separated clusters and optimal fuzzy partitions. J. Cybern. 4, 95–104 (1974)
20. http://www.sciencemag.org/feature/data/984559.shl
21. West, M., et al.: Predicting the clinical status of human breast cancer by using gene expression profiles. PNAS 98(20), 11462–11467 (2001)
22. http://data.cgt.duke.edu/west.php
23. Bertoni, A., Valentini, G.: Discovering multi–level structures in bio-molecular data through the Bernstein inequality. BMC Bioinformatics 9(suppl. 2), S4 (2008)

Combining 2D and 3D Features to Classify Protein Mutants in HeLa Cells

Carlo Sansone[1], Vincenzo Paduano[1,3], and Michele Ceccarelli[2,3]

[1] Dipartimento di Informatica e Sistemistica, University of Naples Federico II, Italy
[2] Dipartimento di Studi Biologici e Ambientali, University of Sannio, Benevento, Italy
[3] Bioinformatics CORE Lab, IRGS Istituto di Ricerche Genetiche G. Salvatore, c/o BioGeM s.c.a r.l., Ariano Irpino (AV), Italy

Abstract. The field of high-throughput applications in biomedicine is an always enlarging field. This kind of applications, providing a huge amount of data, requires necessarily semi-automated or fully automated analysis systems. Such systems are typically represented by classifiers capable of discerning from the different types of data obtained (i.e. classes). In this work we present a methodology to improve classification accuracy in the field of 3D confocal microscopy. A set of 3D cellular images (z-stacks) were taken, each depicting HeLa cells with different mutations of the UCE protein ([Mannose-6-Phosphate] **U**n**C**overing **E**nzyme). This dataset was classified to obtain the mutation class from the z-stacks. 3D and 2D features were extracted, and classifications were carried out with *cell by cell* and *z-stack by z-stack* approaches, with 2D or 3D features. Also, a classification approach that combines 2D and 3D features is proposed, which showed interesting improvements in the classification accuracy.

1 Introduction

Nowadays the field of high-throughput biomedical applications is increasing, and machines and methods high-throughput-capable are being employed more and more.

Such applications, as the name could suggest, furnish as output a huge quantity of data; for example, a study on screens for RNAi in Drosophila cells can generate from 400.000 up to millions of images per time, each containing tens to hundreds of cells [19]. Such huge amount of data must of course be analyzed; being a human analysis too much demanding and time consuming, fully automated analysis methods are required.

A class of applications into this field are the technologies that allow us to obtain three-dimensional fluorescence microscopy images, with techniques such as confocal microscopy or multiphoton microscopy [14,17]. Indeed, many biological research activities require the observation and measurement of 3D, 4D[1] and even 5D[2] cell behavior rather than the common 2D images obtained by classical microscopy.

Therefore it is becoming important to implement algorithms that can successfully and automatically analyze also 3D data (as is nowadays common with 2D images) retrieved from three-dimensional microscopy. Such algorithms should be able, typically, *i*) to

[1] I.e. 3D+time.
[2] 3D+time, multichannel.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 284–293, 2010.

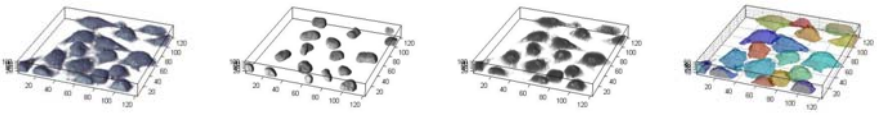


Fig. 1. Sample of a z-stack. (a) Cytoplasm channel; (b) DNA channel; (c) UCE channel; (d) Related segmentation; different cells are shown in different colors.

recognize objects of interest in the analyzed images or volumes (2D or 3D segmentation) and *ii*) to perform further analysis on the segmented objects to achieve features such as proteins locations (i.e. *location proteomics*), cell shape or distribution, cell kind identification, and so on. This second step in the analysis process is typically a classification problem.

One classical interesting study is *location proteomics*, i.e. to identify the position of a protein into the cell [12]; such an information about which organelle or cell structure the protein is in or near is often very useful or sometimes essential to identify the protein function [2].

The aim of the present paper is to classify the *z-stacks* (i.e. 3D images) in a dataset according to the mutations they were depicting. This is a topic of great interest as demonstrated by many recent studies devoted to the automated analysis of sub-cellular patterns [12] and, more generally, bio-molecular imaging [1]. In particular, the work presented in this paper turned out to be an *implicit location proteomics* study.

The protein taken into consideration was the Uncovering Enzyme (UCE) in HeLa cells (see section 2.1). This protein's C-terminus contains multiple signals for trafficking it between lysosomes, the plasma membrane, and the *trans*-Golgi network (TGN) [9]. In this work we considered the mutations of the enzyme's C-terminus depicted in [11], which could alter the subcellular location pattern of the UCE. Stated this, it is obvious that a *location proteomics* study can be carried out to identify the protein mutants. Classifying the UCE mutants could work in the following way: firstly executing a *location proteomics* study, to see the cell location of the mutants in the z-stacks. Then, knowing the behavior and the location each mutant travels near or is stuck into, we could assign the mutant class to each z-stack comparing those informations and the results of the *location proteomics* study.

In this paper, utilizing the same methods employed in *location proteomics*, the z-stacks (or the cells themselves) are assigned to different classes directly from the fluorescence data; this is why we used the term *implicit*.

Location proteomics from 2D and 3D data is not a new field; [16] showed that 3D *location proteomics* allows a higher accuracy with respect to 2D, and that it can even reach a precision superior to an human observer, in particular for proteins located in different positions of the same cellular compartment[3] [2]. In those works the classification, though starting from single-cell analysis, is realized *image by image* or *stack by stack* for the 3D case.

[3] For example proteins located in the *cis*-Golgi and the *trans*-Golgi.

Table 1. Classes and number of samples for each class

	WT	502Stop	Y488-A	YGE-A	YM-A	YMN-A	YN-A	YPL-A	YQ-A	YQE-A	YQEMN-A	Total
Samples	255	393	416	543	203	393	169	347	103	317	131	3270

One of the contributions of the present paper is mutant classification through *implicit location proteomics*; moreover, differently from previous approaches, the classification was performed for each single cell (i.e. *cell by cell*) instead of *image by image* or *stack by stack*. Finally, we propose a new classification approach, involving a combination of 2D and 3D features, that demonstrated better performance with respect to the use of only 2D or 3D feature sets.

The rest of the paper is organized as follows: in Section 2 the materials and methods utilized will be explained; the results obtained with the different classifications realized will be illustrated in Section 3 and some conclusion will be drawn in Section 4.

2 Materials and Methods

2.1 Dataset

The considered data was the 3D UCE HeLa dataset[4], a collection of *z-stacks* from [11] created by Dr. Jack Rohrer's group, consisting of fluorescence microscope images of cells expressing GFP-tagged constructs of the mannose-6-phosphate uncovering enzyme (UCE[5]). The z-stacks were collected using laser-scanning confocal microscopy, and they were composed by three different fluorescence channels: a cytoplasm channel (LRSC labeling), a DNA channel (DAPI labeling) and of course an UCE channel (Fig. 1 a-c). The z-stacks show HeLa cells with different mutants of the UCE; a full functioning one, a truncated one (stopped at amino acid no. 502) and others with different ending amino acid sequence, all truncated at 502. The mutants were created in order to study the behavior of the protein with the purpose to identify the motif for its transportation; of course for each mutant a *location proteomics* study was carried out, depicting if the considered mutants traveled in an unchanged manner from the *trans*-Golgi to the plasma membrane with a fast return or resulted to be stuck somewhere in the cell.

2.2 Segmentation

The z-stacks were firstly segmented with the enhanced 3D Interaction Model proposed in [3] in order to extract the single cells from the z-stacks. This segmentation method is the 3D extension of the Interaction Model proposed in [19], as well with some improvements. It is very suitable for the segmentation of touching and clustered cells, which is the case for many z-stacks of the considered dataset.

[4] http://murphylab.web.cmu.edu/data/#3DUCE

[5] Called also N-acetylglucosamine-1-phosphodiester alpha-N-acetylglucosaminidase (NAGPA), http://www.ncbi.nlm.nih.gov/entrez/dispomim.cgi?id=607985

The enhanced 3D Interaction Model works with three mechanisms: *i*) *Competition*, *ii*) *Repulsion* and *iii*) *Connectedness*. The first one has the purpose to discriminate if a voxel v belongs to the background or a cell, and in this case which. The second one takes into consideration the fact that cells do not intersect each other; they may be tightly clustered, but not compenetrating. The third one manages the fact that living cells always remain compact, i.e. they can never have detached pieces of themselves.

This model is formulated as:

$$\begin{aligned} E &= E_{\text{COMP}} + E_{\text{REP}} + E_{\text{CONN}} \\ &= \lambda_0 \sum_{i=1}^{M} \int_{\Omega} |Z - c_i|^2 \cdot (1 - H(\Psi_i(x,y,z)))\, dxdydz \\ &+ \lambda_b \int_{\Omega} |Z - c_b|^2 \cdot \prod_{i=1}^{M} H(\Psi_i(x,y,z))\, dxdydz \\ &+ \mu \sum_{i=1}^{M} \int_{\Omega} g(Z) |\nabla \Psi_i(x,y,z)| \cdot \delta(\Psi_i(x,y,z))\, dxdydz \\ &+ \omega \sum_{i=1}^{M} \sum_{j=1,\, j\neq i}^{M} \int_{\Omega} (1 - H(\Psi_i(x,y,z))) \cdot (1 - H(\Psi_j(x,y,z)))\, dxdydz \\ &+ \eta \sum_{i=1}^{M} \int_{\Omega} (1 - \xi_i)\, dxdydz \end{aligned} \tag{1}$$

where Z is the z-stack, Ω is the whole observation volume domain, H and δ are the Heaviside function and the delta of Dirac:

$H(x) = \begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases}$ and $\delta(x) = \frac{d}{dx} H(x)$. So, if we indicate the segmenting surfaces as S_i, $inside(S_i) = 1 - H(\Psi_i)$ and $outside(S_i) = H(\Psi_i)$.

c_i and c_b are the intensity mean of the segmented objects and the background respectively, λ_0, λ_b, μ, ω, η are weighting parameters. The segmenting surfaces S_i are not represented by *snakes* as in classical approaches, but with four dimensional Level Set Function (Ψ_i) where the S_i are represented by their zero-set level. g is a crescent sigmoid function defined as

$$g(x) = \left(1 + e^{(\frac{x-\beta}{\alpha})}\right)^{-1}$$

and ξ_i is derived from the following consideration: let us consider two points P_0 and P_1 which both belong to $Q = 1 - H(\Psi_i)$ i.e. the segmented volume, and consider the function $f\colon I \to \Omega$ being a *path* connecting P_0 and P_1 (i.e. $f(0) = P_0$ and $f(1) = P_1$) where I is the unit interval $[0, 1]$ and Ω is the whole stack domain.

Therefore we introduce the function ξ_i which is formulated as:

$$\xi_i = \begin{cases} 1, & if\, f(x) \in Q,\, \forall\, P_0, P_1, x \\ 0, & else \end{cases} \tag{2}$$

where $x \in I$ and $P_0, P_1 \in Q$.

In other terms, if Q is (path-)connected ξ_i will be 1 on its whole domain; otherwise, it will be 0.

A segmentation example is shown in Fig. 1 (d).

2.3 Feature Extraction

3D Features. The feature extraction process is obliged before a classification. Three kind of features were chosen to be extracted and processed: *i*) Haralick texture features [6], *ii*) 3D invariant moments [13,5] and *iii*) volumes.

Haralick texture features are defined for 2D; they are orientation dependent because they must be computed comparing a pixel to its neighbor. To achieve a degree of rotational invariance they are usually computed using a set of offsets sweeping through 180 degrees (i.e. 0, 45, 90, and 135 degrees) at the same distance.

In the 3D case this is no more true because of the extra dimension, so it was decided to cover a 180 square degrees solid angle. To achieve this, the 0-45-90-135 degrees orientations were calculated four times, on a 0, 45, 90 and 135 degrees inclined plane, for a total of $3 \times 4 + 1 = 13$ orientations (45, 90 and 135 degrees were calculated on all four plane orientations, stating that 0 degrees was the same for all the planes). The 13 Haralick features were calculated, each one per channel. So we had: 13 orientations $\times$ 13 features $\times$ 3 channels = 507 Haralick features in total. Those features were also optimized for a faster calculation as described in [10].

The 3D invariant moments $J1$, $J2$ and $J3$, which are numerical shape descriptors, were calculated on the segmentations, and the volumes were calculated for the cells and the UCE quantity, for a total of 512 features total. All the features were calculated for each segmented cell individually, and normalized in $[0, 1]$ on a $128 \times 128 \times slicesno.$ resized version of the z-stacks.

2D Features. A 2D feature extraction was also carried out. To do this, for each z-stack the most representative slice was chosen, and then used as a 2D image. The features were the same for the three-dimensional case, of course adapted for the two dimensions: *i*) Haralick texture features, *ii*) Hu invariant moments [7] and *iii*) areas.
The most significant slice was chosen to be the one in which the whole z-stack Center Of Fluorescence (COF) was located [16]. The needed segmentations were obtained by extracting the correspondent slice from the 3D segmentations.

2.4 Classification

The classification was carried out by using a Support Vector Machine (SVM) with polynomial kernel [15].

As said earlier, the chosen classes were the same as the protein mutants. The cells were extracted from each z-stack, for a total of 3270 samples (Table 1).

Different classifications were carried out with different criteria and aggregation methods: *i*) a 2D *cell by cell* classification, *ii*) a 2D *image by image* classification, where each image was classified by means of a majority voting among the cells in it. Differently from the previous case, cells coming from the same image were used in the training set or in the test set but not in both, *iii*) a 3D *cell by cell* classification, *iv*) a

Table 2. 3D *stack by stack* classification with all features; classification of the cells belonging to the unclassified z-stacks. Rows show the real classes, columns the classification.

	502Stop	WT	Y488_A	YGE-A	YM-A	YMN-A	YN-A	YPL-A	YQ-A	YQE-A	YQEMN-A
WT	0	1	0	0	0	1	0	0	0	0	0
YGE-A	0	0	0	4	0	0	0	0	0	4	0
YGE-A	0	0	0	3	0	0	0	0	0	3	0
YM-A	0	0	0	0	10	0	10	0	0	0	0
YMN-A	0	0	0	3	0	0	0	0	3	0	0
YQ-A	0	0	0	3	0	0	0	0	3	0	0
YQEMN-A	0	0	0	0	0	0	0	0	0	3	3

3D *stack by stack* classification where each z-stack was classified by means of the cells in it; once again cells coming from the same z-stack were used in the training set or in the test set but not in both, and finally *v*) a combination of 2D and 3D classifications (*image by image* + *stack by stack* - see details below).

Also, several classifications were carried out with different features subsets.

The validation technique used was a 10-fold cross-validation: the results reported in the next Section are then computed calculating the average of all the ten validations.

Combining 2D and 3D Classification. As stated before, single z-stacks (in 3D) or single images (in 2D) were classified starting from their contained cells classification. Each z-stack (respectively, image) contained only a single protein mutation, so each one of the cells from the same z-stack (image) belongs to the same class. Therefore, referring to the majority class of the cells in a z-stack (image), we can easily assign the z-stack (image) to the relative class.

For combining 3D and 2D classification we proposed to use a Weighted Majority Voting Rule (*WMV* rule). If we denote as $D_k(x)$ (D_k for short) the weight assigned to the vote V_k of the $k-th$ classifier (i.e., the assignment of the cell x to a class by the 2D or the 3D classifier), the sum of the weighted votes for class C_i on the whole image is given by:

$$W^i = \sum_k D_k \cdot V_k^i \tag{3}$$

Note that each classifier votes m times if the z-stack (image) is composed by m cells. The output provided by the WMV rule is then:

$$Y = \arg\max_i W^i \tag{4}$$

In order to evaluate D_k, i.e. the reliability of the vote given related by the $k-th$ classifier, the most common choice is to use the confusion matrix E^k [18]. The generic element $e^k_{i,j}$ ($1 \leq i,j \leq n$, where n is the number of the classes) of E^k represents the number of samples of the class C_i assigned to the class C_j. More formally, let us

Table 3. 2D+3D *stack by stack* combined classification confusion matrix. This is referred to classification based only on the Haralick texture features. Rows show the real classes, columns the classification.

	502 Stop	WT	Y^{488}-A	YGE-A	YM-A	YMN-A	YN-A	YPL-A	YQ-A	YQE-A	YQEMN-A	Rejected	Total	Recognition Rate
502 Stop	20	0	0	0	0	0	0	0	0	1	0	0	21	95.23%
WT	0	32	1	0	0	0	0	0	0	0	0	0	33	96.96%
Y^{488}-A	0	0	36	0	0	0	0	0	0	0	0	0	36	100.00%
YGE-A	0	0	0	46	0	1	0	1	0	3	0	0	51	90.19%
YM-A	0	0	0	0	6	0	1	0	1	0	2	1	11	54.54%
YMN-A	0	0	0	3	0	29	0	1	0	2	0	0	35	82.85%
YN-A	0	0	0	0	0	0	7	0	0	0	0	1	8	87.50%
YPL-A	1	0	2	6	0	0	0	23	0	0	1	0	33	69.69%
YQ-A	0	0	0	0	0	0	0	0	12	0	1	0	13	92.30%
YQE-A	0	0	0	2	0	2	0	0	0	35	0	0	39	89.74%
YQEMN-A	0	1	0	0	0	0	0	0	0	0	12	0	13	92.30%

denote by $Y_k(x)$ the output of the $k-th$ classifier when the cell x is submitted to it. A reasonable definition of D_k based on E^k is:

$$D_k = \frac{e^k_{i,i}}{\sum_j e^k_{j,i}} \tag{5}$$

given $Y_k = i$. In fact, $e^k_{i,i}$ is the number of cells of C_i which have been correctly classified by the $k-th$ classifier, and $\sum_j e^k_{j,i}$ is the total number of cells of any class assigned to C_i. It follows that $e^k_{i,i} / \sum_j e^k_{j,i}$ is an estimate of the probability that a sample has been correctly classified if it has been assigned to C_i. Thus, Eq. 5 expresses the *a posteriori* probability that the $k-th$ classifier gives the correct answer.

3 Experimental Results

3.1 2D

2D Cell By Cell. This dataset is composed by 3270 cells. The elaborations on the 2D cell by cell were done in two different ways: *i*) the COF was calculated on the DNA channel and *ii*) the COF was calculated on the UCE channel. The classification was carried out with the whole feature set.

Channel	*Recognition Rate*
DNA COF	88.99%
UCE COF	**89.97**%

As in [16] the higher recognition rate was obtained by selecting the COF from the protein channel.

2D Image By Image. The 2D *image by image* dataset was composed by 293 entries.

Features	*Recognition Rate*	*Rejection Rate*
All Features	73.28%	0.02%
2D Haralicks	73.28%	0.02%

This represents a more realistic case with respect to the cell by cell classification; in fact new cells to be classified will come entirely from brand new images, while cells from different images would have likely been used for the classifier training. This justifies the decreasing in classifier performance.

The rejection rate derives from the fact that for some images there was not the majority of cells classified as belonging to a specific class, i.e. a tie occurs between two (or even more) classes.

3.2 3D

3D Cell By Cell. Like the 2D dataset, of course, the 3D dataset was composed by 3270 individual cells.

Features	*Recognition Rate*
All Features	**98.29%**
3D Haralicks	97.37%

As predictable, the biggest amount of information is given to the classifier by the 3D Haralick texture features. Anyway, combining it with the invariant moments and the volumes subsets increases the accuracy of about 1%. Respectfully to the previous 2D classification, an increase in accuracy is notable when turning to 3D, as stated in [2].

3D Stack By Stack. The dataset was composed by 293 z-stacks.

Features	*Recognition Rate*	*Rejection Rate*
All Features	82.87%	0.02%
3D Haralicks	**83.90%**	**0.01%**

Once again, the decreasing of the performance with respect to the previous case is due to the fact that new cells to be classified will come entirely from brand new z-stacks.

As in the 2D case, some images were not classified. The classes of the cells belonging to the unclassified images in the full feature classification are shown in Table 2.

It is worth noting that in this case the use of the whole feature set does not improve the recognition rate with respect to the use of Haralick feature alone.

3.3 Combining 2D and 3D Classification

This classification has been carried out considering only the Haralick feature subset for both 2D and 3D.

Features	*Recognition Rate*	*Rejection Rate*
2D + 3D Haralicks	**88.05%**	**0.02%**

The confusion matrix is shown in Table 3. This approach seems to be particularly promising, because it shows more than a 4% accuracy improvement with respect to the best results obtained by using only 3D features.

4 Conclusions and Future Works

In this paper a set of 3D images (*z-stacks*) obtained by confocal microscopy was taken. In this set HeLa cells with different mutations of a protein called UCE ([Mannose-6-Phosphate] **U**n**C**overing **E**nzyme) were depicted. A classification study has then been carried out; i.e. trying to take the z-stacks and classify them according to their UCE mutation (thus realizing an *implicit location proteomics* study). The z-stacks were segmented with the enhanced 3D Interaction Model to isolate single cells, and the features to be passed to the classifier were extracted: 3D Haralick texture features, 3D invariant moments and volumes. Also, the counterparts 2D features (2D Haralicks, Hu moments and areas) where extracted from the most significant slice in each z-stack (COF criteria).

By using a Support Vector Machine with polynomial kernel, images were classified by using 2D features, 3D features, and a 2D+3D combination.

Results showed a significant improvement in the accuracy of the 3D classification respectfully to the 2D classification. Anyway, 2D+3D showed even better results, depicting the fact that 2D feature also carry extra significant informations that can be used to improve the classification of such dataset. This may be due to the fact that while a 3D analysis is generally more accurate, for a minority of the z-stacks the classifier may not be able to effectively exploit information coming from the extra dimension. A 2D classification support — stated that 2D itself, even if with lower performance than 3D, also gives rise to very good results — overcomes this by bringing extra information used to refine the classification of those z-stacks in a doubtful state of class attribution.

Finally, it has to be remarked that there are other papers [4,8] proposing the use of classifier combination in *location proteomics* or closely related fields. In these papers, however, each multiple classifier system uses only 2D features or only 3D features.

Future work will be devoted to assess the performance of the proposed 2D+3D classification approach on other datasets such as the one used in [16].

References

1. Ahmed, W.M., Leavesley, S.J., Rajwa, B., Ayyaz, M.N., Ghafoor, A., Robinson, J.P.: State of the Art in Information Extraction and Quantitative Analysis for Multimodality Biomolecular Imaging. Proceedings of the IEEE 96(3), 512–531 (2008)
2. Boland, M.V., Murphy, R.F.: A Neural Network Classifier Capable of Recognizing the Patterns of all Major Subcellular Structures in Fluorescence Microscope Images of HeLa Cells. Bioinformatics 17(12), 1213–1223 (2001)
3. Ceccarelli, M., Paduano, V., Sansone, C.: Segmentation of 3D Microscopy Data with an Energy Based Interaction Model. In: IEEE (ed.) MeMeA 2009 - IEEE International Workshop on Medical Measurements and Applications Proceedings, pp. 223–228. IEEE, Los Alamitos (2009)
4. Chen, C., Zhou, X., Tian, Y., Zou, X., Cai, P.: Predicting Protein Structural Class with Pseudo-Amino Acid Composition and Support Vector Machine Fusion Network. Analytical Biochemistry 357, 116–121 (2006)
5. Fisher, B.: 3D Moment Invariants (January 2008), http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FISHER/MOMINV3D/inv3d.htm

6. Haralick, R.M.: Statistical and Structural Approaches to Texture. IEEE Proceedings 67(5), 786–804 (1979)
7. Hu, M.K.: Visual Pattern Recognition by Moment Invariants. IRE Transaction on Information Theory IT-8, 179–187 (1962)
8. Huang, K., Murphy, R.F.: Boosting Accuracy of Automated Classification of Fluorescence Microscope Images for Location Proteomics. BMC Bioinformatics 5, 78 (2004)
9. Lee, W.S., Rohrer, J., Kornfeld, R., Kornfeld, S.: Multiple Signals Regulate Trafficking of the Mannose 6-Phosphate-Uncovering Enzyme. Journal of Biological Chemistry 277(5), 3544–3551 (2002)
10. Miyamoto, E., Merryman, T.: Fast Calculation of Haralick Texture Features (2005)
11. Nair, P., Schaub, B.E., Huang, K., Chen, X., Murphy, R.F., Griffth, J.M., Geuze, H.J., Rohrer, J.: Characterization of the TGN Exit Signal of the Human Mannose-6-Phosphate Uncovering Enzyme. Journal of Cell Science 118(13), 2949–2956 (2005)
12. Newberg, J., Murphy, R.F.: A Framework for the Automated Analysis of Subcellular Patterns in Human Protein Atlas Images. Journal of Proteome Research 7(6), 2300–2308 (2008)
13. Sadjadi, F.A., Hall, E.L.: Three-Dimensional Moment Invariants. IEEE Transaction on Pattern Analysis And Machine Intelligence 2(2), 127–136 (1980)
14. Sapuppo, P., Diaspro, A., Faretta, M.: Microscopia Confocale. Leica Microsystems (2009)
15. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
16. Velliste, M., Murphy, R.F.: Automated Determination of Protein Subcellular Locations from 3D Fluorescence Microscope Images. In: Proceedings of the 2002 IEEE International Symposium on Biomedical Imaging - (ISBI 2002), pp. 867–870. IEEE, Los Alamitos (2002)
17. Vonesc, C., Aguet, F., Vonesch, J.-L., Unser, M.: The Colored Revolution of Bioimaging: an Introduction to Fluorescence Microscopy. IEEE Signal Processing Magazine 23(3), 20–21 (2006)
18. Xu, L., Krzyzak, A., Suen, C.Y.: Method of Combining Multiple Classifiers and Their Application to Handwritten Numeral Recognition. IEEE Transactions on Systems, Man and Cybernetics 22(3), 418–435 (1992)
19. Yan, P., Zhou, X., Shah, M., Wong, S.: Automatic Segmentation of High-Throughput RNAi Fluorescent Cellular Images. IEEE Transactions on Image Processing 12(1), 109–117 (2008)

An Experimental Comparison of Hierarchical Bayes and True Path Rule Ensembles for Protein Function Prediction

Matteo Re and Giorgio Valentini

DSI, Dipartimento di Scienze dell' Informazione,
Università degli Studi di Milano,
Via Comelico 39, 20135 Milano, Italia
{re,valentini}@dsi.unimi.it

Abstract. The computational genome-wide annotation of gene functions requires the prediction of hierarchically structured functional classes and can be formalized as a multiclass, multilabel, multipath hierarchical classification problem, characterized by very unbalanced classes. We recently proposed two hierarchical protein function prediction methods: the Hierarchical Bayes (HBAYES) and True Path Rule (TPR) ensemble methods, both able to reconcile the prediction of component classifiers trained locally at each term of the ontology and to control the overall precision-recall trade-off. In this contribution, we focus on the experimental comparison of the HBAYES and TPR hierarchical gene function prediction methods and their cost-sensitive variants, using the model organism *S. cerevisiae* and the FunCat taxonomy. The results show that cost-sensitive variants of these methods achieve comparable results, and significantly outperform both FLAT and their non cost-sensitive hierarchical counterparts.

1 Introduction

The hierarchical prediction of protein function annotations, such as terms in the Gene Ontology (GO), is a complex computational problem, characterized by several items: the number of functional classes is large, and a gene may belong to multiple classes; functional classes are structured according to a hierarchy; classes are usually unbalanced, with more negative than positive examples [1]. The simplest approach makes predictions for each term independently and, consequently, the predictor may assign to a single protein a set of terms that are inconsistent with one another. A possible solution for this problem is to train a classifier for each term of the reference ontology, to produce a set of prediction at each term and, finally, to reconcile the predictions by tacking into account the structure of the ontology. Many recent published works clearly demonstrated that this approach ensures an increment in precision, but this comes at expenses of the overall recall [2, 3].

Different research lines have been proposed for the hierarchical prediction of gene functions, ranging from structured-output methods, based on the joint

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 294–303, 2010.

kernelization of both input variables and output labels [4, 5], to ensemble methods, where different classifiers are trained to learn each class, and then combined to take into account the hierarchical relationships between functional classes [6, 3, 7].

Our work goes along this latter line of research. Our main contribution to this research area is represented by two methods, the HBAYES and TPR hierarchical ensemble-based gene function predictors [8, 16]. Both the methods are based on the concept of per-term predictions "reconciliation" which exploits information derived from the hierarchical relationships between the terms composing the considered functional ontology [6]. In this approach the first step is constituted by the prediction of protein functions (that is, the Functional Catalogue (FunCat) [10] or the Gene Ontology (GO) [11] terms) on a per-term basis. The obtained predictions are then combined in a post processing stage. The combination step can be realized using many methods and depends on the individual predictions performed by the component classifiers at each term of the ontology.

As observed in [6], many reconciliation methods yield reconciled probabilities with significantly lower precision than the original, unreconciled estimates. This problem can be solved by introducing one or more parameters able to modulate the overall precision but this approach is often associated with a corresponding loss in sensitivity that decrease the practical relevance of the prediction method. In order to ensure the applicability of the HBAYES and TPR ensemble methods in real world problems we recently proposed variants able to control the overall precision-recall trade-off. In this contribution we compare the hierarchical gene function prediction performances of the HBAYES and TPR ensemble methods, both in their respective "vanilla" and cost-sensitive versions, in order to highlight differences in their ability to reconcile base learners predictions and to preserve the overall precision and recall.

2 Methods

2.1 Concepts and Notation

Genome-wide gene function prediction can be modeled as a hierarchical, multiclass and multilabel classification problem. Indeed a gene/gene product $\boldsymbol{x}$ can be assigned to one or more functional classes of the set $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$. The assignments can be coded through a vector of multilabels $\mathbf{y} =< y_1, y_2, \ldots, y_m > \in \{0,1\}^m$, by which if $\boldsymbol{x}$ belongs to class ω_i, then $y_i = 1$, otherwise $y_i = 0$, where the variable $i, 1 \leq i \leq m$, refers to the indices corresponding to the m classes belonging to the set Ω.

In both the *Gene Ontology (GO)* and *FunCat* taxonomies the functional classes are structured according to a hierarchy and can be represented by a directed graph, where nodes correspond to classes, and arcs to relationships between classes. Hence the node corresponding to the class ω_i can be simply denoted by i. We represent the set of children nodes of i by child(i), and the

set of its parents by par(i). Moreover $y_{child(i)}$ denotes the labels of the children classes of node i and analogously $y_{par(i)}$ denotes the labels of the parent classes of i. Note that in FunCat only one parent is permitted, since the overall hierarchy is a tree forest, while in the GO, more parents are allowed, because the relationships are structured according to a directed acyclic graph.

Hierarchical ensemble methods train a calibrated classifier at each node of the taxonomy T. This is used to derive estimates $\widehat{p}_i(\boldsymbol{x})$ of the probabilities $p_i(\boldsymbol{x}) = \Pr\big(V_i = 1 \mid V_{\mathrm{par}(i)} = 1, \boldsymbol{x}\big)$ for all $\boldsymbol{x}$ and i, where $(V_1, \ldots, V_N) \in \{0,1\}^N$ is the vector random variable modeling the multilabel of a gene $\boldsymbol{x}$ and par(i) is the unique parent of node i in T. In order to enforce that only multilabels $\boldsymbol{V}$ that respect T should have nonzero probability, the base learner at node i is only trained on the subset of the training set including all examples $(\boldsymbol{x}, \boldsymbol{y})$ such that $y_{\mathrm{par}(i)} = 1$.

All the experiments presented in this work are performed using FunCat as reference functional ontology.

2.2 The Hierarchical Bayes (HBAYES) Ensemble

The HBAYES ensemble method is a general technique for solving hierarchical classification problems on generic taxonomies [12, 13]. In the evaluation phase, HBAYES predicts the Bayes-optimal multilabel $\widehat{\boldsymbol{y}} \in \{0,1\}^N$ for a gene $\boldsymbol{x}$ based on the estimates $\widehat{p}_i(\boldsymbol{x})$ for $i = 1, \ldots, N$. Namely, $\widehat{\boldsymbol{y}} = \mathrm{argmin}_{\boldsymbol{y}}\, \mathbb{E}\big[\ell_H(\boldsymbol{y}, \boldsymbol{V}) \mid \boldsymbol{x}\big]$, where the expectation is w.r.t. the distribution of $\boldsymbol{V}$. Here $\ell_H(\boldsymbol{y}, \boldsymbol{V})$ denotes the H-loss [12, 13], measuring a notion of discrepancy between the multilabels $\boldsymbol{y}$ and $\boldsymbol{V}$. Given fixed *cost coefficients* $c_1, \ldots, c_N > 0$, $\ell_H(\widehat{\boldsymbol{y}}, \boldsymbol{v})$ is computed as follows: all paths in the taxonomy T from the root 0 down to each leaf are examined and, whenever a node $i \in \{1, \ldots, N\}$ is encountered such that $\widehat{y}_i \neq y_i$, then c_i is added to the loss, while all the other loss contributions from the subtree rooted at i are discarded. As shown in [13], $\widehat{\boldsymbol{y}}$ can be computed via a simple bottom-up message-passing procedure whose only parameters are the probabilities $\widehat{p}_i(\boldsymbol{x})$. In the rest of the paper if there is no ambiguity we denote $\widehat{p}_i(\boldsymbol{x})$ simply by $\widehat{p}_i$.

2.3 Cost-Sensitive Variant of HBAYES

A simple cost-sensitive variant, HBAYES-CS, of HBAYES, described in [8] is suitable for learning datasets whose multilabels are sparse. This variant introduces a parameter α that is used to trade-off the cost of false positive (FP) and false negative (FN) mistakes. We start from an equivalent reformulation of the HBAYES prediction rule:

$$\widehat{y}_i = \operatorname*{argmin}_{y \in \{0,1\}} \left(c_i^- \widehat{p}_i (1 - y) + c_i^+ (1 - \widehat{p}_i) y + \widehat{p}_i \{y = 1\} \sum_{j \in \mathrm{child}(i)} H_j \right) \qquad (1)$$

where $H_j = c_j^- \widehat{p}_j (1 - \widehat{y}_j) + c_j^+ (1 - \widehat{p}_j) \widehat{y}_j + \sum_{k \in \mathrm{child}(j)} H_k$ is recursively defined over the nodes j in the subtree rooted at i with each $\widehat{y}_j$ set according to (1),

and $\{ A \}$ is the indicator function of event A. Furthermore, $c_i^- = c_i^+ = c_i/2$ are the costs associated to a FN (resp., FP) mistake. In order to vary the relative costs of FP and FN, in [8] we introduce a factor $\alpha \geq 0$ such that $c_i^- = \alpha c_i^+$ while keeping $c_i^+ + c_i^- = 2c_i$. Then (1) can be rewritten as

$$\widehat{y}_i = 1 \Longleftrightarrow \widehat{p}_i \left(2c_i - \sum_{j \in \text{child(i)}} H_j \right) \geq \frac{2c_i}{1+\alpha} \ . \tag{2}$$

This is the rule used by HBAYES-CS in our experiments.

2.4 The True Path Rule (TPR) Ensemble

The True Path Rule (TPR) ensemble method [16] not only explicitly takes into account the hierarchical relationships between functional classes, but is also directly inspired by the *true path rule* that can be summarized as follows [14]: "An annotation for a class in the hierarchy is automatically transferred to its ancestors, while genes unannotated for a class cannot be annotated for its descendants". According to this rule, that governs the annotations of both GO and FunCat taxonomies, the proposed ensemble method is characterized by a two-way asymmetric flow of information that traverses the graph-structured ensemble: positive predictions for a node influence in a recursive way its ancestors, while negative predictions influence its offsprings. The resulting ensemble embeds the functional relationships between functional classes that characterize the hierarchical taxonomy. In other words, if a gene is annotated with a specific functional term (functional class), then it is annotated with all the "parent" classes, and with all its ancestors in a recursive way.

The base classifiers estimate local probabilities $\bar{p}_i(\boldsymbol{x})$ that a given example $\boldsymbol{x}$ belongs to class ω_i, but the core of the algorithm is represented by the evaluation phase, where the ensemble provides an estimate of the "consensus" global probability $p_i(\boldsymbol{x})$. Let us consider the set $\phi_i(x)$ of the children of node i for which we have a positive prediction for a given example $\boldsymbol{x}$:

$$\phi_i(\boldsymbol{x}) = \{ j | j \in \text{child}(i), \widehat{y}_j(\boldsymbol{x}) = 1 \} \tag{3}$$

The global consensus probability $\widehat{p}_i(\boldsymbol{x})$ of the ensemble depends both on the local prediction $\bar{p}_i(x)$ and on the prediction of the nodes belonging to $\phi_i(x)$:

$$\widehat{p}_i(\boldsymbol{x}) = \frac{1}{1 + |\phi_i(\boldsymbol{x})|} \left(\bar{p}_i(\boldsymbol{x}) + \sum_{j \in \phi_i(\boldsymbol{x})} \widehat{p}_j(\boldsymbol{x}) \right) \tag{4}$$

The decision $\widehat{y}_i(x)$ at node/class i is set to 1 if $\widehat{p}_i(\boldsymbol{x}) > t$, and to 0 otherwise (a natural choice for t is 0.5). Note that the restriction to nodes belonging to $\phi_i(\boldsymbol{x})$ in the summation of eq. 4 depends on the true path rule: indeed only children nodes for which we have a positive prediction can influence their parent. In the

leaf nodes the sum of eq. 4 disappears and eq. 4 reduces to $\widehat{p}_i(\boldsymbol{x}) = \bar{p}_i(\boldsymbol{x})$. In this way positive predictions propagate from bottom to top. On the contrary if for a given node $\widehat{y}_i(\boldsymbol{x}) = 0$, then this decision is propagated to its subtree.

2.5 The Weighted TPR (TPR-W) Method

In the TPR algorithm there is no way to explicitly balance the local prediction $\bar{p}_i(\boldsymbol{x})$ at node i with the positive predictions coming from its offsprings (eq. 4). By balancing the local predictions with the positive predictions coming from the ensemble, we can explicitly modulate the interplay between local and descendant predictors. To this end we introduced a *weight* w, $0 \leq w \leq 1$, such that if $w = 1$ the decision at node i depends only by the local predictor, otherwise the prediction is shared proportionally to w and $1-w$ between respectively the local parent predictor and the set of its children [15]:

$$\widehat{p}_i(\boldsymbol{x}) = w \cdot \bar{p}_i(\boldsymbol{x}) + \frac{1-w}{|\phi_i(\boldsymbol{x})|} \sum_{j \in \phi_i(\boldsymbol{x})} \widehat{p}_j(\boldsymbol{x}) \tag{5}$$

3 Experimental Setup

In order to compare the capabilities of the HBAYES and TPR methods in hierarchical gene function prediction, we predicted the functions of genes of the unicellular eukaryote *S. cerevisiae* at genome and ontology-wide level using the *FunCat* taxonomy [10], and the data sets described below.

Data sets: In our experiments we used 7 bio molecular data sets, whose characteristics are summarized in Tab. 1. In order to get a not too small set of positive examples for training, for each data set we selected only the FunCat-annotated genes and the classes with at least 20 positive examples. As negative examples we selected for each node/class all genes not annotated to that node/class, but annotated to its parent class. From the data sets we also removed uninformative features (e.g., features with the same value for all the available examples).

Table 1. Data sets

Data set	Description	n. examples	n. feat.	n.classes
Pfam-1	protein domain binary data from *Pfam*	3529	4950	211
Pfam-2	protein domain log E data from *Pfam*	3529	5724	211
Phylo	phylogenetic data	2445	24	187
Expr	gene expression data	4532	250	230
PPI-BG	PPI data from *BioGRID*	4531	5367	232
PPI-VM	PPI data from von Mering experiments	2338	2559	177
SP-sim	Sequence pairwise similarity data	3527	6349	211

Cross validated comparison of ensemble methods: For each ensemble we used gaussian SVMs as base learners. The probabilistic output of the SVMs composing the ensembles has been computed using the sigmoid fitting proposed in [17]. Given a set $\widehat{p}_1, \ldots, \widehat{p}_N$ of trained estimates, we compared on these estimates the results of the HBAYES and TPR ensembles with their corresponding cost-sensitive versions: HBAYES-CS and TPR-W. Both the cost factor α for HBAYES-CS and the w parameter in TPR-W ensembles have been set by internal cross-validation of the F-measure with training data. The threshold t of TPR ensembles has been set to 0.5 in all the experiments. The performance of the ensembles have been compared using external 5-fold cross-validation techniques.

Performances evaluation: Considering the unbalance between positive and negative examples, we could adopt the classical F-score to jointly take into account the precision and recall of the ensemble for each class of the hierarchy. Nevertheless, the classical precision and recall measures, conceived for unstructured classification problems, appear to be inadequate to fully address the hierarchical nature of functional annotation. To this end we used the *Hierarchical F-measure*. This measure is based on the estimation of how much the predicted classification paths correspond to the correct paths. More precisely, given a general taxonomy G representing the graph of the functional classes, for a given gene/gene product x consider the graph $P(x) \subset G$ of the predicted classes and the graph $C(x)$ of the correct classes associated to x, and let be $l(P)$ the set of the leaves (nodes without children) of the graph P. Given a leaf $p \in P(x)$, let be $\uparrow p$ the set of ancestors of the node p that belong to $P(x)$, and given a leaf $c \in C(x)$, let be $\uparrow c$ the set of ancestors of the node c that belong to $C(x)$. Starting from the definitions of Hierarchical Precision (HP), Hierarchical Recall (HR) and Hierarchical F-score (HF) provided in [18], it is easy to demonstrate that in the case of the FunCat taxonomy, since it is structured as a tree, we can simplify HP, HR and HF as follows:

$$HP = \frac{1}{|l(P(x))|} \sum_{p \in l(P(x))} \frac{|C(x) \cap \uparrow p|}{|\uparrow p|}$$

$$HR = \frac{1}{|l(C(x))|} \sum_{c \in l(C(x))} \frac{|\uparrow c \cap P(x)|}{|\uparrow c|}$$

$$HF = \frac{2 \cdot HP \cdot HR}{HP + HR} \tag{6}$$

The hierarchical F-measure expresses the correctness of the structured prediction of the functional classes, taking into account also partially correct paths in the overall hierarchical taxonomy, thus providing in a synthetic way the effectiveness of the hierarchical prediction.

4 Results

4.1 Hierarchical F-Measure Results

As explained in the experimental set-up (Sect. 3), the hierarchical F-measure is a more appropriate performance metric for the hierarchical classification of gene functions. We compared the performances of the considered "vanilla" and cost-sensitive probability reconciliation methods using the Hierarchical F-measure and gaussian SVMs as component classifiers. The results collected in this test are reported in Tab. 2. FLAT ensembles corresponds to predictions directly made by the base learners without any "reconciliation" of the local predictions. According to the Wilcoxon signed-ranks test [19], both HBAYES-CS and TPR-W outperform at 0.01 significance level FLAT, HBAYES and TPR ensemble. No significant difference between HBAYES-CS and TPR-W can be detected (*p-value* $\simeq 0.24$).

Table 2. Hierarchical F-measures comparison between "vanilla" and cost-sensitive hierarchical methods. TPR: True Path Rule hierarchical ensembles; HB-CS: Hierarchical Bayesian bottom-up Cost Sensitive ensembles; TPR-W True Path Rule weighted hierarchical ensembles. Base learners: gaussian SVMs

Data set	FLAT	HBAYES	TPR	HB-CS	TPR-W
Pfam-1	0.1624	0.3359	0.3113	0.4518	0.4188
Pfam-2	0.0402	0.0476	0.1929	0.2030	0.1892
Phylo	0.1196	0.0694	0.2557	0.2682	0.2780
Expr	0.1153	0.0639	0.2390	0.2555	0.2638
PPI-BG	0.0836	0.0847	0.2709	0.2920	0.2315
PPI-VM	0.1720	0.3468	0.3983	0.4329	0.4381
SP-sim	0.1432	0.3246	0.2502	0.4542	0.4501
Average	0.1194	0.1818	0.2732	0.3368	0.3242

4.2 Tuning Precision and Recall in HBAYES-CS and TPR-W Ensembles

In order to compare the capabilities of both the HBAYES and TPR ensemble methods in the modulation of the precision-recall trade-off we tested their performances by varying the values of *alpha* and the parent weight w hyper parameters. Results of this test are reported in (Fig. 1).

The precision/recall characteristics of HBAYES-CS ensemble can be tuned via a single global parameter, the cost factor $\alpha = c_i^- / c_i^+$ (Sect. 2). By setting $\alpha = 1$ we obtain the original version of the hierarchical Bayesian ensemble and by incrementing α we introduce progressively lower costs for positive predictions, thus encouraging the ensemble to make positive predictions. Indeed, by increasing the cost factor, the recall of the ensemble tends to increase (Fig. 1 (a) and (c)). The behavior of the precision is more complex: it tends to increase and then to

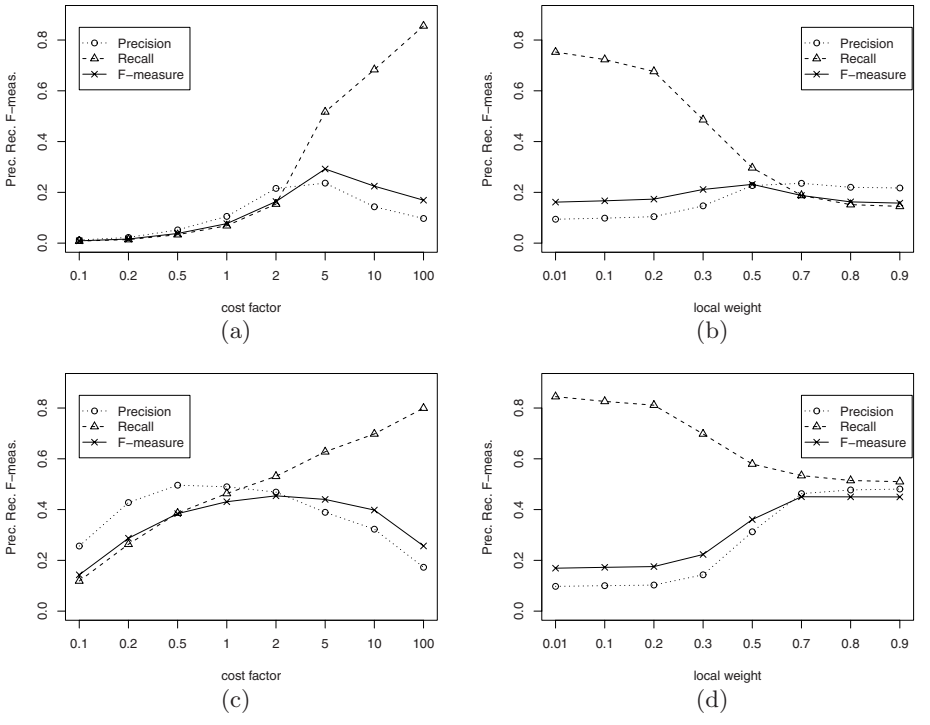


Fig. 1. Precision, Recall and F-measure as a function of the parent weight in TPR-W ensembles and of the global α parameter in HBAYES-CS. PPI BioGRID data: (a) HBAYES-CS (b) TPR-W; Pairwise sequence similarity data: (c) HBAYES-CS (d) TPR-W.

decrease after achieving a maximum. Quite interestingly, the maximum of the hierarchical F-measure is achieved for values of α between 2 and 5 not only for the two data sets reported in Figure 1, but also for all the considered data sets (data not shown).

As seen for HBAYES-CS also the TPR-W ensembles is capable of tuning precision and recall rates, through a single global parameter: the weight w (eq. 5). Fig. 1 (graphs (b) and (d)) shows the hierarchical precision, recall and F-measure as functions of the parameter w. For small values of w (w can vary from 0 to 1) the weight of the decision of the parent local predictor is small, and the ensemble decision depends mainly by the positive predictions of the offsprings nodes (classifiers): as a consequence we obtain a higher hierarchical recall for the TPR-W ensemble. On the contrary higher values of w correspond to a higher weight of the parent predictor, with a resulting higher precision. The opposite trends of precision and recall are quite clear in graphs (b) and (d) of Fig. 1. The best F-score is achieved for "middle" or relatively high values of the w parameter: in practice in most of the analyzed data sets the best F-measure is achieved for w between 0.5 and 0.8, but if we need higher recall rates (at the expense of the precision) we can choose lower w values, and higher values of w

are needed if precision is our first aim. Comparable results were obtained for all the considered data sets (data not shown).

5 Conclusions

In this work we compared the performances of two recently proposed hierarchical gene function prediction methods, the HBAYES and TPR-W ensemble systems. Looking at the results summarized in Tab. 1 it is clear that the usage of hierarchical prediction methods results in a consistent increment in performances if compared with methods that does not take into account the structure of the reference ontology. Also the application of hierarchical methods unable to finely modulate the precision-recall trade-off is suboptimal because the hierarchical F-measure is consistently lowered by the loss in sensitivity associated to the increment in precision due to the reconciliation of the local predictions.

With respect to the comparison of the ability of the cost-sensitive variants of both the HBAYES and TPR methods in the modulation of the overall precision-recall trade-off, the observed results do not allow to define a clear winner. Indeed the performances of the compared methods are quite similar, even if small differences can be found in the performances achieved in the evaluation of different datasets. Both methods share the same top-down strategy to set negative nodes belonging to the subtree rooted at a node predicted as negative by the ensemble, but they pursue very different strategies in the bottom-up computation of the ensemble probabilities. Indeed HBAYES approximates the Bayesian-optimal predictor w.r.t. the H-loss, while TPR is based on a heuristic borrowed from the true path rule. Interestingly enough, these different methods lead to similar results, and their cost-sensitive counterparts significantly outperform FLAT and "vanilla" hierarchical ensembles. The hierarchical algorithms are general enough to be applied to problems other then gene function prediction. Indeed, the cost-sensitive HBAYES and TPR methods, even if conceived for gene function problems, can be applied to other hierarchical classification problems where the descendant classes can be interpreted as parts or subclasses of their corresponding ancestors.

Acknowledgments

The authors thank the reviewers for their comments. The authors gratefully acknowledge partial support by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

[1] Friedberg, I.: Automated protein function prediction-the genomic challenge. Brief. Bioinformatics 7, 225–242 (2006)
[2] Pena-Castillo, L., et al.: A critical assessment of Mus musculus gene function prediction using integrated genomic evidence. Genome Biology 9(S1) (2008)

[3] Guan, Y., Myers, C., Hess, D., Barutcuoglu, Z., Caudy, A., Troyanskaya, O.: Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biology 9(S2) (2008)
[4] Sokolov, A., Ben-Hur, A.: A structured-outputs method for prediction of protein function. In: MLSB 2008, the Second International Workshop on Machine Learning in Systems Biology (2008)
[5] Astikainen, K., Holm, L., Pitkanen, E., Szedmak, S., Rousu, J.: Towards structured output prediction of enzyme function. BMC Proceedings 2(suppl. 4:S2) (2008)
[6] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M.I., Noble, W.S.: Consistent probabilistic output for protein function prediction. Genome Biology 9(S6) (2008)
[7] Jiang, X., Nariai, N., Steffen, M., Kasif, S., Kolaczyk, E.: Integration of relational and hierarchical network information for protein function prediction. BMC Bioinformatics 9(350) (2008)
[8] Cesa-Bianchi, N., Valentini, G.: Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. Journal of Machine Learning Research, W&C Proceedings (to appear)
[9] Valentini, G.: True path rule hierarchical ensembles. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 232–241. Springer, Heidelberg (2009)
[10] Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Guldener, U., Mannhaupt, G., Munsterkotter, M., Mewes, H.: The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research 32(18), 5539–5545 (2004)
[11] The Gene Ontology Consortium: Gene ontology: tool for the unification of biology. Nature Genet. 25, 25–29 (2000)
[12] Cesa-Bianchi, N., Gentile, C., Tironi, A., Zaniboni, L.: Incremental algorithms for hierarchical classification. In: Advances in Neural Information Processing Systems, vol. 17, pp. 233–240. MIT Press, Cambridge (2005)
[13] Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Hierarchical classification: Combining Bayes with SVM. In: Proc. of the 23rd Int. Conf. on Machine Learning, pp. 177–184. ACM Press, New York (2006)
[14] Gene Ontology Consortium: True path rule (2009), http://www.geneontology.org/GO.usage.shtml#truePathRule
[15] Valentini, G., Re, M.: Weighted True Path Rule: a multilabel hierarchical algorithm for gene function prediction. In: MLD-ECML 2009, 1st International Workshop on learning from Multi-Label Data, Bled, Slovenia, pp. 133–146 (2009)
[16] Valentini, G.: True Path Rule hierarchical ensembles for genome-wide gene function prediction. IEEE ACM Trans. on Comp. Biol. and Bioinformatics (in press)
[17] Lin, H., Lin, C., Weng, R.: A note on Platt's probabilistic outputs for support vector machines. Machine Learning 68, 267–276 (2007)
[18] Verspoor, K., Cohn, J., Mnizewski, S., Joslyn, C.: A categorization approach to automated ontological function annotation. Protein Science 15, 1544–1549 (2006)
[19] Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)

Recognizing Combinations of Facial Action Units with Different Intensity Using a Mixture of Hidden Markov Models and Neural Network

Mahmoud Khademi, Mohammad Taghi Manzuri-Shalmani,
Mohammad Hadi Kiapour, and Ali Akbar Kiaei

DSP Lab, Sharif University of Technology, Tehran, Iran
{khademi@ce.,manzuri@,kiapour@ee.,kiaei@ce.}sharif.edu

Abstract. Facial Action Coding System consists of 44 action units (AUs) and more than 7000 combinations. Hidden Markov models (HMMs) classifier has been used successfully to recognize facial action units (AUs) and expressions due to its ability to deal with AU dynamics. However, a separate HMM is necessary for each single AU and each AU combination. Since combinations of AU numbering in thousands, a more efficient method will be needed. In this paper an accurate real-time sequence-based system for representation and recognition of facial AUs is presented. Our system has the following characteristics: 1) employing a mixture of HMMs and neural network, we develop a novel accurate classifier, which can deal with AU dynamics, recognize subtle changes, and it is also robust to intensity variations, 2) although we use an HMM for each single AU only, by employing a neural network we can recognize each single and combination AU, and 3) using both geometric and appearance-based features, and applying efficient dimension reduction techniques, our system is robust to illumination changes and it can represent the temporal information involved in formation of the facial expressions. Extensive experiments on Cohn-Kanade database show the superiority of the proposed method, in comparison with other classifiers.

Keywords: classifier design and evaluation, data fusion, facial action units (AUs), hidden Markov models (HMMs), neural network (NN).

1 Introduction

Human face-to-face communication is a standard of perfection for developing a natural, robust, effective and flexible multi modal/media human-computer interface due to multimodality and multiplicity of its communication channels. In this type of communication, the facial expressions constitute the main modality [1]. In this regard, automatic facial expression analysis can use the facial signals as a new modality and it causes the interaction between human and computer more robust and flexible. Moreover, automatic facial expression analysis can be used in other areas such as lie detection, neurology, intelligent environments and clinical psychology.

Facial expression analysis includes both measurement of facial motion (e.g. mouth stretch or outer brow raiser) and recognition of expression (e.g. surprise or anger). Real-time fully automatic facial expression analysis is a challenging complex topic in

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 304–313, 2010.

computer vision due to pose variations, illumination variations, different age, gender, ethnicity, facial hair, occlusion, head motions, and lower intensity of expressions. Two survey papers summarized the work of facial expression analysis before year 1999 [2, 3]. Regardless of the face detection stage, a typical automatic facial expression analysis system consists of facial expression data extraction and facial expression classification stages. Facial feature processing may happen either holistically, where the face is processed as a whole, or locally. Holistic feature extraction methods are good at determining prevalent facial expressions, whereas local methods are able to detect subtle changes in small areas.

There are mainly two approaches for facial data extraction: geometric-based methods and appearance-based methods. The geometric facial features present the shape and locations of facial components. With appearance-based methods, image filters, e.g. Gabor wavelets, are applied to either the whole face or specific regions in a face image to extract a feature vector [4].

The sequence-based recognition method uses the temporal information of the sequences (typically from natural face towards the frame with maximum intensity) to recognize the expressions. To use the temporal information, the techniques such as hidden Markov models (HMMs) [5], recurrent neural networks [6] and rule-based classifier [7] were applied.

The facial action coding system (FACS) is a system developed by Ekman and Friesen [8] to detect subtle changes in facial features. The FACS is composed of 44 facial action units (AUs). 30 AUs of them are related to movement of a specific set of facial muscles: 12 for upper face and 18 for lower face (see Table 1).

Table 1. Some upper face and lower face action units (for more details see [8, 9])

Upper face AU	Description	Lower face AU	Description
AU 1	Inner brow raiser	AU 12	Lip corner puller
AU 2	Outer brow raiser	AU 15	Lip corner depressor
AU 4	Brow lowerer	AU 17	Chin raiser
AU 5	Upper lid raiser	AU 20	Lip stretcher
AU 6	Cheek raiser	AU 23	Lip tightener
AU 7	Lid tightener	AU 24	Lip pressor
Lower face AU	Description	AU 25	Lip parts
AU 9	Nose wrinkle	AU 26	Jaw drop
AU 10	Upper lip raiser	AU 27	Mouth stretch

Facial action units can occur in combinations and vary in intensity. Although the number of single action units is relatively small, more than 7000 different AU combinations have been observed. They may be *additive*, in which the combination does not change the appearance of the constituent single AUs, or *nonadditive,* in which the appearance of the constituent single AUs does change. To capture such subtlety of human emotion paralinguistic communication, automated recognition of fine-grained changes in facial expression is required (for more details see [8, 9]).

In this paper an accurate real-time sequence-based system for representation and recognition of facial action units is presented. We summarize the advantages of our system as follows:

1) We developed a classification scheme based on a mixture of HMMs and neural network, which can deal with AU dynamics, recognize subtle changes, and it is also robust to intensity variations.
2) HMMs classifier can deal with AU dynamics properly. But, it is impossible to use a separate HMM for each AU combination since the combinations numbering in thousands. We use an HMM for each single AU only. However, by employing a neural network we can recognize each single AU and each combination AU. Also, we use an accurate method for training the HMMs by considering the intensity of AUs.
3) Recent work suggests that spontaneous and deliberate facial expressions may be discriminated in term of timing parameters. Employing temporal information instead of using only the last frame, we can represent these parameters properly. Also, using both geometric and appearance features, we can increase the recognition rate and make the system robust against illumination changes.
4) By employing rule-based classifiers, we can automatically extract human interpretable classification rules to interpret each expression using continues values of AU intensity.
5) Due to the relatively low computational cost in the test phase, the proposed system is suitable for real-time applications.

The rest of the paper has been organized as follows: In section 2, we describe the approach which is used for facial data extraction and representation using both geometric and appearance features. Then, we discuss the proposed scheme for recognition of facial action units in section 3. Section 4 reports our experimental results, and section 5 presents conclusions and a discussion.

2 Facial Data Extraction and Representation

2.1 Geometric-Based Facial Feature Extraction Using Optical Flow

In order to extract geometric features, the points of a 113-point grid, which is called Wincandide-3, are placed on the first frame manually. Automatic registering of the grid with the face has been addressed in many literatures (e.g. see [10]). For upper face and lower face action units a particular subset of points are selected (see Fig. 1a). The pyramidal optical flow tracker [11] is employed to track the points of the model in the successive frames towards the last frame (see Fig. 1b). The loss of the tracked points is handled through a model deformation procedure (for details see [12]). For each frame, the displacements of the points in two directions with respect to the first frame are calculated to form a feature vector. Assume the last frames of the sequences in the training set, have the maximum intensity. Also, define:

$$\text{intensity(f)} = \frac{\text{sumdistances}_1\text{(f)}}{\text{sumdistances}_2} \tag{1}$$

where sumdistances_1 is the sum of the Euclidian distances between point of the Wincandide-3 grid in fth frame of the sequence and their positions in the first frame (a subset of points for upper face and lower face action units are used). Similarly, sumdistances_2 is the sum of the Euclidean distances between points of the model in the last frame of the sequence and their positions in the first frame; e.g. if the

sequence contains t frames, then $\text{intensity}(t) = 1$, and $\text{intensity}(1) = 0$. The frames of each sequence in the training set are divided into three subsets, i.e. three states, based on their intensity:

$0 \leq \text{intensity}(f) < 0.33$, $0.33 \leq \text{intensity}(f) \leq 0.66$, and $0.66 < \text{intensity}(f) \leq 1$

(For some of the single AUs we use five subsets (states)). Then, we apply principal component analysis (PCA) algorithm [13], separately to the feature vectors of these subsets to form the final geometric feature vectors of each state.

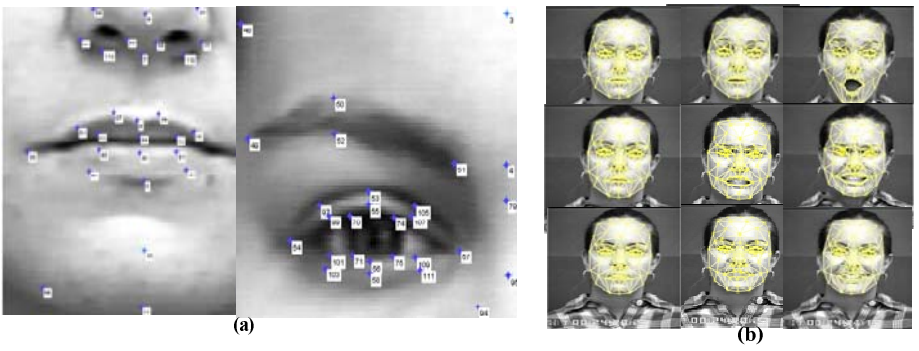
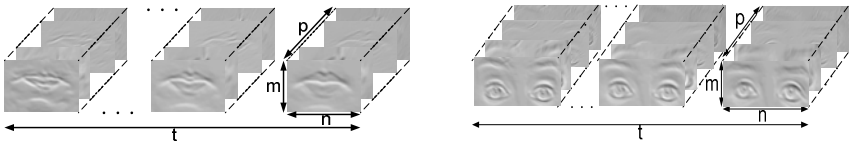


Fig. 1. a. Selected points for upper face and lower face action units. b. Geometric-based facial feature extraction using feature point tracking.

2.2 Appearance-Based Facial Feature Extraction Using Gabor Wavelets

In order to extract the appearance-based facial features from each frame, we use a set of Gabor wavelets. They allow detecting line endings and edge borders of each frame over multiple scales and with different orientations. Gabor wavelets remove also most of the variability in images that occur due to lighting changes [4]. Each frame is convolved with p wavelets to form the Gabor representation of the t frames (Fig. 2).



AU 17+AU 23+AU 24 AU 1+AU 2+AU 5

Fig. 2. Examples of the image sequences and their representation using Gabor wavelets

The frames of each sequence in the training set are divided into three or five subsets (states), based on grouping that we discussed in the previous subsection. In order to embed facial features in a low-dimensionality space and deal with curse of dimensionality dilemma, we should use a dimension reduction method. We apply 2D principal component analysis (2DPCA) algorithm [14] to each feature matrices of each subset separately. Then, we concatenate the vectorized representation of the reduced feature

matrices of each frame, and apply a 1DPCA [13] to them. The resulted feature vectors from each frame, are used for classification.

3 Facial Action Unit Recognition

The flow diagram of the proposed system is shown in Fig. 3. We will assume that sth ($s = 1,2,\ldots,M$) HMM is characterized by the following set of parameters (M is the number of HMMs, i.e. the number of single AUs):

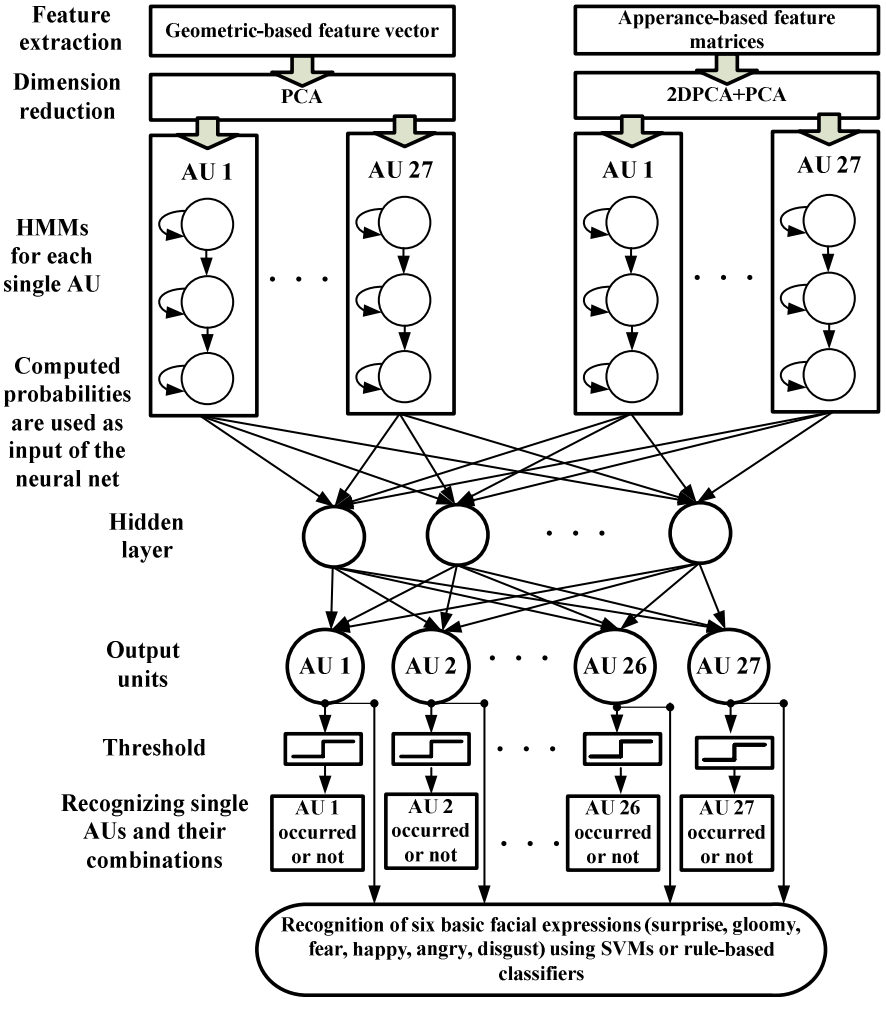


Fig. 3. Flow diagram of the proposed system

1) N_s, the number of the states of sth HMM (we use three or five-state left to right HMM for each single AU).
2) The probability densities $p(x \mid j)$ $(j = 1,2,\ldots,N_s)$, describing the distribution of the observations emitted from state j (we use three Gaussians for each state, the mean and variance of the Gaussians are computed using training data and maximum likelihood estimation method).
3) The transition probabilities $P(i \mid j)$ $(i,j = 1,2,\ldots,N_s)$, among the various states (the transition probabilities, which have not been depicted in left to right HMMs of Fig. 3, are zero, for others we use equal transition probabilities).
4) The probabilities $P(i)$ $(i = 1,2,\ldots,N_s)$, of the initial state ($P(1) = 1$ and $P(j) = 0$ for $j \neq 1$ because each sequence is started from natural face).
The training phase contains two steps: In the first step, we find probability density functions. Then, having observed sequence of feature vectors $X: x_1,\ldots,x_t$, (t is number of frames) in the respective sequence of states $\Omega_i{}^s: \omega_{i_1}{}^s,\ldots,\omega_{i_t}{}^s$, the quantity:

$$P(X|\Omega_*{}^s) = \max_i P(X|\Omega_i{}^s) = \max_i P(\omega_{i_1}{}^s)p(x_1|\omega_{i_1}{}^s)\prod_{k=2}^{N_s} P(\omega_{i_k}{}^s|\omega_{i_{k-1}}{}^s)p(x_k|\omega_{i_k}{}^s) \quad (2)$$

is computed for each of the M reference models. Its efficient computation can be achieved via Viterbi algorithm. In the second step of the training phase, we train the neural network by the values $P(X|\Omega_*{}^s)$ $(s = 1,\ldots,M)$ as the inputs and using back propagation algorithm. We trained the neural network with an output unit for each single AU and by allowing multiple output units to fire when the input sequence consists of AU combinations. The sth value of the target vector is 1 if sth single AU occurs in the corresponding image sequence, otherwise it would be 0). Moreover, we can use continuous values between 0 and 1 for sequences with different intensity of expression, as target value of the feature vectors. We also use some sequences several times with different intensities, i.e. by using intermediate frames as the last frame and removing the frames which come after it. Applying this method, we can properly recognize lower intensity and combinations of AUs. We model each single AU two times, using geometric and appearance features separately. In test phase, having observed sequence of feature vectors $X: x_1,\ldots,x_t$, in the sequence of states $\Omega_i{}^s: \omega_{i_1}{}^s,\ldots,\omega_{i_t}{}^s$, the quantities $P(X|\Omega_*{}^s)$ $(s = 1,\ldots,M)$ are first computed. Then, the outputs of the neural network are passed through a threshold (we set it 0.5). When several outputs are on, it signals that a combination of AUs has been occurred.

Although we can use a SVMs for classification of six basic facial expressions (by feature vectors directly or AU intensity values), employing rule-based classifiers such as JRIP [15], we can automatically extract human interpretable classification rules to interpret each expression. Thus, novel accurate AU-to-expression converters by continues values of the AU intensity can be created. These converters would be useful in animation, cognitive sciences, and behavioral sciences areas.

4 Experimental Results

To evaluate the performance of the proposed system and other methods like support vector machines (SVMs) [12], mixture of HMMs and SVMs, and neural network (NN) classifiers, we test them on Cohn-Kanade database [16]. The database includes 490 frontal view image sequences from over 97 subjects. The final frame of each image

sequence has been coded using Facial Action Coding System which describes subject's expression in terms of action units. For action units that vary in intensity, a 5-point ordinal scale has been used to measure the degree of muscle contraction. In order to test the algorithm in lower intensity situation, we used each sequence five times with different intensities, i.e. by using intermediate frames as the last frame. Of theses, 1500 sequences were used as the training set. Also, for upper face and lower face AUs, 240 and 280 sequences were used as the test set respectively. None of the test subjects appeared in training data set. Some of the sequences contained limited head motion.

Image sequences from neutral towards the frame with maximum intensity, were cropped into 57×102 and 52×157 pixel arrays for lower face and upper face action units respectively. To extract appearance features we applied 16 Gabor kernels to each frame. After applying the dimension reduction techniques, depending on the single AU that we want to model it, the geometric and appearance feature vectors were of dimension 6 to 10 and 40 to 60 respectively. The best performance was obtained by three and five states HMMs depend on the corresponding single AU. Table 2 and Table 3 show the upper face and lower face action unit recognition results respectively. In the proposed method, an average recognition rate of 90.0 and 96.1 percent were achieved for upper face and lower face action units respectively. Also, an average false alarm rate of 5.8 and 2.5 percent were achieved for upper face and lower face action units respectively.

Table 2. Upper face action unit recognition results (R=recognition rate, F=false alarm)

Proposed method (HMMs+NN)				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
1	20	18	1(1+2+4), 1(1+2)	0
2	10	8	2(1+2)	0
4	20	19	1(1+2+4)	0
5	20	20	0	0
6	20	19	0	1(7)
7	10	9	0	1(6)
1+2	40	37	2(2), 1(1+2+4)	0
1+2+4	20	18	1(1), 1(2)	0
1+2+5	10	8	2(1+2)	0
1+4	10	8	2(1+2+4)	0
1+6	10	8	1(1+6+7)	1(7)
4+5	20	18	1(4), 1(5)	0
6+7	30	27	2(1+6+7), 1(7)	0
Total	240	217	20	3
R	90.0%			
F	5.8%			

HMMs+SVMs				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
1	20	17	2(1+2+4), 1(1+2)	0
2	10	6	2(1+2+4), 1(1+2)	1(1)
4	20	18	1(1+2+4)	1(2)
5	20	20	0	0
6	20	18	1(1+6)	1(7)
7	10	7	3(6+7)	0
1+2	40	38	1(1+2+4)	1(4)
1+2+4	20	18	1(2), 1(1+2)	0
1+2+5	10	7	3(1+2)	0
1+4	10	5	3(1+2+4)	2(5)
1+6	10	6	2(1+6+7)	2(7)
4+5	20	15	2(4), 1(5)	2(2)
6+7	30	25	3(1+6+7), 2(7)	0
Total	240	200	30	10
R	83.3%			
F	12.5%			

SVMs				
AUs	Sequences	Recognized AUs		
		True	Missing or Extra	False
1	20	15	2(1+2+4), 1(1+2)	2(2)
2	10	6	2(1+2+4)	2(1)
4	20	18	1(1+2+4)	1(2)
5	20	20	0	0
6	20	19	1(1+6)	0
7	10	7	0	3(6)
1+2	40	35	1(2), 2(1+2+4)	2(4)
1+2+4	20	15	2(1), 2(2)	1(5)
1+2+5	10	6	2(1+5)	2(4)
1+4	10	4	3(1+2+4)	3(5)
1+6	10	6	3(1+6+7)	1(7)
4+5	20	15	2(1+2+5)	3(2)
6+7	30	24	2(1+6+7), 2(7)	2(1)
Total	240	190	28	22
R	79.2%			
F	17.1%			

NN [17]				
AUs	Sequences	Recognized AUs		
		True	Missing or Extra	False
1	20	14	3(1+2+4)	3(2)
2	10	5	4(1+2+4)	1(1)
4	20	18	1(1+2+4)	1(2)
5	20	18	1(4+5)	1(5)
6	20	18	2(1+6)	0
7	10	6	2(6+7)	2(6)
1+2	40	36	2(2), 2(1+2+4)	0
1+2+4	20	16	2(1), 2(2)	0
1+2+5	10	7	2(1+2)	1(4)
1+4	10	5	4(1+2+4)	1(5)
1+6	10	6	2(1+6+7)	2(7)
4+5	20	16	3(4)	1(1)
6+7	30	24	3(1+6+7), 3(7)	0
Total	240	189	38	13
R	78.8%			
F	15.4%			

In HMMs+SVMs method, for each single AU an HMM was trained. Then we classify the quantities $P(X|\Omega_*^s)$ $(s = 1, \ldots, M)$ using M two-class (occurred or not) SMVs classifiers [12] with Gaussian kernel.

Table 3. Lower facial action unit recognition results (R=recognition rate, F=false alarm)

Proposed method (HMMs+NN)				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
9	8	8	0	0
10	12	12	0	0
12	12	12	0	0
15	8	8	0	0
17	16	16	0	0
20	12	12	0	0
25	48	48	0	0
26	24	18	4(25+26)	2(25)
27	24	24	0	0
9+17	24	24	0	0
9+17+23+24	4	3	1(19+17+24)	0
9+25	4	4	0	0
10+17	8	5	2(17), 1(10)	0
10+15+17	4	4	0	0
10+25	8	8	0	0
12+25	16	16	0	0
12+26	8	7	1 (12+25)	0
15+17	16	16	0	0
17+23+24	8	8	0	0
20+25	16	16	0	0
Total	280	269	9	2
R	96.1%			
F	2.5%			

HMMs+SVMs				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
9	8	8	0	0
10	12	12	0	0
12	12	12	0	0
15	8	6	2(15+17)	0
17	16	16	0	0
20	12	12	0	0
25	48	45	1(25+26)	2(26)
26	24	19	3(25+26)	2(25)
27	24	24	0	0
9+17	24	22	2(9)	0
9+17+23+24	4	2	2(19+17+24)	0
9+25	4	4	0	0
10+17	8	3	2(10+12)	3(12)
10+15+17	4	2	2(15+17)	0
10+25	8	5	3(25)	0
12+25	16	16	0	0
12+26	8	5	2(12+25)	1(25)
15+17	16	16	0	0
17+23+24	8	7	1(17+23)	0
20+25	16	13	3(20+26)	0
Total	280	249	23	8
R	88.9%			
F	7.5%			

SVMs				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
9	8	8	0	0
10	12	8	2(10+7)	2(17)
12	12	12	0	0
15	8	6	2(15+17)	0
17	16	14	2(10+17)	0
20	12	12	0	0
25	48	43	2(25+26)	3(26)
26	24	18	3(25+26)	3(25)
27	24	24	0	0
9+17	24	22	2(9)	0
9+17+23+24	4	1	3(9+17+24)	0
9+25	4	4	0	0
10+17	8	2	4(10+12)	2(12)
10+15+17	4	2	2(15+17)	0
10+25	8	7	1(25)	0
12+25	16	16	0	0
12+26	8	3	3(12+25)	2(25)
15+17	16	16	0	0
17+23+24	8	6	2(17+24)	0
20+25	16	11	3(20+26)	2(26)
Total	280	235	31	14
R	83.9%			
F	12.5%			

NN [17]				
AUs	Sequences	Recognized AUs		
		True	Missing or extra	False
9	8	7	1(9+17)	0
10	12	8	2(10+7)	2(17)
12	12	11	1(12+25)	0
15	8	6	2(15+17)	0
17	16	13	2(10+17)	1(10)
20	12	12	0	0
25	48	42	3(25+26)	3(26)
26	24	19	2(25+26)	3(25)
27	24	23	1(27+25)	0
9+17	24	22	2(9)	0
9+17+23+24	4	1	3(9+17+24)	0
9+25	4	4	0	0
10+17	8	4	2(10+12)	2(12)
10+15+17	4	2	2(15+17)	0
10+25	8	7	1(25)	0
12+25	16	16	0	0
12+26	8	3	3(12+25)	2(25)
15+17	16	16	0	0
17+23+24	8	6	2(17+24)	0
20+25	16	12	3(20+26)	1(26)
Total	280	234	32	14
R	83.6%			
F	12.9%			

Although this method can deal with AU dynamics properly, due to use of crisp value for targets, this method suffers from intensity variations. In SVMs method, we first concatenated the reduced geometric and appearance feature vectors for each single AU. Then, we classify them using M two-class (occurred or not) SMVs classifiers with Gaussian kernel. This method cannot deal with AU dynamics. Moreover,

due to use of crisp value for targets, it suffers from intensity variations. Finally, in NN methods we trained a neural network (NN) with an output unit for each single AU and by allowing multiple output units to fire when the input sequence consists of AU combinations (like [17]). The best performance was obtained by one hidden layer. Although this method can deal with intensity variations, by using continues values for target of feature vectors, it suffers from trapping in local minima due to large number of inputs, i.e. feature vectors. Table 4 shows the facial expression recognition results using JRIP [15] classifier. We used several classifiers such as SVMs for classification of six basic facial expressions, but the results were almost the same. By applying each rule-based classifier we can develop an AU-to-expression converter.

Table 4. Facial expression recognition results using JRIP [15] classifier (S=surprise, G=gloomy, F=fear, H=happy, A=angry, D=disgust)

Confusion matrix for JRIP classifier (total number of samples=2916, correctly classified samples=2675 (91.74%), incorrectly classified samples=241 (8.26%)):

Classified as →	S	G	F	H	A	D
S	581	8	4	1	6	0
G	8	446	0	2	30	0
F	21	8	393	1	51	0
H	0	0	0	618	0	0
A	28	17	0	1	398	18
D	6	2	0	1	28	239

Detailed accuracy by class for JRIP classifier:

True positive rate	False positive rate	Precision	ROC area	Class
0.968	0.027	0.902	0.985	Surprise
0.918	0.014	0.927	0.991	Gloomy
0.829	0.002	0.990	0.980	Fear
1.000	0.003	0.990	1.000	Happy
0.861	0.047	0.776	0.958	Angry
0.866	0.007	0.930	0.982	Disgust

The resulted tree for converting the AU intensities to expressions using JRIP classifier (the value of each AU is between 0 and 1):

(AU9 >= 0.268941) and (AU24 == 0) and (AU12 == 0) => class=D (257.0/18.0)
(AU24 >= 0.5) and (AU23 >= 0.5) => class=A (198.0/0.0)
(AU1 == 0) and (AU12 == 0) and (AU15 == 0) and (AU20 == 0) and (AU26 <= 0.310026) and (AU27 == 0) and (AU7 >= 0.268941) => class=A (30.0/0.0)
(AU1 == 0) and (AU12 == 0) and (AU15 == 0) and (AU20 == 0) and (AU26 <= 0.083173) and (AU27 == 0) and (AU25 >= 0.064969) => class=A (233.0/78.0)
(AU20 >= 0.5) and (AU4 >= 0.5) => class=F (263.0/0.0)
(AU20 >= 0.390682) and (AU12 == 0) and (AU27 == 0) => class=F (133.0/0.0)
(AU23 >= 0.715669) and (AU16 >= 0.715669) => class=F (6.0/0.0)
(AU15 >= 0.017986) and (AU4 <= 0.907687) and (AU5 == 0) => class=G (301.0/1.0)
(AU4 >= 0.5) and (AU2 == 0) => class=G (162.0/17.0)
(AU25 == 0) and (AU1 >= 0.758204) and (AU2 <= 0.847391) => class=G (8.0/0.0)
(AU4 >= 0.929) and (AU5 == 0) => class=G (20.0/5.0)
(AU27 >= 0.064969) => class=S (456.0/6.0)
(AU12 == 0) and (AU26 >= 0.152609) => class=S (183.0/57.0)
(AU1 >= 0.898605) and (AU24 == 0) => class=S (14.0/1.0)
=> class=H (652.0/34.0)

5 Discussion and Conclusions

We proposed an accurate sequence-based system for representation and recognition of single and combinations of facial action units, which is robust to intensity and illumination variations. As an accurate tool, this system can be applied to many areas such as recognition of spontaneous and deliberate facial expressions, multi modal/media human computer interaction and lie detection efforts.

Although the computational cost of the proposed method can be high in the training phase, when the neural network were trained, it needs only some matrix products to reduce the dimensionality of the geometric and appearance features in the test phase. Employing a 3×3 Gabor kernel and a grid with low number of vertices, we can construct the Gabor representation of the input image sequence and also track the grid in less than two seconds with moderate computing power. As a result, the proposed system is suitable for real-time applications. Future research direction is to consider variations on face pose in the tracking algorithm.

Acknowledgment. The authors would like to thank the Robotic Institute of Carnegie Mellon University for allowing us to use their database. Thanks to referees for suggestions and for carefully reading this paper.

References

1. Mehrabian, A.: Communication without words. Psychology Today 2(4), 53–56 (1968)
2. Patnic, M., Rothkrantz, J.: Automatic analysis of facial expressions: the state of art. IEEE Transactions on PAMI 22(12) (2000)
3. Fasel, B., Luettin, J.: Automatic facial expression analysis: a survey. Pattern Recognition 36(1), 259–275 (2003)
4. Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with Gabor wavelets. In: 3rd IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 200–205 (1998)
5. Cohen, I., Sebe, N., Cozman, F., Cirelo, M., Huang, T.: Coding, analysis, interpretation, and recognition of facial expressions. Journal of Computer Vision and Image Understanding Special Issue on Face Recognition (2003)
6. Rosenblum, M., Yacoob, Y., Davis, L.: Human expression recognition from motion using a radial basis function network architecture. IEEE Transactions on Neural Network 7(5), 1121–1138 (1996)
7. Cohn, J., Kanade, T., Moriyama, T., Ambadar, Z., Xiao, J., Gao, J., Imamura, H.: A comparative study of alternative faces coding algorithms, Technical Report CMU-RI-TR-02-06, Robotics Institute, Carnegie Mellon University, Pittsburgh (2001)
8. Ekman, P., Friesen, W.: The facial action coding system: A technique for the measurment of facial movement. Consulting Psychologist Press, San Francisco (1978)
9. Tian, Y., Kanade, T., Cohn, F.: Recognizing action units for facial expression analysis. IEEE Transactions on PAMI 23(2) (2001)
10. Wiskott, L., Fellous, J.-M., Kuiger, N., Vonder Malsburg, C.: Face recognition by elastic bunch graph matching. IEEE Transactions on PAMI 19(7), 775–779 (1997)
11. Bouguet, J.: Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm, Technical Report, Intel Corporation, Microprocessor Research Labs (1999)
12. Kotsia, I., Pitas, I.: Facial expression recognition in image sequences using geometric deformation features and support vector machines. IEEE Transactions on Image Processing 16(1) (2007)
13. Aleix, M., Avinash, C.: PCA versus LDA. IEEE Transactions on PAMI 23(2) (2001)
14. Yang, D., Frangi, A., Yang, J.: Two-dimensional PCA: A new approach to appearance-based face representation and recognition. IEEE Transactions on PAMI 26(1), 131–137 (2004)
15. William, C.: Fast effective rule induction. In: Twelfth Int. Conf. on Machine Learning, pp. 115–123 (1995)
16. Kanade, T., Tian, Y.: Comprehensive database for facial expression analysis. In: IEEE In. Conf. on Face and Gesture Recognition, pp. 46–53 (2000)
17. Tian, Y., Kanade, T., Cohn, J.: Evaluation of Gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity. In: IEEE Int. Conf. on Automatic Face and Gesture Recognition (2002)

Some Thoughts at the Interface of Ensemble Methods and Feature Selection

Gavin Brown

We are in a very exciting time for Machine Learning. The field is making its first steps toward true industrial-strength technology, slowly transitioning from a disparate collection of techniques, to a mature science. Multiple Classifier Systems in particular, are showing repeated successes at the most competitive of levels: winning the Netflix challenge, forming the backbone of cutting edge real-time computer vision, and most recently steering Google's interests in quantum algorithms. It is thus becoming more and more difficult to generate truly meaningful contributions with our research. In the context of multiple classifier systems, we must ask ourselves, "how can we generate new MCS research that is truly *meaningful*?"

One answer to this is to reach out and apply the principles of classifier fusion in other areas, and see what those areas can provide in return. This talk will review how a perspective gained from working on problems in multiple classifier systems helped to solve a prominent question from *feature selection*—and, how in return, this is now providing interesting new directions for MCS. If we treat the classifier outputs as abstract random variables, irrespective of the algorithm which produced them, then *information theory* becomes a very natural language to work with. Parallels can be drawn between the definitions of feature "relevancy" and "redundancy" in feature selection, to the *accuracy-diversity* trade-off for classifier ensembles. This leads to a principled framework that unifies numerous feature selection criteria published since 1990; and, for MCS suggests a new way to characterize the ensemble "diversity" question. I will describe the perspectives and mathematical tools that lead up to this, and point out possible new directions in applying those tools to other aspects of the MCS field.

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, p. 314, 2010.

Multiple Classifier Systems for the Recogonition of Human Emotions

Friedhelm Schwenker, Stefan Scherer, Miriam Schmidt,
Martin Schels, and Michael Glodek

University of Ulm, Institute of Neural Information Processing, 89069 Ulm
friedhelm.schwenker@uni-ulm.de

Abstract. Research in the area of human-computer interaction (HCI) increasingly addressed the aspect of integrating some type of emotional intelligence in the system. Such systems must be able to recognize, interprete and create emotions. Although, human emotions are expressed through different modalities such as speech, facial expressions, hand or body gestures, most of the research in affective computing has been done in unimodal emotion recognition. Basically, a multimodal approach to emotion recognition should be more accurate and robust against missing or noisy data. We consider multiple classifier systems in this study for the classification of facial expressions, and additionally present a prototype of an audio-visual laughter detection system. Finally, a novel implementation of a Java process engine for pattern recognition and information fusion is described.

1 Introduction

Research in affective computing aim to provide simpler and more natural interfaces for human-computer interaction applications. In this kind of applications detecting and recognizing the emotional status of an user is important in order to develop efficient and productive human-computer interaction interfaces [5]. Analysis of human emotions and processing recorded data, for instance the speech, facial expressions, hand gestures, body movements, etc is a multi-disciplinary field that has been emerging as a rich area of research in recent times [6,14,19].

In this paper we focus on the pattern recognition approach of emotion recognition. Subsequently, we briefly describe two applications that have been developed in our lab recently. In Section 2 the classification of human emotions from facial expressions utilizing the hidden Markov model approach is discussed. Here we present results for various unimodal multiple classifer systems which have been applied to four different facial regions using three different input features. The Cohn-Kanade database was used for the numerical evaluation. Audio-visual laughter detection is the topic of Section 3. Audio features from the modulation spectrum and visual features derived from body and face movement were computed. These feature vectors serve as inputs to recurrent echo state networks which were trained to detect laughters. In the final section we present a novel Java process engine for information fusion and pattern recognition tasks which

N. El Gayar, J. Kittler, and F. Roli (Eds.): MCS 2010, LNCS 5997, pp. 315–324, 2010.

has been developed in the Institute of Neural Information Processing at the University of Ulm.

2 Hidden Markov Models for Facial Expression Recognition

A hidden Markov model (HMM) $\lambda = (Z, V, \pi, A, E)$ is a statistical model described through two random processes [15]. The first process is a Markov chain consisting of a finite number of states $Z = (z_1, ..z_n)$ and corresponding state transition probabilities given by a socalled transition matrix $A = (a_{ij})$. We simply used left-right-models in our work with $a_{ij} = 0$ for $i > j$ and $j > i + 1$, i.e. only forward connections were defined. The initial state probabilities are denoted as π. The second random process describes possible observations $V = (v_1...v_m)$ and the observation matrix $E = \{e_j(k)\}$ where $e_j(k)$ is the probability of an observation v_k being produced in state z_j. Typically, in this type of application are applied Gaussian mixture model (GMM) to estimate these emission probabilities. As HMMs were utilized for classification, one HMM λ_i, $i = 1, ..., n$ for each emotion class was trained by using all collected data of a particular class. The probabilities $P(O \mid \lambda_i)$ of the i-th HMM for an unclassified observation O are estimated using the socalled forward algorithm. The the most likely class is determined through a simple maximum detection among the probabilities $P(O \mid \lambda_i)$.

2.1 Classifier Fusion

Combining classifiers is a promising approach to improve classifier performance. Such a team of classifiers should be accurate and diverse [12]. While the requirement to the classifiers to be as accurate as possible is obvious, diversity roughly means that classifiers should not agree on the set of missclassified data. In this paper different feature views of the data are utilized in order to produce diversity.

Voting: The results of the classifiers at best lead all to the same class, but normally they are different. To nullify this problem, the first fusion method vote the results of selected classifiers.

Probabilistic-fusion: The other fusion approach did not combine the results but the probabilities $P(O \mid \lambda_i)$ of selected classifiers. To combine several classifiers, the class-wise products are computed. Only then is the maximum determined to obtain the most likely class. This implies statistic independence of the models which is unfortunately not fully given, since the features were generated from the same sequence.

2.2 Data Collection

The Cohn-Kanade dataset is a collection of image sequences with emotional content [4], which is available for research purposes. It contains 432 image sequences

which were recorded in a resolution of 640×480 pixels with a temporal resolution of 33 frames per second. Every sequence is played by an amateur actor who is filmed from a frontal view. The sequences always start with a neutral facial expression and end with the full blown emotion which is one of the six categories "joy", "anger", "surprise", "disgust", "sadness" or "fear".

To acquire a suitable label, the sequences were presented to 15 human labelers (13 male and two female). The sequences were presented as a video and the test persons were asked to select a label. Thus, a label for every sequence was determined through majority vote. The resulting data collection showed to be highly imbalanced: the class "joy" (105 samples) occurred four times more often than the class "fear" (25 samples) while "anger" (49 samples), "surprise" (91 samples), "disgust" and "sadness" (both 81 samples) are caught in between.

In all automatic facial expression recognition systems first some relevant features are extracted from the facial image and these feature vectors are utilized to train some type of classifier to recognize the facial expression. Finding the most relevant features is definitely the most important step in designing a recognition system. In our approach prominent facial regions such as the eyes, including the eyebrows, the mouth and for comparison the full facial region have been considered. For these four regions orientation histograms, principal components and optical flow features have been computed. Principal components (eigenfaces approach) are very well know in face recognition [25] and orientation histograms were successfully applied for the recognition of hand gestures and faces [22]. In order to extract the facial motion in these regions optical flow[1] features from pairs of consecutive images have been computed.

Table 1. Correlation between the number of states, the corresponding number of mixtures per state and the detection rate for all twelve models

no.	feature	region	no. st/mix	det. rate
1	PCA	face	9 / 2	0.764
2	PCA	mouth	8 / 2	0.685
3	PCA	right eye	7 / 2	0.456
4	PCA	left eye	3 / 2	0.451
5	Orientation histograms	face	4 / 2	0.704
6	Orientation histograms	mouth	4 / 3	0.729
7	Orientation histograms	right eye	4 / 2	0.500
8	Orientation histograms	left eye	9 / 2	0.479
9	Optical flow	face	8 / 2	0.639
10	Optical flow	mouth	9 / 2	0.607
11	Optical flow	right eye	7 / 3	0.442
12	Optical flow	left eye	8 / 4	0.491

[1] We were using a biologically inspired optical flow estimator, which was developed by the Vision and Perception Science Lab at the University of Ulm [1].

2.3 Experiments and Results

Numerical test runs were conducted with 10-fold cross-validation to adjust the number of Gaussian components, GMM components and HMM states. The results of this process are shown in Table 1. For a better overview the individual models were coded with the number of the row. The best single model with a total detection rate of 76.4 % was the model number 1 with PCA-features in the region face.

There was no model which was optimal for all emotions. For instance, model 1 recognized the emotion "joy" with 97.1 %, while it reached less than 25 % for the emotion "anger". On the other hand, model 12 showed different results: for emotion "anger" a detection rate of 63.3 % was obtained, but the detection rate for "joy" was lower than with model 1. To combine the advantages of these models and neutralize the disadvantages as possible, the decisions of the models were fused. In the following the results of the two fusion approaches will be presented:

Voting: A simple fusion approach is to vote the detected emotions of all twelve classifiers. If a test sequence is classified incorrectly from one model but the other eleven models detect it correctly, it will be outvoted. After the training of the individual classifiers, we investigated whether the classification rate could be improved, if not all twelve models would be combined. Therefore, all $2^{12} - 1$ model combinations were computed. The results of the top ten combinations are shown in Table 2. The appearance of the models 1 and 6 in all displayed combinations is originated from their good individual recognition performance. Models using the eye regions occur only sporadically in this list. This shows that these models reduced the classification rate because of their minor accuracy. The combination of all face and mouth regions led to a detection rate of 81.7 %. This means that, compared to the best single model, this combination has the ability to recognize 23 sequences more.

Probabilistic-fusion: This approach combines the posterior probabilities which are obtained from the six HMMs of each model. In many cases, when a sample was misclassified, the probability of the incorrect emotion was just slightly higher than the one of the correct emotion. The class-wise multiplication of the posterior probabilities has the potential to overcome the limitations of the early hardening in the vote-fusion. The results are also shown in Table 2. Again, the face and mouth regions played an important role. The result showing the highest classification rate of 86.1 % (372 out of 432) consisted of all models using the features of the regions face and mouth and the models 7 and 11. This was the highest achieved recognition rate in this study. Thus the probability-fusion could recognize 19 sequences more than the vote-fusion.

3 Audio-Visual Laughter Detection

Nonverbal discourse elements such as gestures, laughs, moans and other forms of back channeling, are elements of human to human interaction, which are

Table 2. Top ten combinations with vote-fusion and probability-fusion of the developed twelve single models. The coding of the models corresponds to the first column in Table 1.

	voting		probabilistic-fusion	
comb.	no. of models	det. rate	no. of models	det. rate
1	1 2 5 6 9 10	0.817	1 2 5 6 7 9 10 11	0.861
2	1 2 5 6 9	0.815	1 2 3 5 6 9 10 12	0.859
3	1 2 6 8 9 10	0.815	1 2 3 5 6 9 10 12	0.859
4	1 2 6 9 10 11	0.815	1 2 5 6 7 9 10 12	0.859
5	1 2 4 5 6 9 10	0.815	1 5 6 9	0.857
6	1 2 6 7 9 10	0.813	2 5 6 9 10 12	0.857
7	1 5 6 9 10 12	0.813	1 2 4 5 6 7 9	0.857
8	1 2 6 9 12	0.810	1 2 4 5 6 9 11	0.857
9	1 2 5 6 8 9 10	0.810	1 5 6 9 10	0.854
10	1 2 5 6 10 11 12	0.810	1 2 4 5 6 10 12	0.854

almost as important as direct verbal communication using speech. They convey essential information such as agreement, disagreement or simply confirmation of active listening in an efficient and natural way [2]. Furthermore, laughs are a great indicator of the positive perception of an ongoing conversation, or a symptom of uncertainty considering nervous or social laughter [3]. All in all laughs are a universal and frequent form of communicating with other humans [16,18,10,13]. Healthy and lively communication including nonverbal elements are important in face to face communication, and are therefore also crucial for the acceptance and believability of artificial agents, such as robots, or expert systems, with human like communication modalities, e.g. gestures, and speech [23]. Additionally, laughter is acoustically highly variable. It may be expressed in the form of giggles, exhaled or inhaled laughs, or snorted sounds, and thus, the spotting of laughs in continuous speech poses a challenging problem to pattern recognition [3,24].

3.1 Echo State Network Approach

In order to detect laughs in natural conversations a relatively novel kind of recurrent neural networks (RNN) is used, the so called Echo state network (ESN) [9]. ESN learning outperforms classical approaches such as RNN learning with its robustness towards noisy inputs [17] and its efficiency to adapt its weights [8]. Using the direct pseudo inverse adaptation method the ESN is trained in a very efficient way. The detailed training steps are described in the following paragraphs and can be further deepened by consulting the papers cited at the corresponding text passages. The ESN incorporates temporal dynamics for deciding whether or not laughter is present and also renders it a fitting candidate for the task to spot laughs in continuous conversations.

An ESN input layer K is fully connected to the dynamic reservoir M, and M is fully connected to the output layer L. ESNs are characterized by their dynamic memory that is realized by a sparsely (about 2% of the connections are set) interconnected reservoir M which is initialized randomly and comprises the recurrent connections. Loosely connected small cliques of neurons are sensitive to a certain dynamic within the data received through the input. A connection w_{ij} between neuron m_i and neuron m_j is represented by a weight $w_{ij} \neq 0$ in the connection matrix W. Due to the feedback and the recursive connections within the reservoir, not only the input is taken into account, but also the current state of each of the neurons and the history of all the inputs. Therefore, ESNs are an ideal candidate for encoding dynamic processes such as movement patterns or nonverbal utterances [17,18,11]. A spectral norm with a width parameter $\alpha < 1$ is applied to the randomly initialized connection matrix to guarantee that the activity within the reservoir decays to zero if no input is received. To train the ESN, the output weights W^{out} needs to be adapted towards the target signal T which may be done using the direct pseudo inverse method. The optimal values for the weights from the reservoir M to the output layer L can be computed by solving the linear equation system $W^{out} = (S^{+}T)^{t}$, where t indicates the transpose operation and S indicates a state collection matrix further described in the paragraphs below. The method minimizes the distance between the predicted output of the ESN and the target signal T. A detailed description for the calculation of the activation in the ESN is given in the following paragraphs:

Now we consider an ESN network with $|K|$ inputs, $|M|$ internal neurons and $|L|$ output neurons. Activations of input neurons at time step n are $U(n) = (u_1(n), \ldots, u_{|K|}(n))$, of internal units are $X(n) = (x_1(n), \ldots, x_{|M|}(n))$, and of output neurons are $Y(n) = (y_1(n), \ldots, y_{|L|}(n))$. Weights for the input connection in a $(|M| \times |K|)$ matrix are $W^{in} = (w^{in}_{ij})$, for the internal connection in a $(|M| \times |M|)$ matrix are $W = (w_{ij})$, and for the connection to the output neurons in an $|L| \times |M|$ matrix are $W^{out} = (w^{out}_{ij})$.

The activation of internal and output units is updated according to:

$$X(n+1) = f(W^{in}U(n+1) + WX(n)),$$

where $f = (f_1, \ldots, f_{|M|})$ are the internal neurons output sigmoid functions. The outputs are computed according to:

$$Y(n+1) = f^{out}(W^{out}X(n+1)),$$

where $f^{out} = (f^{out}_1, \ldots, f^{out}_{|L|})$ are the output neurons output sigmoid functions.

A detailed description of the offline learning procedure is given in the following:

1. Given I/O training sequence $(U(n), D(n))$
2. Generate randomly the matrices (W^{in}, W, W^{out}), scaling the weight matrix W such that it's maximum eingenvalue $|\lambda_{max}| \leq 1$.

3. Drive the network using the training I/O training data, by computing

$$X(n+1) = f(W^{in}U(n+1) + WX(n))$$

4. Collect at each time the state $X(n)$ as a new row into a state collection matrix S, and collect similarly at each time the sigmoid-inverted teacher output $tanh^{-1}D(n)$ into a teacher collection matrix T.
5. Compute the pseudo inverse S^+ of S and put

$$W^{out} = (S^+T)^t$$

3.2 Utilized Features and Data

The data used in the experiments is comprised of two different modalities, namely audio and video. A conversation between four people sitting around a table recorded with an unobtrusive 360 degree camera and a centrally placed microphone served as the basis of the experiments. Suitable features for the laugh detection task were extracted from both modalities. In the following two short paragraphs the extraction procedures are explained in some detail.

From the audio channel modulation spectrum features (MODSPEC), which also found their application in several other tasks including emotion recognition, were extracted [7]. They are computed using standard methods like Mel filtering and fast Fourier transformations (FFT). In short they represent the rate of change of frequency, since they are based on a two level Fourier transformation. Furthermore, they are biologically inspired since the data is split up into perceptually scaled bands using a common Mel filter bank. The perception ability of the human auditory system is optimized for the slow temporal modulations of speech. Earlier studies reported that the MODSPEC components from the range between 2 and 16 Hz, with dominant component at around 4 Hz, contain important linguistic information. Dominant components represent strong rate of change of the vocal tract shape in the corresponding MODSPEC frequency.

From the video relatively coarse features were extracted from the 360 degree recordings of the centrally placed camera. The face tracking using the well established Viola Jones approach provided data comprising coordinates of the faces at each frame. However, since these coordinates are highly dependent on the distance of the person to the camera and relative movement, data of the face and upper torso including arm movement were taken as input to the classifier. The body related features were easy to extract, since people were seated around a table and the assumption that bodies are usually below faces seemed sound. The coordinates were normalized to movement data of a mean value of 0 and a standard deviation of 1. Therefore, the movement ranged from -1 to 1 for each tracked face and body individually.

3.3 Experimental Results

Trying to take advantage of the sequential dynamic characteristics of the features as well as the nature of the conversation, ESNs are utilized to conduct the

experiments. The ESNs are composed by a dynamic reservoir of 1500 neurons, sparsely interconnected with each other. The probability for a connection between neuron m_i and m_j in the reservoir as well as the recursive connections is set to 2%, which empirically proofed to be the optimal connectivity. The spectral width α is crucial for the behavior of the network, since it strongly influences the dynamics of the reservoir. It has been evaluated that the spectral width $\alpha = 0.15$ performs best for the given problem. The ESNs are trained to return an output of 1 if a laugh is present and -1 if not.

As mentioned before features from two modalities are used in the experiments. Since the two modalities are sampled at different rates (50 Hz audio features; 10 Hz video features) a decision fusion architecture was chosen. Two separate ESNs, one for each modality, are used to spot laughter in their respective modality and the decisions are merged by simply adding the outputs. A final decision if a laugh is present or not for each frame is done by comparing the merged output to a threshold. Using this multimodal approach a recognition rate of 91% was achieved in comparison to the unimodal approaches, which were outperformed clearly (audio only: 87%; video only: 82%). In two control experiments GMM super vector approach and a simple GMM approach were conducted for comparison. Both control architectures were outperformed by the multimodal ESN approach, which may be explained by the fact that in contrast to the ESNs they do not take the dynamics of the features into account. Their error rates are 28% for the GMM super vectors and 16% for the GMM approach. For a more detailed discussion it is recommended to sift through [18].

4 Implementation of Multiple Classifier Systems

In many of the aforementioned applications such as the detection or spotting of laughs in continuous conversations it is crucial to be able to utilize the resources of multiple machines and cores in order to fusion and incorporate data from multiple sensors in real-time. Therefore, a novel process engine for pattern recognition and information fusion tasks, namely the *pepr framework*, has been introduced recently [20,21] and is available online under the Apache license at: `www.pepr-framework.org`. The process engine enables the development of applications for real-time tasks on an abstraction level above common programming languages, and thus reducing the time and costs necessary for the implementation of such solutions, allowing and encouraging the use of established and well tested components. The highly expandable framework allows easy adaptations for many unique challenges and is equipped with a user interface for the creation of solutions in a graph model. This graphical representation of the processes serves rapid prototyping and a better overview in complicated applications.

A sample application could again be the multimodal detection of laughs of two or more people talking via several machines with each other using an enhanced video conference application as introduced in [20]. In this task, all machines process video and audio in real-time and join the data of the other machines via timestamps using external libraries integrated in the framework. For example,

OpenCV, libsvm, and even Matlab, are integrated and render the framework a powerful tool for multi-machine/multi-sensor applications. The technical details of the framework, however, would exceed the scope of the current paper and therefore the reader is advised to review details in forthcoming publications and user manuals listed on www.pepr-framework.org.

Acknowledgments

The authors would like to thank Prof. Dr. H. Neumann, Dr. P. Bayerl and S. Ringbauer from the Vision and Perception Science Lab of the Institute of Neural Processing at the University of Ulm for generous assistance in extracting the optical flow features. This paper is based on work done within the "Information Fusion" subproject of the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" and DFG project SCHW 623/4-3, both funded by the German Research Foundation. The work of Martin Schels is supported by a scholarship of the Carl-Zeiss Foundation.

References

1. Bayerl, P., Neumann, H.: A fast biologically inspired algorithm for recurrent motion estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 29, 246–260 (2007)
2. Bousmalis, K., Mehu, M., Pantic, M.: Spotting agreement and disagreement: A survey of nonverbal audiovisual cues and tools. In: Proceedings of the International Conference on Affective Computing and Intelligent Interaction, vol. 2, pp. 121–129 (2009)
3. Campbell, N., Kashioka, H., Ohara, R.: No laughing matter. In: Proceedings of Interspeech, pp. 465–468. ISCA (2005)
4. Cohn, J.F., Kanade, T., Tian, Y.: Comprehensive database for facial expression analysis. In: Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition, pp. 46–53 (2000)
5. Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., Taylor, J.: Emotion recognition in human-computer interaction. IEEE Signal Processing Magazine 18(1), 32–80 (2001)
6. Devillers, L., Vidrascu, L., Lamel, L.: Challanges in real-life emotion annotation and machine learning based detection. Neural Networks 18, 407–422 (2005)
7. Hermansky, H.: The modulation spectrum in automatic recognition of speech. In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 140–147. IEEE, Los Alamitos (1997)
8. Jaeger, H.: Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. Tech. Rep. 159, Fraunhofer-Gesellschaft, St. Augustin Germany (2002)
9. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304, 78–80 (2004)
10. Knox, M., Mirghafori, N.: Automatic laughter detection using neural networks. In: Proceedings of Interspeech 2007, pp. 2973–2976. ISCA (2007)

11. Krause, A.F., Blaesing, B., Duerr, V., Schack, T.: Direct control of an active tactile sensor using echo state networks direct control of an active tactile sensor using echo state networks. In: Ritter, H., Sagerer, G., Dillmann, R., Buss, M. (eds.) Proceedings of 3rd International Workshop on Human-Centered Robotic Systems (HCRS 2009). Cognitive Systems Monographs, pp. 11–21. Springer, Heidelberg (2009)
12. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. 51(2), 181–207 (2003)
13. Laskowski, K.: Modeling vocal interaction for text-independent detection of involvement hotspots in multi-party meetings. In: Proceedings of the 2nd IEEE/ISCA/ACL Workshop on Spoken Language Technology (SLT2008), pp. 81–84 (2008)
14. Oudeyer, P.Y.: The production and recognition of emotions in speech: features and algorithms. International Journal of Human Computer Interaction 59(1-2), 157–183 (2003)
15. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 257–286 (1989)
16. Scherer, S., Campbell, W.N.: Automatic laughter detection for measuring discourse engagement. In: Autumn Meeting of the Acoustical Society of Japan 2008 (ASJ 2008), pp. 265–266 (2008) (in japanese)
17. Scherer, S., Oubbati, M., Schwenker, F., Palm, G.: Real-time emotion recognition from speech using echo state networks. In: Prevost, L., Marinai, S., Schwenker, F. (eds.) ANNPR 2008. LNCS (LNAI), vol. 5064, pp. 205–216. Springer, Heidelberg (2008)
18. Scherer, S., Schwenker, F., Campbell, W.N., Palm, G.: Multimodal laughter detection in natural discourses. In: Ritter, H., Sagerer, G., Dillmann, R., Buss, M. (eds.) Proceedings of 3rd International Workshop on Human-Centered Robotic Systems (HCRS 2009). Cognitive Systems Monographs, pp. 111–121 (2009)
19. Scherer, S., Schwenker, F., Palm, G.: Classifier fusion for emotion recognition from speech. In: Proceedings of Intelligent Environments 2007, pp. 152–155 (2007)
20. Scherer, S., Fritzsch, V., Schwenker, F.: Multimodal real-time conversation analysis using a novel process engine. In: Proceedings of International Conference on Affective Computing and Intelligent Interaction 2009 (ACII 2009), pp. 253–255. IEEE, Los Alamitos (2009)
21. Scherer, S., Fritzsch, V., Schwenker, F., Campbell, N.: Demonstrating laughter detection in natural discourses. In: Interdisciplinary Workshop on Laughter and other Interactional Vocalisations in Speech (2009)
22. Schwenker, F., Sachs, A., Palm, G., Kestler, H.A.: Orientation histograms for face recognition. In: ANNPR, pp. 253–259 (2006)
23. Strauss, P.M., Hoffmann, H., Scherer, S.: Evaluation and user acceptance of a dialogue system using wizard-of-oz recordings. In: 3rd IET International Conference on Intelligent Environments 2007 (IE 2007), pp. 521–524. IEEE, Los Alamitos (2007)
24. Truong, K.P., Van Leeuwen, D.A.: Evaluating laughter segmentation in meetings with acoustic and acoustic-phonetic features. In: Workshop on the Phonetics of Laughter, Saarbrücken, pp. 49–53 (2007)
25. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience 3(1), 71–86 (1991)

Erratum to Biggio, B., Fumera, G., Roli, F., "Multiple classifier systems for adversarial classification tasks." In Benediktsson, J.A., Kittler, J., Roli, F., eds.: MCS. Volume 5519 of Lecture Notes in Computer Science., Springer (2009) 132141

In the above mentioned paper we reported in a plot (Fig. 3) the classification cost and the robustness to evasion of a single classifier and different MCSs obtained according to Eqs. (3)-(8). The classification cost was measured as $FP + FN/C$ (y-axis) for several C values (1,2,10,100). The robustness (x-axis) was measured as the expected value of the minimum number of (Boolean) features which have to be modified on malicious samples to evade a classifier. We compared a linear classifier (with identical weights) trained on $n = 300$ i.i.d. features with four MCSs obtained by performing a logical OR on the class labels outputted by N linear classifiers trained on n/N features, with N from 2 to 5. In Fig. 3, the single classifier was shown to be more accurate than the MCSs at any given operating point, when the classifiers were not under attack, but it was also easier to evade. Moreover, while the MCS accuracy decreased for increasing ensemble sizes, the robustness to evasion increased. However Fig. 3 was wrong, due to an error on computing the values of Eq. 8. In Fig. 1 the correct plot is shown. It can be seen that the MCSs exhibit both lower accuracy and lower robustness than the single classifier. We point out, however, that the experimental results were correct: the MCS considered in the experiments turned out to be more robust than the single classifier.

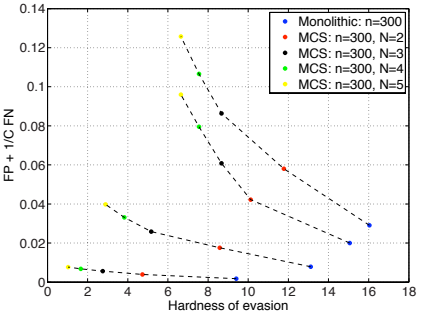


Fig. 1. Classification cost $FP + \frac{1}{C}FN$ as a function of the robustness for the single classifier and four MCSs with ensemble size $N = 2, ..., 5$, $n = 300$ features, and four C values. Each dashed line corresponds to a different C value: from top to bottom, $C = 1, 2, 10, 100$.

Author Index