The 18th European Conference on Artificial Intelligence

Proceedings

Workshop on Supervised and Unsupervised Ensemble Methods and their Applications

Monday July 21, 2008

Tuesday July 22, 2008

Patras, Greece

Oleg Okun and Giorgio Valentini

Table of Contents

Preface

Welcome to the 2nd Workshop on **S**upervised and **U**nsupervised **E**nsemble **M**ethods and their **A**pplications (SUEMA) to be held in conjunction with the 18th European Conference on Artificial Intelligence (Patras, Greece) on 21-22 July, 2008. This workshop follows a similar event organized last year in Spain. The goal of SUEMA workshops is to provide the forum for informal constructive discussion and the exchange of ideas among a small group of scientists working in the field of ensemble methods. Nevertheless, this forum does not intend to be exclusively reserved for a small group of selected persons.

Despite of its small format, SUEMA attracted scholars and researchers from diverse areas of ensemble methods. However, what is important for the whole field, in general, and for SUEMA, in particular, is the application-driven research reported in many papers. This means that ensemble methods have found their ways in solving various practical tasks. We thank all authors for their contributions that embrace a wide area of applications of ensemble methods.

This year, SUEMA is proud to host our PASCAL2 invited speaker, Ludmila Kuncheva from UK, who is the author of the well-known book "Combining Pattern Classifiers. Methods and Algorithms", published by Wiley in 2004 and probably the first-ever book exclusively devoted to ensemble methods. PASCAL2 (Pattern Analysis and Statistical Modelling and Computational Learning) is a Network of Excellence funded by the Seventh Framework Programme of the European Union. Its main research focus is on the emerging challenges created by the ever expanding applications of adaptive systems technology and their central role in the development of large scale cognitive systems. From this standpoint the research topics of the workshop largely fall within the scope of PASCAL2.

The SUEMA program comprises 13 accepted papers submitted by scientists from UK, Italy, France, USA, Austria, Germany, Greece, Spain, Lithuania, and Portugal, with the largest part come from Italy (4 papers).

The invited talk by Ludmila Kuncheva will present an overview and perspectives of applying classifier ensembles for detecting concept change or drift in data streams. This is a challenging and hard to solve real-world problem since accurate decisions must be made on-line as data arrive.

Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas will present an overview of ensemble selection approaches intended to reduce the number of predictive models in an ensemble while improving ensemble performance. It is known that if the number of models is too large, an ensemble can suffer from time/memory problems and hence it will not scale well to large datasets.

Battista Biggio, Giorgio Fumera, and Fabio Roli discuss ensemble applications for security, in particular, for spam filtering. They provide the formal explanation of why ensembles succeed, i.e. why ensembles are more secure to use than a single classifier. Their explanation is based on the elements of game theory.

Francesco Gargiulo, Antonio Penta, Antonio Picariello, and Carlo Sansone continue the spam filtering theme by using a behavior-knowledge space approach for spam detection. Spam is only one of the examples of dynamically changing data.

Pedro Gago and Manuel Filipe Santos analyze the concept drift in medical services (e.g. intensive care units) with an algorithm inspired by the dynamic weighted majority voting.

Facial expression recognition including multiple classes of expressions can be very difficult even for humans to perform. Nevertheless, Terry Windeatt, Raymond Smith, and Kaushala Dias will show how to successfully apply the weighted decoding ECOC (error-correcting-output-coding) for this task.

Sebastian Nusser, Clemens Otte, and Werner Hauptmann concentrate on imbalanced misclassification costs in multi-class safety-related problems. They will demonstrate that a hierarchy of such costs when taken into account can reduce/avoid inconsistencies in ensemble decisions, caused by multiple classes.

Manifold learning by the tensor voting framework used in various computer vision tasks forms the basis of the neighbors voting algorithm to be presented by Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. The result of their approach is particularly interesting to see in image patching/continuation, where a part of the image is missing or deleted and the task is to reconstruct the missing/deleted part so that it will be visually consistent with the rest of the image.

Ensembles are successful not only in classification but also in data clustering, and a number of papers prove this point. Carlotta Domeniconi and Muna Al-Razgan explore the combination of clustering ensembles and semi-supervised clustering, aided by a set of constraints obtained directly from the data in order to address the ill-posed nature of clustering.

Sébastien Derivaux, Germain Forestier, and Cédric Wemmert discuss how supervised learning can benefit from multiple clusterings.

Angelo Ciaramella, Giulio Giunta, Angelo Riccio, and Stefano Galmarini propose a hierarchical agglomerative algorithm to clusterize models tailored to the prediction of the distribution of pollutants, comparing the different distributions predicted by the models using the negentropy and the Kullback-Leibler divergence.

One of the main reasons for ensemble success is the diversity of predictions made by the algorithms composing an ensemble. Georg Krempl and Vera Hofer explore the combination of boosting and arbitrating, which they call partitioner trees, for improving the performance of specific classifier ensembles, for instance, such as those of the generalized linear models.

Diversity injection into ensembles of decision trees is considered by Jesús Maudes, Juan Rodríguez, and César García-Osorio; their approach is called disturbing neighbors, since it adds extra features, based on the nearest neighbor rule, to the original feature vector.

Erinija Pranckeviciene addresses the problem of how to construct a diverse ensemble by integrating feature selection into ensemble training.

Oleg Okun and Giorgio Valentini

Acknowledgments

The workshop chairs would like to thank the PASCAL 2 Network of Excellence and the Dipartimento di Scienze dell'Informazione (University of Milan, Italy) for the partial financial support of this event.

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

Classifier Ensembles for Detecting Concept Change in Streaming Data: Overview and Perspectives

Ludmila I. Kuncheva[1]

Abstract. We address adaptive classification of streaming data in the presence of concept change. An overview of the machine learning approaches reveals a deficit of methods for explicit change detection. Typically, classifier ensembles designed for changing environments do not have a bespoke change detector. Here we take a systematic look at the types of changes in streaming data and at the current approaches and techniques in online classification. Classifier ensembles for change detection are discussed. An example is carried through to illustrate individual and ensemble change detectors for both unlabelled and labelled data. While this paper does not offer ready-made solutions, it outlines possibilities for novel approaches to classification of streaming data.

Keywords: Concept drift, Change detection, Changing Environments, Classifier ensembles, Online classifiers

1 INTRODUCTION

It has been argued that the current state-of-the-art in pattern recognition and machine learning is falling short to answer modern challenges in classification [12]. Among these challenges is classification of streaming data, especially when the data distribution changes over time. The ideal classification scenario is to detect the changes when they come, and retrain the classifier automatically to suit the new distributions. Most methods for novelty detection rely on some form of modelling of the probability distributions of the data and monitoring the likelihood of the new-coming observations [27]. These methods work well for small number of features and static distributions. They typically require access to all past data without strong considerations about processing time. Detecting changes in multi-dimensional streaming data has been gaining speed in recent years [25, 36]. The most useful indicator of adverse change, however, is a peak or a steady upward trend in the running error of the online classifier (or classifier ensemble). Thus the change detection method receives as input a binary string (correct/wrong label), and raises an alarm if a "sizable" increase in the error rate has occurred. Simple as it looks, the problem of designing a reliable and, at the same time, sensitive change detector is far from being solved.

2 A LANDMAP OF CLASSIFICATION APPROACHES FOR STREAMING DATA

The approaches to handling concept drift in streaming data can be categorised with respect to the following properties

- *Instances versus batches of data.* Streaming instances of data can be converted into streaming batches or "chunks" of data. The reverse is also possible but batch data usually comes in massive quantities, and instance-based processing may be too time-consuming.
- *Explicit versus implicit change detection.* After explicit change detection, action is taken, for example, setting up a window of latest data to re-train the classifier. Implicit detection is equivalent to using a forgetting heuristic regardless of whether or not change is suspected. For example, using an online classifier ensemble, the weights of the ensemble members are modified after each instance, based on the recent accuracy records of the ensemble members. Without concept drift, the classification accuracy will be stable and the weights will converge. Conversely, if change does occur, the weights will change to reflect it without the need of explicit detection.
- *Classifier-specific versus classifier-free.* In the classifier-specific case, the forgetting mechanism can only be applied to a chosen and fixed classifier model [1, 3] or classifier ensemble [30]. In the classifier-free case any classifier can be used because the change detection and the window update depend only on the accuracy of the classification decision, and not on the model [10, 20].
- *Classifier ensembles versus single classifiers.* A categorisation of classifier ensemble methods for changing environment is offered in [22]. The methods are grouped with respect to how they adapt to the concept drift: by updating the combination rule for fixed classifiers ("horse racing"); by using adaptive online classifiers as the ensemble members; and by changing the ensemble structure ("replace the loser").

Explicit or implicit detection of concept drift can be based upon change in:

A. Probability distributions. If the class-conditional distributions or prior probabilities for the classes drift away from their initial values, the new data will not fit the old distributions. Based on how well the assumed distribution accommodates most recent data, a change can be detected and old data should be forgotten [4,7,9,11,27,33]. Methods for change detection in this case include estimating the likelihood of new data with respect to the assumed distributions, and comparing the likelihood with a threshold.

B. Feature relevance. A concept drift may lead to a different relevance pattern of the features describing the instances. Features or even combinations of attribute values that were relevant in the past may no longer be sufficiently discriminative [3, 8, 13, 40]. Keeping track on the best combination of predictive features (clues) makes it possible to train an up-to-date classifier for the most recent data distribution.

[1] School Computer Science, Bangor University, United Kingdom, e-mail: l.i.kuncheva@bangor.ac.uk

C. Model complexity. Some classifier models are sensitive to change in the data distribution. For example, explosion of the number of rules in rule-based classifiers or the number of support vectors in SVM classifiers may signify concept drift.

D. Classification accuracy. This is the most widely used criterion for implicit or explicit change detection. Included in this group are most ensemble methods for changing environments: Winnow variants [24, 28], AdaBoost variants [5, 30], "replace-the-loser" approaches [21,22,34,35,39]. Many single classifier models also evaluate the accuracy either to select the window size for the next classifier [2, 10, 18–20, 23] or to update the current model [1, 15, 41].

3 TYPES OF CHANGES AND THEIR DETECTION

Change may come as a result of the changing environment of the problem (we call this "novelty"), e.g., floating probability distributions, migrating clusters of data, loss of old and appearance of new classes and/or features, class label swaps, etc.

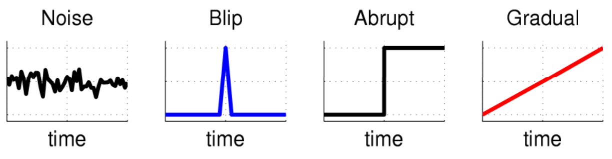


Figure 1. Types of concept change in streaming data

Figure 1 shows four examples of simple changes that may occur in a single variable along time. The first plot (Noise) shows changes that are deemed non-significant and are perceived as noise. The classifier should not respond to minor fluctuations, and can use the noisy data to improve its robustness for the underlying stationary distribution. The second plot (Blip) represents a 'rare event'. Rare events can be regarded as outliers in a static distribution. Examples of such events include anomalies in landfill gas emission, fraudulent card transactions, network intrusion and rare medical conditions. Finding a rare event in streaming data can signify the onset of a concept drift. Hence the methods for online detection of rare events can be a component of the novelty detection paradigm. The last two plots in Figure 1 (Abrupt) and (Gradual) show typical examples of the two major types of concept drift represented in a single dimension.

3.1 Rare events detection in batch data

Finding outliers in batch 1-dimensional static data has a longstanding history within statistical literature [14]. Hodge and Austin [14] point out that there is no universally accepted definition of outlier, and that alternative terminologies pervade in the recent literature, e.g., novelty detection, anomaly detection, noise detection, deviation detection or exception mining. Here we adopt the following definition: "An outlier is an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data." In this simplest set up, one can model the relevant probability distributions and calculate the likelihood of the suspect observations. A threshold on this likelihood will determine whether an observation should be marked as an outlier. Outlier detection methods for static data typically assume unlimited (finite) computation time, and unrestricted access to every observation in the data.

In multi-dimensional static data, outliers are not as obvious. Such outliers, e.g., fraudulent telephone calls, may need to be labelled by hand and communicated to the system designer for an outlier detection algorithm to be trained. The main difficulty compared to the one-dimensional case is that the approximation of multi-dimensional probability density functions is not as trivial. Detecting outliers in multidimensional batch data can be approached as a pattern recognition task. Assuming that outliers are very rare, the data can be used to train the so called "one-class classifier". In a way, the training will lead to a form of approximation of the pdf for class "norm" or will evaluate geometrical boundaries in the multi-dimensional space, beyond which an observation is labelled as an outlier [16,26,38]. Alternatively, the probability density functions of the individual variables can be modelled more precisely, and an ensemble of one-class classifiers may be more accurate in identifying rare events [6, 37].

3.2 Novelty (concept drift) detection in streaming data

For streaming data, the one-dimensional change detection has been extensively studied in engineering for the purposes of process quality control (control charts). Consider a streaming line of objects with probability p of an object being defective. Samples of N objects (batches) are taken for inspection at regular intervals. The number of defective objects is counted and an estimate $\hat{p}$ is plotted on the chart. It is assumed that the true value of p is known (from product specification, trading standards, etc.) Using a threshold of $f\sigma$, where $\sigma = \sqrt{p(1-p)/N}$, a change is detected if $\hat{p} > p + f\sigma$. This model is known as *Shewhart* control chart, or also p-chart when binary data is being monitored. The typical value of f is 3, but many other alternative and compound criteria have been used[2]. Better results have been reported with the so called CUSUM charts (CUmulative SUM) in terms of detecting small changes [29,32]. Reynolds and Stoumbos [31] advocate another detection scheme based on Wald's Sequential Probability Ratio Test (SPRT). SPRT charts are claimed to be even more successful than CUSUM charts.

Finding rare events in multi-dimensional streaming data, and especially when changing environments may be present, has not been systematically explored. While there are tailor-made solutions for specific problems, new technologies are still in demand. The prospective solutions may draw upon sources of information A-D detailed in Section 2.

4 CLASSIFIER ENSEMBLES FOR CHANGE DETECTION

Note that the change-detection process is separate from the classification. Below we illustrate the operation of the classifier equipped with a change detector. Upon receiving a new data point **x** the following set of actions takes place before proceeding with the next point in the stream:

(1) The classifier predicts a label for **x**.

(2) The true label is received.

(3) **x**, the predicted and the true labels are submitted to the change detector.

(4) If change is detected, the classifier is re-trained.

The classification of a data point from the stream is shown diagrammatically in Figure 2.

[2] `http://en.wikipedia.org/wiki/Control_chart`

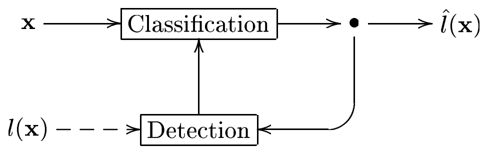


Figure 2. Online classification with explicit detection of concept change. Notations: $\hat{l}(\mathbf{x})$ is the class label assigned by the classification block (a single classifier or an ensemble); $l(\mathbf{x})$ is the true label of $\mathbf{x}$ received after the classification. The result of the change detection is an action related to the classifier training.

The ideal detector will signal a change as soon as it occurs (true positive) and will not signal false changes. In practice, the data needed to detect the change come after the change, so a delay of Δ observations is incurred. The expected number of batches of data to detection is called the Average Run Length (ARL) in control chart literature. Here we are interested in the average number of observations to detection. The desired characteristics of a change detector are $\Delta_{\text{true }+} \to 0$ and $\Delta_{\text{false }+} \to \infty$. Hence change detectors can be compared with respect to their Δs

While classification ensembles are a popular choice now for classification of streaming data with concept drift, explicit change detection is usually only mentioned on the side. We are interested in the change detector, and in the possibility to implement it as a "detection ensemble".

4.1 Detection from unlabelled data

Changes in the unconditional probability distribution may or may not render the current classifier inaccurate. Consider the following example. In a classification problem with two equiprobable Gaussian classes, both classes migrate apart, symmetrically about the optimal discriminant line. Suppose that the perfect classifier has been trained on the original distributions. Even though there are changes in the underlying probability density function (pdf), the classifier need not change as it will stay optimal.

When will change detection from unlabelled data be useful? Here are three possible answers:

1. *More sensitive to error.* When change in the distributions can be detected early to herald a future error change.
2. *Label delay.* The labels are not available at the time of classification but come with a substantial delay. For example, in applying for a bank loan, the true class label (good credit/bad credit) becomes known two years after the classification has taken place [17].
3. *Missed opportunities.* The chance to improve the classifier may be easily ignored if only labelled data is used for the change detection. It is always assumed that a change will adversely affect the classifier, giving raise to the classification error. Consider again the example with the two equiprobable Gaussian classes. Suppose now that one of the classes migrates away from the discrimination line, while the other stays in its original place. The classification error of the initial classifier will drop a little because there will be much fewer errors from the migrated class. In this case, however, if change is not detected, we are missing the opportunity to train a new classifier with much smaller error.

The most popular method for detecting change from unlabelled streaming data is to monitor the cluster structure of the data and signal a change when a "notable" change occurs, e.g. migration of clusters, merging, appearance of new clusters, etc. The monitoring can be done using a sliding window of fixed size to re-calculate the cluster parameters. These parameters are then compared with the previous ones. If the difference exceeds a given threshold, change is signalled. More formally, the cluster-monitoring approach can be cast as fitting a mixture of Gaussians and monitoring the parameters, e.g., means of the components (centres of the clusters), covariance matrices (shapes of the clusters) or mixture coefficients (cluster density). The likelihood of the sample in the window can be calculated. There are at least two parameters of the algorithm that need to be set up in advance: the size of the sliding window and the threshold for comparison of the latest likelihood with the old one. This makes way for an ensemble approach.

An ensemble for change detection from unlabelled data can be constructed by putting together detectors with different parameter values. An example is discussed next. Figure 3 shows the "old" and the "new" distributions.

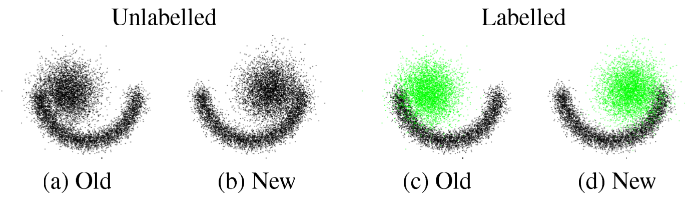


Figure 3. Old and new distributions, with and without labels

First, 400 i.i.d data points are generated from the old distribution, followed by 400 points from the new distribution. The classes (black and green) are generated with equal prior probabilities. The following protocol is used. The first M observations are taken as the first window W_1. K-means clustering is run on this data, where the number of clusters, K is fixed in advance (we chose $M = 20$, $K = 2$). It can be shown that, under some basic assumptions, the log-likelihood of the data in window W_i is related to the sum of the distances of the data points to their nearest cluster centre. We compare the mean log-likelihood of window W_i (L_i) with the mean log-likelihood of the stream so far ($\bar{L}_i$). Change is signalled if $L_i < \bar{L}_i + 3\sigma_i/\sqrt{M}$, where σ_i is the standard deviation of the log-likelihood up to point i. The cluster means are updated after each observation. Figure 4 shows the running mean (total average, $\bar{L}_i$), the mean of the log-likelihood of the moving window (L_i), the change point at observation 400, and the detected changes.

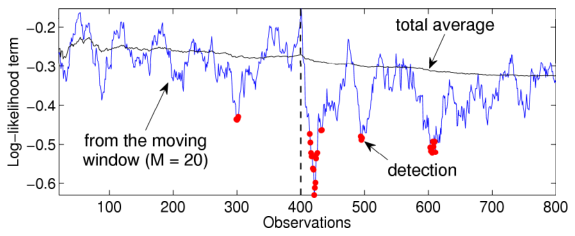


Figure 4. Change detection (single detector) from unlabelled data using the likelihood of a data window of size $M = 20$.

The detection pattern is not perfect. There were 4 false detections in this run (observations 299-302), coming before the true change. On the other hand, after the change, consistent detection is signalled for all observations from 414 to 426. Later on, since there is no forgetting in this scenario, the means start moving towards the second distribution, and the log-likelihood will eventually stabilise.

The benefit of using an ensemble is demonstrated in Figure 5.

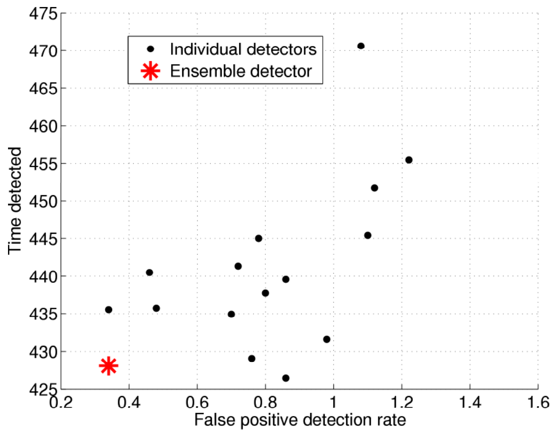


Figure 5. Single and ensemble detectors: **unlabelled** data

All values of $M = \{5, 10, 15, 20, 25\}$ and $K = \{2, 3, 4\}$ were tried, resulting in 15 individual detectors. Fifty runs were carried out with each combination (M, K) with different data generated form the distributions in question. The dots in the figure correspond to the individual detectors, where the false positive rate and the true positive detection times were averaged across the 50 runs. An ensemble was created by pooling all 15 individual detectors. The ensemble signals a detection if at least two individual detectors signal detection. The star in the figure shows the ensemble detector. Better detectors are those that have both small false positive rate and small time to detection $\Delta_{\text{true}+}$. The ensemble has a joint smallest false detection rate and a second smallest time to detection. The two together make the ensemble the most desirable detector compared to the individual detectors.

4.2 Detection from labelled data

The strongest change indication is a sudden or gradual drop in the classification accuracy, consistent over a period of time. An ensemble detector can make use of different window sizes, thresholds or detections heuristics. To illustrate this point we continue the example shown in Figure 3, this time with the true class labels supplied immediately after classification. For the purposes of the illustration we chose the nearest mean classifier (NMC), in spite of the fact that it is not optimal for the problem. The same values of M were used as in the detection from unlabelled data. NMC was built on the first window W_1 and updated with the new observations without any forgetting. The guessed label of observation i was obtained from the current version of NMC. The true label was received and the error was recorded (0 for no error, 1 for error). NMC was retrained using observation i and the true label. The running error was the error of NMC built on all the data from the start up to point i. The window of size M containing the error records of the past M observations was used to estimate the current error and compare it with the mean error hitherto. The 3 sigma threshold was used again. The error rate does not increase dramatically after observation 400 where the old and the new distributions are swapped. Hence the change detection patterns were erratic, often not signalling a change throughout the whole run. The ensemble was constructed by pooling the detectors for the 5 different values of M. The ensemble signals a change at observation i if at least one of the detectors signals a change. Figure 6 shows the individual detector scores and the ensemble score for the labelled data case. The same two axes are used: false positive rate and the detection time (Note that Detection time = 400 + $\Delta_{\text{true}+}$). The numbers in brackets show in how many out of the 50 runs the respective detector has missed the change altogether. The ensemble detector is once again better than the single detectors, being closest to the bottom left corner of the plot.

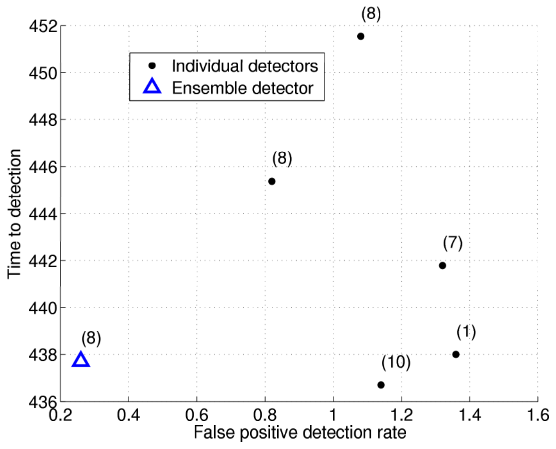


Figure 6. Single and ensemble detectors: **labelled** data. The numbers in brackets show in how many out of the 50 runs the respective detector has missed the change altogether.

The control chart detection methods fall in the category of detectors from labelled data. Ensembles of control charts may be an interesting research direction. Based on the currently available theoretical fundament, desirable ensemble properties may turn out to be theoretically provable.

5 CONCLUSIONS

The true potential of ensembles for change detection lies in the ability of different detection modes and information sources to sense different types and magnitudes of change. The purpose of this study was not to offer answers but rather to open new questions. The idea of constructing an ensemble for change detection was suggested and illustrated using both labelled and unlabelled data. The detection ensembles could be homogeneous (using the same detection methods with different parameters) as well as heterogeneous (detection based on labelled and unlabelled data, feature relevance, and more). The detector may be used in conjunction with the classifier. For example, the detector may propose how long after the detection the old classifier can be more useful than an undertrained new classifier. With gradual changes, the detector can be used to relate the magnitude of the change with the size of a variable training window for the classifier. Various problem-specific heuristics can be used to combine the individual detectors. Control chart methods come with a solid body of theoretical results that may be used in designing the ensemble detectors.

One possibility which was not discussed here is to couple detectors with the members of the classifier ensemble responsible for the labelling of the streaming data. Thus various changes can be handled by different ensemble members, and a single overall change need not be explicitly signalled.

REFERENCES

[1] D. W. Aha, D. Kibler, and M. K. Albert, 'Instance-based learning algorithms', *Machine Learning*, **6**, 37–66, (1991).

[2] M. Baena-García, J. Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, 'Early drift detection method', in *Fourth International Workshop on Knowledge Discovery from Data Streams*, pp. 77–86, (2006).

[3] A. Blum, 'Empirical support for Winnow and weighted-majority based algorithms: results on a calendar scheduling domain', in *Proc. 12th International Conference on Machine Learning*, pp. 64–72, San Francisco, CA., (1995). Morgan Kaufmann.

[4] S. J. Delany, P. Cunningham, and A. Tsymbal, 'A comparison of ensemble and case-based maintenance techniques for handling concept drift in spam filtering', Technical Report TCD-CS-2005-19, Trinity College Dublin, (2005).

[5] W. Fan, S. J. Stolfo, and J. Zhang, 'Application of adaboost for distributed, scalable and on-line learning', in *Proc KDD-99*, San Diego, CA, (1999). ACM Press.

[6] T. Fawcett and F. J. Provost, 'Adaptive fraud detection', *Data Mining and Knowledge Discovery*, **1**(3), 291–316, (1997).

[7] G. Folino, C. Pizzuti, and G. Spezzano, 'Mining distributed evolving data streams using fractal GP ensembles', in *Proceedings of EuroGP*, volume LNCS 4445, pp. 160–169. Springer, (2007).

[8] G. Forman, 'Tackling concept drift by temporal inductive transfer', Technical Report HPL-2006-20(R.1), HP Laboratories Palo Alto, (June 2006).

[9] M.M Gaber and P.S. Yu, 'Classification of changes in evolving data streams using online clustering result deviation', in *Third International Workshop on Knowledge Discovery in Data Streams*, Pittsburgh PA, USA, (2006).

[10] J. Gama, Pedro Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection', in *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence*, volume 3171 of *Lecture Notes in Computer Science*, pp. 286–295. Springer Verlag, (2004).

[11] V. Ganti, J. Gehrke, and R. Ramakrishnan, 'Mining data streams under block evolution', *ACM SIGKDD Explorations Newsletter*, **3**, 1–10, (2002).

[12] D.J. Hand, 'Classifier technology and the illusion of progress (with discussion)', *Statistical Science*, **21**, 1–34, (2006).

[13] M. Harries and K. Horn, 'Detecting concept drift in financial time series prediction using symbolic machine learning', in *Eighth Australian Joint Conference on Artificial Intelligence*, pp. 91–98, Singapore, (1995). World Scientific Publishing.

[14] V.J. Hodge and J. Austin, 'A survey of outlier detection methodologies', *Artificial Intelligence Review*, **22**(2), 85–126, (2004).

[15] G. Hulten, L. Spencer, and P. Domingos, 'Mining time-changing data streams', in *In Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 97–106. ACM Press, (2001).

[16] P. Juszczak, *Learning to recognise. A study on one-class classification and active learning*, Ph.D. dissertation, Delft University of Technology, 2006.

[17] M. G. Kelly, D. J. Hand, and N. M. Adams, 'The impact of changing populations on classifier performance', in *KDD '99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 367–371, (1999).

[18] R. Klinkenberg, 'Using labeled and unlabeled data to learn drifting concepts', in *Workshop notes of the IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, pp. 16–24, Menlo Park, CA, USA, (2001). IJCAI, AAAI Press.

[19] R. Klinkenberg and T. Joachims, 'Detecting concept drift with support vector machines', in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pp. 487–494, San Francisco, CA, USA, (2000). Morgan Kaufmann.

[20] R. Klinkenberg and I. Renz, 'Adaptive information filtering: Learning in the presence of concept drifts', in *AAAI-98/ICML-98 workshop Learning for Text Categorization*, Menlo Park,CA, (1998).

[21] J. Z. Kolter and M. A. Maloof, 'Dynamic weighted majority: A new ensemble method for tracking concept drift', in *Proc 3rd International IEEE Conference on Data Mining*, pp. 123–130, Los Alamitos, CA, (2003). IEEE Press.

[22] L. I. Kuncheva, 'Classifier ensembles for changing environments', in *5th International Workshop on Multiple Classifier Systems, MCS 04, LNCS*, volume 3077, pp. 1–15. Springer-Verlag, (2004).

[23] M.M. Lazarescu and S. Venkatesh, 'Using selective memory to track concept drift effectively', in *Intelligent Systems and Control*, volume 388, Salzburg, Austria, (2003). ACTA Press.

[24] N. Littlestone, 'Learning quickly when irrelevant attributes abound: A new linear threshold algorithm', *Machine Learning*, **2**, 285–318, (1988).

[25] J. Ma and S. Perkins, 'Online novelty detection on temporal sequences', in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 613–618. ACM Press, (2003).

[26] L. M. Manevitz and M. Yousef, 'One-class svms for document classification', *Journal of Machine Learning Research*, **2**, 139–154, (2001).

[27] M. Markou and S. Singh, 'Novelty detection: A review, Part I: Statistical approaches', *Signal Processing*, **83**(12), 2481– 2521, (2003).

[28] C. Mesterharm, 'Tracking linear-threshold concepts with winnow', *Journal of Machine Learning Research*, **4**, 819–838, (2003).

[29] E. S. Page, 'Continuous inspection schemes', *Biometrika*, **41**, 100–114, (1954).

[30] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, 'Learn++: An incremental learning algorithm for supervised neural networks', *IEEE Transactions on Systems, Man and Cybernetics*, **31**(4), 497–508, (2001).

[31] M. R. Reynolds Jr and Z. G. Stoumbos, 'The SPRT chart for monitoring a proportion', *IIE Transactions*, **30**, 545–561, (1998).

[32] M. R. Reynolds Jr and Z. G. Stoumbos, 'A general approach to modeling CUSUM charts for a proportion', *IIE Transactions*, **32**, 515–535, (2000).

[33] M. Salganicoff, 'Density-adaptive learning and forgetting', in *Proceedings of the 10th International Conference on Machine Learning*, pp. 276–283, (1993).

[34] K. O. Stanley, 'Learning concept drift with a committee of decision trees', Technical Report AI-03-302, Computer Science Department, University of Texas-Austin., (2003).

[35] W. N. Street and Y. S. Kim, 'A streaming ensemble algorithm (SEA) for large-scale classification', in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 377–382. ACM Press, (2001).

[36] J. Takeuchi and K. Yamanishi, 'A unifying framework for detecting outliers and change points from time series data', *IEEE Transactions on Knowledge and Data Engineering*, **18**(4), 482–492, (2006).

[37] D.M. Tax and R.P.W. Duin, 'Combining one-class classifiers', in *Proc. of the 2nd Int Workshop on MCS*, volume LNCS 2096, pp. 299–308. Springer, (2001).

[38] D.M.J. Tax, *One-class classification; Concept-learning in the absence of counter-examples*, Ph.D. dissertation, Delft University of Technology, 2001.

[39] H. Wang, W. Fan, P. S. Yu, and J. Han, 'Mining concept drifting data streams using ensemble classifiers', in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 226–235. ACM Press, (2003).

[40] G. Widmer, 'Tracking context changes through meta-learning', *Machine Learning*, **27**(3), 259–286, (1997).

[41] G. Widmer and M. Kubat, 'Learning in the presence of concept drift and hidden contexts', *Machine Learning*, **23**, 69–101, (1996).

Evade Hard Multiple Classifier Systems

Battista Biggio and **Giorgio Fumera** and **Fabio Roli** [1]

Abstract. Multiple classifier systems are widely used in security applications like biometric personal authentication, spam filtering, and intrusion detection in computer networks. Several works experimentally showed their effectiveness in these tasks. However, their use in such applications is motivated only by intuitive and qualitative arguments. In this work we give a first possible formal explanation of why multiple classifier systems are harder to evade, and therefore more secure, than a system based on a single classifier. To this end, we exploit a theoretical framework recently proposed to model adversarial classification problems. A case study in spam filtering illustrates our theoretical findings.

1 Introduction

The effectiveness of multiple classifier systems has been proven in several real applications [4, 3]. Various authors showed that using classifier ensembles can allow to improve the detection capability in security applications like biometric authentication, and intrusion detection in computer networks. The classifier ensemble approach is also used in commercial and open source spam filters. A short survey of the literature on these applications is reported in section 2. For the purposes of this work, it is important to note that the use of multiple classifier systems have been also proposed to improve the so called "hardness of evasion" of a security system, namely, to increase the effort that the attacker has to do to evade the system [6]. However, the main motivations proposed so far in favour of the classifier ensemble approach in security applications are indeed based on qualitative and intuitive arguments, besides the experimental evidence. In particular, to our knowledge, no work tried to develop formal arguments to analyse the improvements attainable by multiple classifier systems on the hardness of evasion.

The aim of this work is to make a first step in this direction. We consider a theoretical framework recently proposed in [1] to model *adversarial classification* problems, namely, problems in which a system based on machine learning techniques is used against an *adaptive* adversary which can modify malicious patterns to evade the system. This is clearly the case of many security applications, including the ones mentioned above. This framework is summarised in section 3.1. In section 3.2 we use it to model the particular case in which the system is made up of an ensemble of classifiers, and new classifiers can be added to the system in response to adversarial actions aimed at evading it. This allow us to give a possible formal explanation of why multiple classifiers are harder to evade than a single classifier. These findings are then experimentally investigated in section 4 with a case study in spam filtering, using the SpamAssassin open source spam filter and a real corpus of legitimate and spam e-mails.

[1] Dept. of Electrical and Electronic Engineering, University of Cagliari, Piazza d'Armi, 09123 Cagliari, Italy, e-mails: {battista.biggio, fumera, roli}@diee.unica.it

2 Previous works on multiple classifiers for security applications

The use of classifier ensembles to improve the detection accuracy or the hardness of evasion has been recently proposed by several authors in different security applications [4, 3, 6, 2, 7]. It also turns out that the design of well-known spam filters like SpamAssassin (`http://spamassassin.apache.org`) and intrusion detection systems (IDSs) like Snort (`http://www.snort.org/`) implicitly follows the approach based on combining an ensemble of detectors.

SpamAssassin and Snort are based on a similar rationale. They consist of a set of classification rules ("test") in the if-then form, which analyse different characteristics of input patterns (respectively e-mails and network packets) to detect the presence of "signatures" denoting a malicious origin of the pattern. Each test outputs a score which denotes the "likelihood" that the pattern is malicious. Tests are often unrelated to each other, in the sense that they are based on unrelated characteristics of the input pattern. In some cases, they are focused on specific signatures of known attacks. In the case of SpamAssassin, the scores provided by the tests which *fired* (namely, the ones whose antecedent part is true for the processed e-mail) are summed up. The final decision about the input pattern (to be labelled either as malicious or as innocent) is taken by thresholding the sum of the scores. [2] This architecture makes easy to add new tests (i.e., binary classification rules) or delete old ones, to keep SpamAssassin and Snort updated and to customise them to the specific traffic of the network on which they operate. We point out that SpamAssassin and Snort can be considered as instances of classifier ensembles. Each rule can indeed be viewed as a single classifier based on a specific set of features, whose score is then fused with the ones of the other classifiers through the well known sum rule [4]. This kind of ensemble-like modular architecture is used also in commercial spam filters and IDSs.

The classifier ensemble approach has been explicitly investigated in the literature of IDSs, as an alternative to the approach based on a "monolithic" classifier [2, 6]. The ensemble architecture proposed in [2] and [6] is similar: it consists in fusing classifiers each trained on a distinct feature representation of patterns. The main motivations are derived from the field of classifier ensembles. It is indeed known that, if different sets of heterogeneous and loosely correlated features are available, as happens for IDSs, combining the outputs of different classifiers trained on different feature sets can be more effective than

[2] Actually Snort rules give a boolean output, and the decision function is a logic OR between all the outputs. Nevertheless, this is equivalent to having a binary score, say 0 and 1, and to threshold the sum of such scores with a value between 0 and 1.

designing a single classifier in the feature space made up of all the available features (see for instance [4]). Moreover, if the overall number of features is large, such a single classifier would be more prone than a classifier ensemble to the so-called curse of dimensionality problem. In [2] it was also pointed out that the ensemble approach "reflects the behaviour of network security experts, who usually look at different traffic statistics in order to produce reliable attack signatures". The above arguments qualitatively support the use of classifier ensembles to improve the effectiveness of an IDS, i.e. to attain both low false alarm rates and high attack detection rates. In [6] it was also pointed out that classifier ensembles can allow to improve the hardness of evasion. The reason is that fusing classifiers that work on different feature spaces "forces the attacker to devise a mimicry attack that evades multiple models of normal traffic at the same time, which is intuitively harder than evading just one model".

Classifier ensembles have been very investigated also in biometric applications, in which they can be straightforwardly exploited in several ways [7]. For instance, it is intuitive that using different biometric traits (like face, fingerprints, and speech) the recognition accuracy of a system as well as its hardness of evasion can be improved. With regard to the hardness of evasion, the use of multiple classifiers using different biometric traits strongly discourages fraudulent attempts to deceive personal identity verification systems. In fact, deceiving a multi-modal biometric system would require the construction of different kinds of fake biometric traits, which is a very challenging task. However, no work theoretically analysed the hardness of evasion of biometric systems that use the classifier ensemble approach.

As can be seen from the above overview, so far the hardness of evasion of security systems based on MCSs is not motivated theoretically, but is rather intuitively suggested by the fact that the MCS architecture naturally allows adding new classifiers (e.g., new filtering rules in SpamAssassin) in response to adversarial actions aimed at evading it, and experimental evidences show that adding new classifiers, using different features, makes more difficult for the attacker to evade the system. Experience and intuition also suggest the designer of these security systems that the characteristics which allow detecting malicious patterns can be very different and heterogeneous, and can change over time due to new tricks used by spammers and hackers to defeat spam filters and IDSs. Also this motivates the heuristic strategy commonly used to increase the hardness of evasion by adding new classifiers using different input features. In the next section we will address this open issue, providing a first theoretical justification of the classifier ensemble approach as tool for improving the hardness of evasion.

3 Why are multiple classifiers harder to evade?

In the literature, the first attempt to formalise a scenario in which an adaptive adversary tries to defeat a system based on machine learning techniques was made in [1]. The authors formalised this class of problems under the statistical framework of the minimum risk theory, as a two-class classification problem in which a classifier has to discriminate between positive (malicious) and negative (innocent) instances (e.g., spam and legitimate e-mails, genuine or impostor users in access control systems, attack or normal packets in computer network traffic, etc.), while the adversary can modify either training or testing instances to defeat the classifier. Assuming that both the classifier and the adversary are guided by utility and cost functions, the model proposed in [1] allows to show how the adversary should compute the optimal strategy for modifying instances, and how the classifier should take this into account by modifying its decision function accordingly. This can also allow to evaluate the effectiveness of a given strategy followed by the classifier to improve its detection capability and its hardness of evasion. This analytical framework is summarised in section 3.1. In section 3.2 we will exploit it to analyse the case when the system is made up of an ensemble of classifiers working on different feature sets, and the decision function is obtained by thresholding the sum of the scores provided by each classifier. This classifier architecture is commonly used in spam filters, IDSs and biometric authentication systems, as described in section 2. Our aim is to give a support based on a formal model to the strategy used to improve the detection capability and the hardness of evasion in these kinds of systems, based on adding new classifiers each working on homogeneous features, different from the ones used by other classifiers.

3.1 A theoretical framework for adversarial classification

When machine learning or pattern recognition techniques are used in applications like spam filtering, intrusion detection, biometric authentication, etc., their task can be formalised as a two-class classification problem. Denoting with y the class label, instances belong either to a positive ($y = +$) or to a negative class ($y = -$). The positive class is made up of malicious instances, while innocent or legitimate instances belong to the negative class. Instances are considered as realisations of a random variable X, and are represented as vectors of N feature values which are random variables themselves, $(X_1, \ldots, X_i, \ldots, X_N)$. A realisation of such random variable is denoted as $x = (x_1, \ldots, x_i, \ldots, x_N)$, where x_i is a possible value for feature X_i. It is assumed that instances are generated i.i.d. according to a given distribution $P(X)$, which can be rewritten as $P(X) = P(X|+)P(+) + P(X|-)P(-)$. The set of all possible realisations of X defines the feature space $\mathcal{X}$. In [1] it is assumed that the adversary can modify only positive instances at the operation phase, to make them being misclassified as legitimate by the classifier, but it can not modify any negative instance nor positive instances belonging to the training set. We point out that this assumption is true for several real applications. For instance, in the spam filtering task, where instances correspond to e-mails, spammers can modify only positive instances (their own e-mails) to evade the anti-spam filter, but they can not modify legitimate e-mails or any training instance, given that the training process is usually carried out *offline*, over a hand-labelled corpora of e-mails. There are however some cases, like IDSs trained *online*, in which the adversary can modify training instances: the model in [1] can not be directly applied to these cases.

In [1] it is then assumed that the classifier and the adversary act according to given utility and cost functions. Denoting with $y_C(x)$ the label given to the instance x by the classifier, the utility function of the classifier is represented as $U_C(y_C, y)$. It is reasonable to assume that such utility is positive for correctly classified instances and negative for misclassified ones, that is, $U_C(+,+) > 0$, $U_C(-,-) > 0$, $U_C(+,-) \leq 0$ and $U_C(-,+) \leq 0$. The cost for the classifier is assumed to be due to measuring each feature. The cost for measuring the i-th feature is denoted as V_i. Finally, it should be taken into account that the adversary could modify any positive instance x using some function $\mathcal{A}(x)$. For example, in spam filtering the function $\mathcal{A}(x)$ could be implemented by adding words or using synonyms in the spam mails. It follows that the expected utility for the classifier is given by

$$U_{\mathcal{C}} = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P(x,y)[U_C(y_{\mathcal{C}}(\mathcal{C}(\mathcal{A}(x))),y) - \sum_{i=1}^{N} V_i], \quad (1)$$

where $\mathcal{Y} = \{+,-\}$ and $P(x,y)$ is the joint probability of pattern x being generated with true label y (note that, by the above assumptions, $\mathcal{A}(x) = x$ if $y = -$, namely for each negative instance).

Similarly, denoting with $U_A(y_c, y)$ the utility function of the adversary, it is assumed that it is positive for positive instances misclassified by the classifier as legitimate ($U_A(-,+) > 0$), negative for correctly classified positive instances ($U_A(+,+) \leq 0$), and zero for negative instances ($U_A(-,-) = U_A(+,-) = 0$) whatever the label assigned by the classifier, namely, the adversary utility is not affected by the classification of negative instances. The cost for the adversary is that faced for modifying an instance x according to the function $\mathcal{A}(x)$ defined above. It is assumed that the cost is given by $W(x,\mathcal{A}(x)) = \sum_{i=1}^{N} W_i(x,\mathcal{A}(x))$, being W_i the cost for modifying the i-th feature. Of course, $W_i = 0$ if the i-th feature is not changed, $W_i > 0$ otherwise. The expected utility for the adversary is thus

$$U_{\mathcal{A}} = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} P(x,y)[U_A(y_{\mathcal{C}}(\mathcal{C}(\mathcal{A}(x))),y) - W(x,\mathcal{A}(x))]. \quad (2)$$

Using the above model, the adversarial classification problem is formulated as a game between classifier and adversary, in which the two players make one move at a time. A move by the classifier consists in choosing a decision function $y_{\mathcal{C}}(\cdot)$, taking into account both the training set and any knowledge it may have about the strategy $\mathcal{A}(\cdot)$ defined in the previous move by the adversary. Analogously, a move by the adversary consists in choosing a strategy $\mathcal{A}(\cdot)$, taking into account the available knowledge about the decision function chosen by the classifier at the previous move. Although game theory could in principle be applied to find the optimal sequence of moves by both players according to their utility and cost functions, it was shown in [1] that this is computational intractable, and anyway it requires the knowledge of the distribution $P(x,y)$, which is unrealistic. Therefore, a simplified single-shot version of the game was considered in [1]. Initially, classifier constructs a decision function using its training set, assuming that training instances are untainted (i.e., $\mathcal{A}(x) = x$). Then the adversary chooses his strategy $\mathcal{A}(\cdot)$ with the aim of maximising his expected utility (given by eq. 2), assuming perfect knowledge of the decision function chosen by the classifier and of its utility and cost functions. Finally, classifier moves again by choosing a new decision function, taking into account the move by the adversary and assuming to know his utility and cost functions. We point out that the optimal adversary strategy consists in defining, for each positive instance x, a modification x' which maximises the corresponding summand of the adversary expected utility in eq. 2, that is:

$$\mathcal{A}(x) = \underset{x'\in\mathcal{X}}{\operatorname{argmax}}[U_A(y_{\mathcal{C}}(\mathcal{C}(x'),+) - W(x,x')]. \quad (3)$$

Given the above definition of the adversary utility and cost functions, it is easy to see that the adversary will change a given instance x, only if it is correctly classified by the classifier as positive, and if there is any instance $\mathcal{A}(x) \neq x$ which is misclassified by the classifier as negative, and the modification cost $W(x,\mathcal{A}(x))$ is lower than the utility gain $U_A(-,+) - U_A(+,+)$. Otherwise, it turns out that the best strategy is to leave the instance x unchanged, namely $\mathcal{A}(x) = x$.

In [1] the above framework was applied to find the optimal adversary and classifier strategies, when the classifier is a Naive Bayes, under the standard assumption of game theory that both players have perfect knowledge of each other, namely each one knows the utility and cost functions of the other one, and the adversary also knows the classification algorithm and the training set used by the classifier. The corresponding adversarial classification system was then experimentally evaluated on a spam filtering task, quantitatively showing that the classifier effectiveness significantly degrades if the adversarial nature of this task is not taken into account, while an adversary-aware classifier can perform significantly better.

3.2 Multiple Classifier Systems for Adversarial Classification

The framework for adversarial classification proposed in [1] applies also to the case we are interested in, namely a system made up of an ensemble of N classifiers working on different feature sets, whose decision function consists in thresholding the sum of the scores provided by each classifier. To this aim, denoting with $s_i(x_i)$ the score provided by the i-th classifier for the instance x represented in the i-th feature space, and with t the decision threshold, it is sufficient to define the decision function as

$$y_{\mathcal{C}}(x) = \begin{cases} +, & \text{if } \sum_{i=1}^{N} s_i \geq t, \\ -, & \text{if } \sum_{i=1}^{N} s_i < t, \end{cases} \quad (4)$$

and to consider V_i and $W_i(x,\mathcal{A}(x))$ as the cost incurred respectively by the classifier for measuring the i-th feature set of the instance x and by the adversary for modifying it.

We now define the classifier strategy against the adversary as the addition of one or more classifiers to the previous ensemble, based on different feature sets. The rationale of this strategy is intuitive, as pointed out in section 2: taking into account different characteristics, or views, of the same instances, it should be easier to discriminate positive from negative ones, and at the same time it should be more difficult for the adversary to evade all of them. We would like to formally analyse this rationale, in light of the framework in [1]. Analogously to [1], we assume that initially the classifier is trained on a given training set and operates on untainted instances, namely $\mathcal{A}(x) = x$. Then the adversary reacts by devising a strategy $\mathcal{A}(x)$, which is likely to decrease the classifier effectiveness. Next, the classifier adds some new classifiers to the previous ensemble. The adversary can in turn react again by devising a new strategy to defeat the new version of the MCS, and so on.

Let us now define the adversary strategy, namely the optimal way in which the adversary should choose the function $\mathcal{A}(x)$ against a given ensemble of N classifiers. To this aim, we assume that the adversary knows the feature set used by each of the individual classifiers, the score $s_i(x), i = 1,\ldots,N$, provided by each individual classifier for any positive instance x, and the threshold t. We also assume that the cost $W_i(x,\mathcal{A}(x))$ for modifying the i-th feature set is equal to the absolute difference between the corresponding scores $s_i(x)$ and $s_i(\mathcal{A}(x))$. This means that the total cost $W(x,\mathcal{A}(x))$ equals the Manhattan distance in the N-dimensional score space between the corresponding score vectors. This is a reasonable assumption without any specific knowledge about the nature of features. It is reasonable to assume that the higher the score reduction the adversary would like to attain, the more the modifications he has to make. Thus for getting a lower score he has to face a higher cost. We point out that a similar assumption about the cost of modifying an instance in a given feature space was made in [5]: in that work, the cost was assumed to be a function of the distance between x and $\mathcal{A}(x)$ in the feature space. The optimal strategy of the adversary against an ensemble of N classifiers, defined in eq. 3 for the general case, can be rewritten as follows:

$$\mathcal{A}(x) = \begin{cases} x' \neq x, & \text{if } \exists x' : y_{\mathcal{C}}(x') = -, \Delta U_A > W(x, x'), \\ & x' = \text{argmax}_{x'' \in \mathcal{X}} \Delta U_A - W(x, x''), \\ x, & \text{otherwise}, \end{cases} \tag{5}$$

where ΔU_A is given by $U_A(-,+) - U_A(+,+)$. Under the above assumptions, the above optimal strategy can be rephrased as finding, for any given instance x which is correctly classified as positive by the classifier, namely $\sum_{i=1}^{N} s_i(x) \geq t$, an instance $\mathcal{A}(x)$ which is misclassified as negative by the classifier, namely $\sum_{i=1}^{N} s_i(\mathcal{A}(x)) < t$, and for which the utility gain ΔU_A exceeds the cost for making the modification, which by the above assumptions is given by:

$$W(x, \mathcal{A}(x)) = \sum_{i=1}^{N} |s_i(x) - s_i(\mathcal{A}(x))|. \tag{6}$$

If no such instance can be found, then x is left unchanged. It is not difficult to see that the minimum cost the adversary has to incur so that the modified instance is misclassified as negative equals the difference between the total score given to x by the classifier and the decision threshold t: $\sum_{i=1}^{N} s_i(x) - t$.

It is now possible to give a formal explanation, according to the above framework, of the reasons why the classifier ensemble approach, and in particular adding new classifiers (using new features) to a given ensemble, can be effective in the considered kind of applications. We consider the simplest case in which the previous N classifiers and the decision threshold s are left unchanged. A consequence of adding M new classifiers ($M \geq 1$) is that the score of any positive pattern could increase. In particular, considering the optimal strategy $\mathcal{A}(x)$) against N classifiers, if $\mathcal{A}(x) \neq x$, as seen above the modified instance $\mathcal{A}(x)$) evades the classifier, namely $s^N(x) = \sum_{i=1}^{N} s_i(\mathcal{A}(x)) < t$. However, this is no more guaranteed if one ore more new classifiers are added. Indeed the new score $s^{N+M}(x)$ will be given by the sum of the previous one and of that of the new classifiers, $s^N(x) + \sum_{i=N+1}^{N+M}(x)$, which could exceed t. In other words, the optimal strategy of the adversary against N classifiers could be less effective, in the sense that it could allow to evade the classifier for a smaller set of instances, if new classifiers are added to the ensemble. This means that the detection capability of the classifier has improved. Moreover, the cost to evade the classifier could increase, just because the score of any positive pattern could increase. For some positive patterns x correctly classified by the new classifier ensemble, such increase of the score from $s^N(x)$ to $s^{N+M}(x)$ could make the difference $s^{N+M}(x) - t$ larger than the utility, $U_A(-,+)$, that the adversary would gain by modifying x so that it is misclassified as negative. This implies that there could be some positive instances that the adversary can afford to modify to evade N classifiers, but not to evade $N + M$ classifiers. This means that the classifier has become harder to evade.

The above analysis gives a first theoretical support to the arguments proposed in favour of the classifier ensemble approach in many works related to security applications, mentioned in section 2. In the next section we will apply the above framework to a case study related to the spam filtering task, using a real spam filter and a real corpus of spam and legitimate e-mails, to quantitatively evaluate the improvement in the effectiveness and in the hardness of evasion attainable using the classifier ensemble approach.

4 A case study in spam filtering

In this section we apply the theoretical framework of section 3.2 to a case study involving a real spam filter, SpamAssassin. This is a well known and widely used spam filter, and is particularly suitable to our aim since it is an open source software. The architecture of SpamAssassin (as well as the one of most commercial spam filters), summarised in section 2, fits the one considered in section 3.2. This allows us to analyse the effect of adding new filtering modules (in other words, new classifiers based on different feature sets) on the detection capability and on the hardness of evasion of this spam filter. For our experiments we used the latest version of SpamAssassin (3.2.4) and the configuration named "bayes+net", which includes all the available filtering modules (in particular, a Naive Bayes text classifier) comprising the optional ones (Razor, Pyzor, etc.). Each module has its own scoring system, and scores can be continuous valued or discrete. SpamAssassin is deployed with a predefined scoring system for each module, and a predefined detection threshold t set to 5. We chose to use these predefined settings, although the scoring system of each rule and the value of t can be modified by users. The only module which includes a trainable classification algorithm is the one based on the Naive Bayes text classifier: we trained this classifier as described below.

Our experiments were carried out on the publicly available TREC 2007 e-mail data set.[3] It is made up of 75,419 real e-mail messages, received by a mail server between April 2007 and July 2007, and contains 25,220 legitimate and 50,199 spam messages. For training the Naive Bayes classifier used by SpamAssassin, we used the first 10,000 messages of the data set in chronological order. This training set was made up of 1,969 legitimate e-mails and 8,031 spam e-mails. The remaining 65,419 e-mails were used as test set.

For the purpose of our experiments, the main problem in using a real spam filter made up by a large number of different modules is that it is very difficult to define for each of them the possible modifications that the adversary (namely, a spammer) can make to evade the filter. Moreover, this also makes difficult to define commensurable modification costs for the adversary, for modules using heterogeneous features.[4] Nevertheless, taking into account that in the theoretical framework of section 3.2 the modification cost for modifying the feature vector representation of instances corresponding to any given module was defined as the absolute distance between the scores provided by that module before and after the modification, we decided to avoid the explicit definition of the possible modifications. We simply assume that the adversary can make any modification which reduces arbitrarily the score provided by each module. So the only constraint on the adversary is that the cost faced for modifying any e-mail has to be lower than the utility gained by getting that e-mail misclassified by the filter, as explained in section 3.2. We point out that this simplifying assumption is totally in favour of the adversary, not of the classifier, since we are not setting any constraint on the actual modifications which can be made on spam e-mails.

In our experiments we considered the utility functions $U_A(y_{\mathcal{C}}, y)$

[3] `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/`

[4] This turned out to be possible in the experiments reported in [1] on the same application, because the experiments were made using a single Naive Bayes text classifier: in that work, the modifications were defined as adding dictionary words or using synonyms.

and $U_C(y_C, y)$ reported below:

$$U_A = \begin{cases} U_A(+,+) = 0 \\ U_A(-,+) = 1,5 \\ U_A(+,-) = 0 \\ U_A(-,-) = 0 \end{cases} \quad U_C = \begin{cases} U_C(+,+) = 1 \\ U_C(-,+) = -10 \\ U_C(+,-) = -1 \\ U_C(-,-) = 1 \end{cases} \qquad (7)$$

Note that two different utility functions for the adversary were considered, differing only in the value of $U_A(-,+)$, which is the utility gained by the adversary when he evades the filter. This value affects the capability of adversary of evading the filter, since it equals the maximum cost the it can afford for modifying e-mails (we remind the reader that an e-mail x can be changed to any e-mail x', only if $W(x, x') \leq U_A(-,+) - U_A(+,+)$). Finally, we assumed that the cost V_i faced by the MCS for measuring the feature vector of the i-th classifier is zero (such cost is just a negative constant added to the expected value of the utility function in the framework of section 3.2).

The version of SpamAssassin we used is made up of 619 filtering rules each based on different characteristics (features) of e-mails (the list of the rules can be found at `http://spamassassin.apache.org/tests_3_2_x.html`). Since this number is rather high, we did not add rules one at a time. Instead we subdivided them into n disjoint subsets $S_1, \ldots, S_n$, and added at each step all the rules of a given subset. For the purposes of these experiments we chose $n = 6$. The number of rules for each subset was set to 119 for S_1 and to 100 for all the other subsets. In the real SpamAssassin filter new rules are usually added in response to new spammers' tricks which are identified in real spam e-mails. Accordingly, it would have been reasonable to subdivide the rules taking into account their chronological order. Unfortunately the time in which each rule was introduced is not reported in the SpamAssassin documentation. So we resort to make a random subdivision. To make experiments easily reproducible, we sorted the rules alphabetically according to their names as listed in `http://spamassassin.apache.org/tests_3_2_x.html`. The only exception was the rule related to the Naive Bayes classifier: since it is known that text classifiers are used in spam filters since several years, we included this rule in the first subset, S_1.

To evaluate how the addition of filtering modules affects the detection rate and the hardness of evasion of SpamAssassin using the above framework, the experiments were carried according to the above procedure.

```
1. $R \leftarrow \emptyset, \mathcal{A}^0(x) = x$
2. For $k = 1, \ldots, n$:
  2.1 $R \leftarrow R \bigcup S_k$
  2.2 Evaluate the expected utility of the classifier and of the adversary, when the classifier uses the rules in $R$ and the adversary uses the strategy $\mathcal{A}^{k-1}(x)$ which was optimal for the previous set of rules
  2.3 Compute the optimal adversarial strategy $\mathcal{A}^k(x)$ against rules in $R$
  2.4 Evaluate the expected utility of the classifier and of the adversary, when the classifier uses the rules in $R$ and the adversary uses the corresponding optimal strategy $\mathcal{A}^k(x)$
```

The above experimental set up can be phrased as follows. At each step, we first evaluate the performance of the classifier and the adversary after a new set of rules is added to the classifier, and the adversary uses the strategy which was optimal against the previous set of rules (in the first step, this means that the adversary does not modify his instances). This simulates what happens in real cases, as soon as a spam filer is updated. Then the optimal strategy of the adversary against the new set of rules is computed, and the performances of the classifier and the adversary are evaluated again. This simulates what happens when spammers devise new tricks to evade the last version of a spam filter.

The optimal adversarial strategy $\mathcal{A}^k(x)$ at each step was computed as follows, according to section 3.2 and to the above assumption about how the adversary can modify his instances. Denoting $S_1 \bigcup \ldots \bigcup S_k$ as R, for any positive instance x correctly classified by the filter (namely $y_C(x) = +$, or equivalently $\sum_{i \in R} s_i(x) \geq t$), we compute the set of feasible values s'_i for the scores of rules in R which would correspond to an instance x' classified as negative (namely $\sum_{i \in R} s'_i < t$), such that the corresponding cost $W(x, x') = \sum_{i \in R} |s'_i - s_i(x)|$ is minimum and is lower than the utility gain. If such scores can be found, then we assume that the adversary evades the filters by modifying x, otherwise it is assumed that the adversary can not afford to modify x to evade the filter.

The results are shown in figures 1 and 2, for both utility functions considered for the adversary, in terms of the expected utility of the adversary and of the classifier as a function of the number of rules used in SpamAssassin. The results in the top left graph refer to the case in which the adversary does not modify his instances. As one could expect, the expected utility of the classifier increases as the number of rules increases while the opposite happens for the adversary. In other words, this means that adding new filtering modules based on different features allows to improve the detection capability. The only exception is when passing from 419 to 519 rules. This can be due to the fact that after adding rules to SpamAssassin it would be better to reweight all of them, although this was not made in our experiments for the sake of simplicity. The bottom left graph shows what happens when the adversary uses the optimal strategy against each set of rules. The expected utility of the adversary significantly improves with respect to the previous case. The one of the classifier still increases as the number of rules increases, but obviously attains lower values than in the previous case. However, it is worth noting that the improvement attained by the adversary, reported in the top right graph from top, tends to decrease as the number of rules increases. Similarly, the decrease in classifier's expected utility tends to be higher for lower number of rules. The reason is that the modification cost the adversary has to face to evade the classifier increases as the number of rules increases, until it exceeds the utility gain for some positive instances, making it no more convenient to modify them. This is a clear evidence that adding new filtering modules based on different features allows to improve not only the classifier's discriminant capability, but also its hardness of evasion. Consider finally the bottom right graph, corresponding to the case when the classifier adds new rules, and the adversary uses the strategy which was optimal against the *previous* set of rules. For lower number of rules (up to 319), the expected utility of the adversary is between the ones of the first two graphs: this is reasonable, because it is now trying to evade only *some* of the rules used by the classifier. However, for higher number of features its expected utility is even worse than the one it attained *without* trying to evade any rule. The expected utility of the classifier is instead very close to the one it attained when the adversary did not try to evade any rule. This means that the addition of new rules allowed to compensate the tricks introduced by the adversary to evade the previous rules. In other words, most spam e-mails which evaded the previous version of the filter were detected by the new rules.

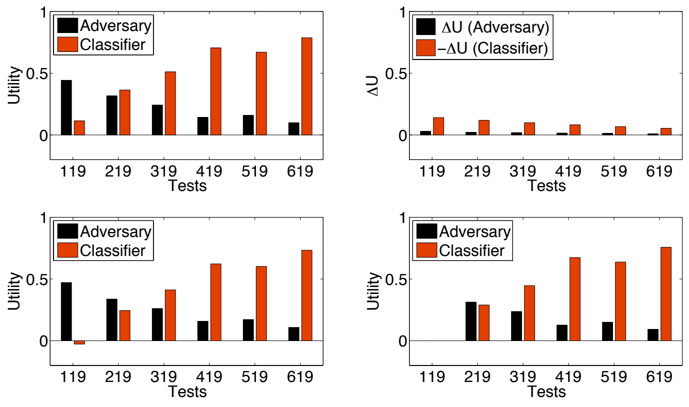


Figure 1. Expected utility for the adversary and the classifier, as a function of the number of rules used by the classifier, when $U_A(-,+) = 1$. Top left: the adversary does not modify his instances. Bottom left: the adversary uses the optimal strategy against the classifier. Top right: gain and loss in expected utility attained respectively by the adversary and the classifier, when passing from the situation in the top left graph to that in the bottom left one, for each number of rules used by the classifier. Bottom right: for each set of rules, the adversary uses the optimal strategy against the *previous* set of rules.

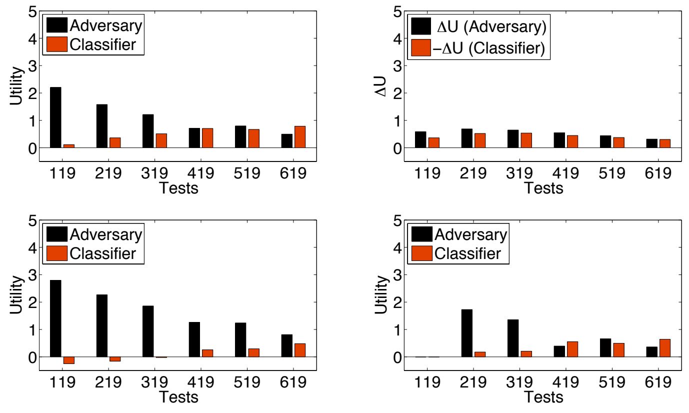


Figure 2. Expected utility for the adversary and the classifier, as a function of the number of rules used by the classifier, when $U_A(-,+) = 5$. See caption of figure 1 for the other details.

The behaviour of the expected utility in figures 1 and 2 is similar, with the obvious difference that the expected utility of the adversary is higher in the graphs of figure 2 than in figure 1, since it can afford a higher cost to modify instances (the opposite happens for the classifier). These experimental results on a real case study give thus a quantitative confirmation to the theoretical explanation given in section 3.2 on the effectiveness of the classifier ensemble approach in improving both the detection capability and the hardness of evasion of a security system like a spam filter.

ACKNOWLEDGEMENTS

We would like to thank Nilesh Dalvi and Mausam for providing us the code used in [1].

REFERENCES

[1] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma, 'Adversarial classification', in *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 99–108, Seattle, (2004).

[2] Giorgio Giacinto, Fabio Roli, and Luca Didaci, 'Fusion of multiple classifiers for intrusion detection in computer networks', *Pattern Recognition Letters*, **24**, 1795–1803, (2003).

[3] Michal Haindl, Josef Kittler, and Fabio Roli, eds. *Multiple Classifier Systems, 7th International Workshop, MCS 2007, Prague, Czech Republic, May 23-25, 2007, Proceedings*, volume 4472 of *Lecture Notes in Computer Science*. Springer, 2007.

[4] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas, 'On combining classifiers', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–239, (1998).

[5] Daniel Lowd and Christopher Meek, 'Adversarial learning', in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ed., ACM Press, Chicago, IL., (2005).

[6] Roberto Perdisci, Guofei Gu, and Wenke Lee, 'Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems', in *International Conference on Data Mining (ICDM)*, pp. 488–498. IEEE Computer Society, (2006).

[7] Arun A. Ross, Karthik Nandakumar, and Anil K. Jain, *Handbook of Multibiometrics*, Springer Publishers, 2006.

A Behaviour-Knowledge Space Approach for Spam Detection

Francesco Gargiulo and **Antonio Penta** and **Antonio Picariello** and **Carlo Sansone** [1]

Abstract. Unsolicited Commercial E-mails (UCE), commonly known as *spam*, are becoming a serious problem for e-mail accounts of single users, small companies and large institutions. In this paper we describe a novel method for detecting spam messages analyzing both text and image attached components. In particular, we describe an architecture in order to overcome some problems that are still boarded on the state-of-the-art spam-filters. The proposed system takes into account both the semantic richness of natural language and the recent spam evolution based on images, by using a Behaviour Knowledge Space approach for fusing results coming from different analysis of the e-mails.

1 Introduction

The presence of spam can seriously compromise normal user activities, forcing to navigate through mailboxes to find the - relatively few - interesting e-mails, so wasting time and bandwidth and occupying huge storage space. The types of those messages vary: some of them contains advertisements, other e-mails provides winning notifications, and sometimes we get messages with executable files, which finally emerge as malicious codes, such as viruses and Trojan horses. In addition, spam e-mails may often have unsuitable content (as a pornographic material advertising) that is illegal and sometimes dangerous for non adult users.

The recognition of spam content is not a trivial problem: there are some factors that are related with human perception, economic behavior, legal context, that are hardly measurable or summarized in adequate features. The same definition of *spam e-mails* requires a common agreement that is not easy to find.

In our opinion, *all* kind of spam e-mails have several common characteristics, such as: i) they are unsolicited, ii) they have a commercial content, even though the content itself is continuously evolving, trying to outsmart the classical countermeasures adopted by anti-spam filters. Consequently, a great variety of technical methodology have been implemented in current anti-spam systems [4].

We focused our attention on that measures related to e-mail contents, in particular *texts and images*, rather then on networking and identity strategies [14].

The textual filtering methods are widely deployed and they varies in the inspected content and the proposed methodology. Some filters consider only the header or the body of an e-mail, while other ones take both. These approaches use different models, considering word-tokens, their frequencies and their combinations. In *rule based-filters* [5] the users define some rules related to the headers or the bodies, considering particular words as *sign* of spam content; anyway, this simple solution is strongly dependent on how the words used by spammers can change .

Differently, *Signature-based* methods do not really deal with whole messages or specific tokens, transforming the message into a *signature*. Clearly, the methods effectiveness is related to the robustness of the signature function. Note that a signature database must be distributed and kept up to date very frequently, due to the rapid variation of spam e-mails.

A different kind of filters is based on *collaborative solutions*, in particular on Peer-to-Peer (P2P) networks for signature distribution [16, 6].

Statistical filters based on the the Bayes theory have also been investigated [1, 13].

Other approaches consider spam detection as a *binary classification problem* and several algorithms from the learning theory research field have been used. In these solutions, e-mails are mapped into multidimensional space, each dimension representing the words in the e-mail content; several measures are proposed such as the terms-frequency (tf) or the product between the documents-frequency (df) and terms-frequency, as in [8].

Image spam has been extensively studied using several techniques primarily developed from the Image Processing and Computer Vision community, using features related to text areas[2, 15] or color distribution [2]. A classifier is usually trained on such features, trying to discriminate spam images from legitimate ones. The main idea of these approaches is that images which contain texts are likely to be spam. In [7], the authors present features that are focused on simple property of image, making classification very fast. A different approach (Fumera et al. [9]) proposes to process each spam image using an OCR system, thus extracting embedded text instead of image features.

In this paper we combine the visual clues with the semantic information related to the e-mail body, to determine whether a message is spam. In order to address the problem of combining a non-constant number of modules, since it is not possible to *a priori* known if there is one or more images attached to the e-mail and/or there are textual information to be processed, we propose the use of a *Behaviour Knowledge Space* [11] approach.

The paper is organized as follows: section 2 describes at a glance the main component of the proposed system; in 3, and 4, we describe text and image features respectively, while in 5 we show how to combine them. In 6 several experiments are discussed, and in 7 we describe the conclusion of our work.

[1] Dipartimento di Informatica e Sistemistica, University of Naples Federico II, Italy, e-mail: {francesco.grg,a.penta,picus,carlosan}@unina.it

2 System Architecture

As shown in figure 1, we design a system that integrates image-based and text-based analysis. The mails, initially, are parsed by a Multipurpose Internet Mail Extensions (MIME) parser, that can retrieve the different parts of the e-mails. The text is thus processed by a *Text Analyzer* module, while the images are forwarded to an *Image Analyzer* module. The *Fusion block* has the role to use the results produced by the previous modules, furnishing the final classification of each e-mail.

Both the Text and the Image analyzer can be implemented by means of different classifiers, each one using different features. In the following, we will describe into details the different feature sets used and the combination process.

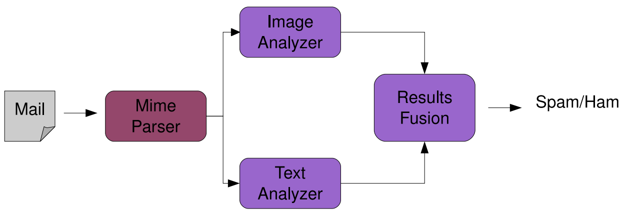


Figure 1. System Architecture

3 Textual Features

We propose a strategy based on text processing and analysis in order to process what we call *semantic* and *syntactical* features. Generally speaking, our main idea is to characterize how e-mails belonging to the same class (ham or spam) have the same meaning, using a set of semantic features.

In addition, we process e-mails in order to detect special characters that are typically used into spam context.

3.1 Semantic Features

We propose to use a feature set based on a modified version of Vector Space Model (VSM)[12]. The representation of e-mail textual content in the vector space model has a number of advantages, including the uniform treatment of queries and documents as vectors, the ability to differently weight the different terms; anyway, it suffers from its inability to cope with two classic problems arising in natural languages, i. e. synonymy and polysemy. We briefly recall that *synonymy* refers to a case where two different words (say ”pupil“ and ”scholar“) have the same meaning, and *polysemy* refers to the case where a term such as ”play” has multiple meanings according to different contexts.

Because the vector space representation fails to capture this kind of relationships, we choose a modified version of VSM, the Latent Semantic Analysis or LSA [12]. Despite LSA is a traditional and well accepted technique used to stick out the semantic contents in text-process community, there are few application in the spam framework.

LSA is an application of Singular Value Decomposition (SVD) to document-by-term $N \times M$ matrices A. In particular, SVD provides a suitable matrix decomposition as described in the following:

$$A = TSD^T$$

being S=diag$(\sigma_1, \dots, \sigma_r)$ a $M \times N$ matrix, with $\sigma_i = \sqrt{\lambda_i}$ and $\lambda_i \geq \lambda_{i+1}$ with $1 \leq i \leq r$; the $\lambda_1, ..., \lambda_r$ be the eigenvalues of AA^T, r being the rank of A. Note that A^TA has the same eigenvalues of AA^T.

The values σ_i are also denoted as the *singular values* of A. In the LSA technique it is used a reduced version of A , known as $M \times N$ matrix A_k, k being a positive integer that is the maximum rank of A_k. This approximation is computed taking into account the distance between the two matrices X = A-A_k that is minimal according to a Frobenius norm [12].

In other words, we have a reduced space in which the words that have similar co-occurrence patterns are projected (or collapsed) into the same dimension, and in the indexing phase the technique projects the documents into the new generated space with latent semantic dimensions. In order to derive the features to learn a classifier during the training phase, we adopt the LSA similarity measure that is the projection of the document in the space obtained by $S_k \times D_k^T$, S_k and D_k being the matrices after the SVD reduction. In the testing phase we use also this matrix product T_k^TQ in order to compute the text feature thanks to the SVD equation:

$$T_k^TQ = S_kD_k^T \tag{1}$$

$(*)_k$ being the matrices obtained after the reduction process and Q being the $N \times 1$ matrix representing the input document.

We also propose an intelligent filter that is able to detect and reject those words not *human-readable*, i.e. “fsdrx”, “jkdld”: this solution is based on an SVM classifier trained on several features derived from bigrams and trigrams composition of English words. Note that the use of this kind of filter has also the aim of enhancing the recognition of the semantic content that can be used in particular spammer attacks, such as the ones which use to put random words into e-mail texts, thus trying to reduce the effectiveness of current antispam algorithms.

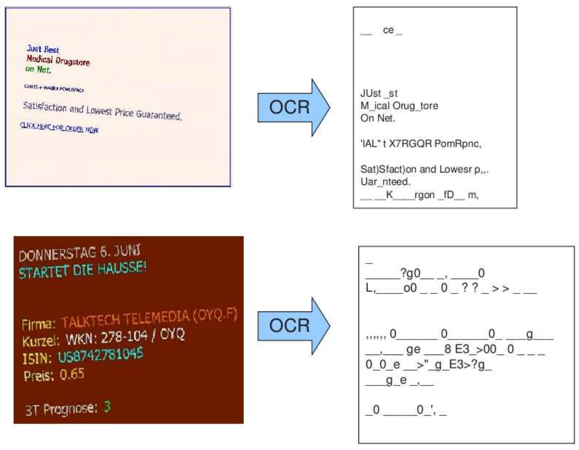
3.2 Syntactical Features

We propose to use some syntactic features that can be extracted from mail texts, in order to estimate usual and suspected mail formats.

Spammers, in fact, usually try to obfuscate the textual part of an e-mail’s body by substituting some characters in order to bypass the effectiveness of antispam filters.

So, we defined another set of features for obtaining a characterization of this kind of obfuscated text. The features we are investigating on, are mainly based on the presence of *special* characters, i.e. those characters that should not be frequently present in a legitimate text. The whole set we considered is made up of the following characters: { **!, ”, #, $, %, &, ’, (,), *, +, ,, -, . . . , /, @**}. Starting from this set we defined six *syntactical features*:

- **text_length:** the number of characters of the whole text
- **words_number:** the number of words in the text
- **ambiguity:** the ratio between the number of special and normal characters
- **correctness:** the ratio between the number of words that do not contain special characters and the number of words that contain special characters
- **special_length:** the maximum length of a continuous sequence of special characters
- **special_distance:** the maximum distance between two special characters belonging to the above considered set.



```
__   ce _



JUst _st
M_ical Orug_tore
On Net.

'IAL" t X7RGQR PomRpnc,

Sat)Sfact)on and Lowesr p,,.
Uar_nteed.
__ __K____rgon _fD__ m,
```

```
_
______?g0__ _, _____0
L,_____o0 _ _ 0 _ ? ? _ > > _ __


,,,,,, 0________0________0_ ____g____
__,____ ge ____8 E3_>00_ 0 _ _ _
0_0_e __>"_g_E3>?g_
____g_e _,__

_0 ______0_', _
```

Figure 2. Outputs obtained by applying *gocr* (available at http://jocr.sourceforge.net) to some spam images

4 Image Features

We propose a novel approach for the detection of the image spam in which two different image processing techniques are used. The first one is devoted to directly extract some global features from each image attached to the e-mails. Such features should also be able to detect if images were adulterated or not, by considering the complexity of the image itself as it is perceived from an human being. The second processing is carried out by means of two steps: first, there is a preprocessing phase with the use of an OCR, then a feature extraction process starting from the OCR output try to characterize it in order to detect if the embedded text has been voluntarily obfuscated and/or distorted.

4.1 Visual Features

The first set of features, that we called *visual features*, are directly obtained from the image attached to the mails. In order to give an image characterization that should be able to discriminate between normal and adulterated images, we considered features that describe the image texture from a statistic point of view. As said before, in fact, spammers typically now try to bypass filters that use an OCR for detecting texts within an image by obfuscating such texts with the addition of some noise or by superimposing a texture (see also Figure 2). So, texture detection can help in individuating images that contain spam messages. For the sake of simplicity, in the following we will present the considered features in case of gray-level images, but the same operators can be applied to color images too.

We will use $\{I(x,y), 0 \le x \le N-1, 0 \le y \le M-1\}$ to denote a $N \times M$ image with G gray levels. All the considered statistical texture measures are based on the co-occurrence matrices. Spatial gray level co-occurrence estimates image properties related to second-order statistics. The $G \times G$ gray level co-occurrence matrix $P_{\mathbf{d}}$ for a displacement vector $d = (dx, dy)$ is defined as follows. The entry (i,j) of $P_{\mathbf{d}}$ is the number of occurrences of the pair of gray levels i and j which are a distance **d** apart. Formally, it is given as:

$$P_{\mathbf{d}}(i,j) = |\{((r,s),(t,v)) : I(r,s) = i, I(t,v) = j\}|$$

where $(r,s),(t,v) \in N \times M$, $(t,v) = (r+dx, s+dy)$, and $|.|$ is the cardinality of a set.

As regards the choice of the displacement vector **d**, we considered the four direct neighbors of each pixel, i.e. we used four pairs as values of dx and dy for calculating the number of co-occurrences, namely $(0,1)$, $(1,0)$, $(-1,0)$ and $(0,-1)$. We do not perform a normalization of $P_{\mathbf{d}}$ in order to preserve the dependence of the considered features on the image size.

As suggested in [10], from the co-occurrence matrix it is possible to extract features that can be used for detecting a texture within an image. In particular, we considered the following five features:

- **Contrast**
$$\sum_i \sum_j (i-j)^2 P_{\mathbf{d}}(i,j)$$
is the difference in terms of visual properties that makes an object (or its representation within an image) distinguishable from other objects and the background. In the visual perception of real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view. In practice, it is the ratio between the brightest and the darkest value of the image. In the case of a B/W image, note that the increase of the contrast is equal to erase gray values.

- **Entropy:**
$$-\sum_i \sum_j P_{\mathbf{d}}(i,j) log P_{\mathbf{d}}(i,j)$$
is an index of the brightness variation among the pixel in an image. More the values of brightness are different each others, more the entropy will be higher.

- **Energy:**
$$\sum_i \sum_j P^2_{\mathbf{d}}(i,j)$$
is the spectral content of an image

- **Correlation:**
$$\frac{\sum_i \sum_j (i-\mu_x)(j-\mu_y) P_{\mathbf{d}}(i,j)}{\sigma_x \sigma_y}$$
is an index of the correlation degree among the pixel. Here μ_x and μ_y are the means and σ_x and σ_y are the standard deviations of $P_{\mathbf{d}}(x)$ and $P_{\mathbf{d}}(y)$ respectively, where $P_{\mathbf{d}}(x) = \sum_j P_{\mathbf{d}}(x,j)$ and $P_{\mathbf{d}}(y) = \sum_i P_{\mathbf{d}}(i,y)$

- **Homogeneity:**
$$\sum_i \sum_j \frac{P_{\mathbf{d}}(i,j)}{1+|i-j|}$$
is a measure of the brightness variation within the image. If the image is completely black or white, its homogeneity value will be the maximum. On the contrary, if the image contains several brightness variations, this value will be very low.

Another category of features that can be used for characterizing images from a global point of view is based on the complexity of an image for a human reader. We have chosen to consider a feature also proposed in [3]:

- **Perimetric Complexity:** is defined as the squared length of the boundary between black and white pixels (the perimeter) in the whole image, divided by the black area.

Note that, differently from [3], we evaluate the perimetric complexity on the whole image, after performing a binarization with a fixed threshold.

4.2 OCR-based Features

Here we propose to use the same features considered in Section 3.2. In this case, however, special characters are extracted from the output of an OCR that has received an attached image as input.

We have noticed, in fact, that characters embedded into an image are opportunely distorted and/or obfuscated in spam e-mails. Thus, most of the words cannot be correctly detected, as we can see in Figure 2. Furthermore, several special characters that typically are not present in commonly used words can appear in the OCR output.

5 Combining Text-based and Image-based Classifiers

It has been experimentally shown that the combination of an ensemble of classifiers can be of great benefit in many practical pattern recognition applications. Through the appropriate choice of a combination rule, it is possible to dampen the overall effect of the *independent* errors in each observation domain, thus reaching performance better than those of a single classifier.

The combination of classifiers is then an important part of our architecture. Anyway, there are some problems that must be taken into account in this case:

- It is necessary to define a method for combining a non-constant number of classifiers, since it is not possible to *a priori* known if there is one or more images attached to the e-mail and/or there are textual information to be processed.
- It should be avoided *padding-attacks* from spammers. That is, the possibility that an attacker puts a spam message within a *normal* context, for example by attaching an image containing an embedded spam message to an e-mail that contains *normal* images.

As shown in Figure 3 we used a two-stage approach. The first stage (denoted as *Classification* in Fig. 3) consists in a simple *logical OR*. Through this approach we try to address the problem of *padding attacks*.

Then, at the second stage we adopt a *Behavior Knowledge Space* (BKS) combining rule [11]. The idea behind this rule is to avoid making unjustified assumption on the classifier ensemble such as classifier independence.

A BKS is a K-dimensional space where each dimension corresponds to the decision of a classifier. Given an e-mail to be assigned to one of 2 possible classes, the ensemble of K classifiers can in theory provide 2^K different decisions.

Anyway, we also consider the case in which a classifier cannot be activated. It happens, for example, when there are no images within the e-mail, or when there are no words to be processed by the semantic analysis. In this situations, we assume that the output of the classifier is *undefined*. So, each classifier can attribute a mail to one out of three possible classes, i.e. {*ham*, *spam*, *und*} and the number of different decisions becomes 3^K.

Each one of these decisions constitutes one unit of the BKS. In the learning phase each BKS unit can record 2 different values e_i, by considering that the actual classes are only *ham* and *spam*. Given a suitably chosen training set, each sample x of this set is classified by all the classifiers and the unit that corresponds to the particular classifiers' decision is activated. It records the actual class of x, say C_j,

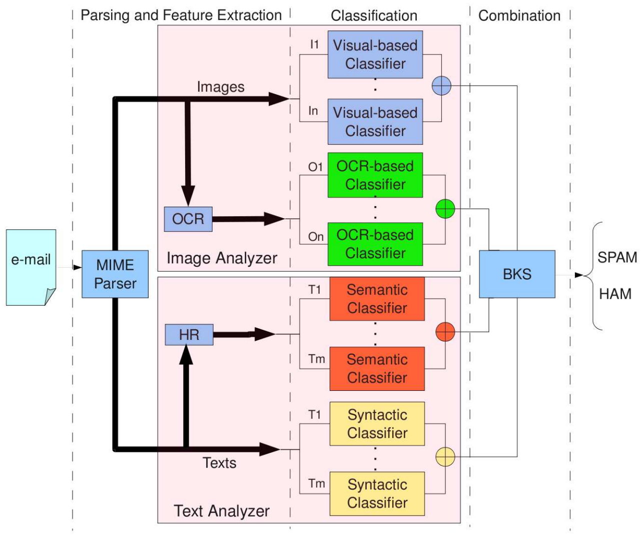


Figure 3. A BKS approach for combining text-based and image-based classifiers

by adding one to the value of e_j. At the end of this phase, each unit can calculate the best representative class associated to it, defined as the class that exhibits the highest value of e_i. This class corresponds to the most likely class, given a classifiers' decision that activates that unit.

In the operating mode, for each e-mail to be classified, the K decisions of the classifiers are collected and the corresponding unit is selected. Then the e-mail is assigned to the best representative class associated to that unit. Since we consider all the possible combinations of classifiers outputs as the number of available classifiers varies, we are implicitly handling the fact that the number of available classifiers can be different for each e-mail.

It is worth noting that the proposed combining scheme could be also easily extended using different feature sets, and then other classifiers, are defined. In this case, the problem is that the number of BKS unit grows exponentially and a wider training set is needed in order to achieve good results. However, as it will be shown in the following Section, only a subset of all the possible units are activated in practice, since some configuration of the classifiers' decisions are not allowed.

6 Experimental Results

In the following we will first present the database used for assessing the effectiveness of the proposed approach, then evaluate if the use of both visual and textual features can improve the performance of the system with respect to the use of a single set of features. Finally, we make a comparison of our approach with a state-of-the-art anti-spam filter, i.e. *SpamAssassin* equipped with two different spam image plug-ins.

As regards the dataset, whose details are given in Table 1, it is composed by 11652 e-mails, 9173 of which contains spam messages. e-mails were collected from the mailboxes of some users of the `studenti.unina.it` mailserver in a period of about three years (2005-2007). This mailserver hosts the mailboxes of all the students of the University of Naples Federico II. Among those e-mails, 151 contain *ham* images and 1802 contain *spam* images.

Total # of e-mails		*e-mails with Images*	
Spam	*Ham*	*Spam*	*Ham*
9173	2479	1802	151

Table 1. The dataset used in our tests.

As regards the first stage of our architecture (the *Classification* one), we chose a *Decision Tree* for implementing each classifier. In particular, a C4.5 (J48) coming from the open source tool *Weka*[2] was selected.

Each single classifier was trained on a set of 1,000 mails (500 for each class) different from those belonging to the dataset reported in table 1. In order to train the BKS rule, the dataset was splitted into two sets. Then, two experiments have been made, by using a set for training and the other one for testing. Results are finally obtained as the average value of the accuracy reported in these two tests.

Classifiers	**Accuracy**
Syntactic	85.89%
Semantic	86.65%
OCR-based	84.84%
Visual-based	92.06%
Proposed system (BKS fusion)	91.92%

Table 2. The accuracy of the four considered single classifiers and of the proposed system. Note that the last two single classifiers - third and fourth rows - processed only e-mails with attached images

In Table 2 the performance of the single classifiers and of the proposed systems are reported. It can be noted that the use of the BKS significantly improves the performance of the single classifiers. It must be remarked, in fact, that the visual-based classifier operates on a subset of the whole dataset (only 1953 mails out of 11652). It is also interesting to note that the number of BKS units that are really activated on the whole dataset is only 18, while their total number is 3^4, i.e. 81. This confirms the considerations made in the previous Section.

Finally, in Table 3 we report a comparison of the results obtained by our system with those obtainable with *SpamAssassin* in its standard configuration and equipped with two plug-ins devised for filtering image spam, namely *Bayes-OCR*[3] and *Fuzzy-OCR*. It clearly aperars that our approach significantly outperforms both *Bayes-OCR* and *Fuzzy-OCR*, by reaching a significantly higher accuracy. Finally, note the time needed for processing the whole dataset by our system are practically the same needed by SpamAssassin with *Fuzzy-OCR*, while is significantly faster than *SpamAssassin* equipped with *Bayes-OCR*.

7 Conclusion

In this paper we presented an approach for addressing the spam e-mail problems, which takes into account some of the recent evolutions of the spammers tricks and the limits of previous methodologies. Tests on a dataset of e-mails containing attached images con-

	Accuracy
Our System	91.92%
Bayes-OCR	83.56%
Fuzzy-OCR	83.57%

Table 3. Comparison between the proposed system and *SpamAssassin* with *Bayes-OCR* and *Fuzzy-OCR*.

firmed the effectiveness of the approach and its applicability with respect to other widely used opensource tool such as *SpamAssassin*.

In the future, we plan to better characterize the contribution in terms of CPU time of the various component of our architecture, in order to find the best tradeoff between obtainable accuracy and computational complexity.

REFERENCES

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouas, and C. Vassilakis, *An Evaluation of Naive Bayesian Anti-Spam Filtering*, Proc. of the 11th European Conf. on Machine Learning, pp. 9–17, 2000.

[2] H.B. Aradhye, G.K. Myers, J.A. Herson *Image Analysis for Efficient Categorization of Image-based Spam E-mail*, Proc. of the Eighth Intern. Conf. on Document Analysis and Recognition, Washington, DC, USA, 2005.

[3] B. Biggio, G. Fumera, I. Pillai, F. Roli *Image Spam Filtering Using Visual Information*, Proc. of the 14th Intern.l Conf. on Image Analysis and Processing, pp. 914–918, Modena, Italy, pp. 105–110, 2007.

[4] E. Blanzieri, A. Bryl, *A Survey of Anti-Spam Technique*, Technical Report DIT-06-056, Informatica e Telecomunicazioni, University of Trento, 2006.

[5] W. Cohen, *Learning rules that classify e-mail*, Papers from the AAAI Spring Syposium on Machine Learning in Information Access, pp. 18–25, 1996.

[6] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati, *P2p-based collaborative spam detection and filtering*, Proc. of the Fourth Intern. Conf. on Peer-to-Peer Computing, pp. 176–183, 2004.

[7] M. Dredze, R. Gevaryahu, A. Elias-Bachrach. *Learning Fast Classifiers for Image Spam*. In proceedings of the Conference on e-mail and Anti-Spam (CEAS), 2007.

[8] H. Drucker, D. Wu and V.N. Vapnik: 1999, *Support Vector Machines for Spam Categorization*, IEEE Transactions on Neural Networks 10(5), pp. 1048–1054.

[9] G. Fumera, I. Pillai, F. Roli, *Spam filtering based on the analysis of text information embedded into images*, Journal of Machine Learning Research, 7:2699-2720, 2006.

[10] Haralick, R.M.: Statistical and Structural Approaches to Texture. Proceedings of the IEEE, 67 (5) (1979) 786–804

[11] Y.S. Huang and C.Y. Suen. A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals. IEEE Trans. Pattern Analysis and Machine Intelligence, 17(1), pp. 90–94, 1995.

[12] C.D Manning, and H. Schtze, *Foundations of Statistical Natural Language Processing*, June 1999, The MIT Press.

[13] V. Metsis, I. Androutsopoulos, G. Paliouras, *Spam Filtering with Naive Bayes - Which Naive Bayes?* In Proceedings of the second Conference on e-mail and Anti-Spam (CEAS), Mountain View, CA, USA, 2006.

[14] G. Schryen, *Anti-Spam Measures, Analysis and Design*, Springer-Verlag, 2007.

[15] C.-T. Wu, K.-T. Cheng, Q. Zhu, Y.-L. Wu, *Using visual features for anti-spam filtering*, Proc. IEEE Conference on Image Processing, vol. 3, pp 509-512, 2005

[16] F. Zhou, L. Zhuang, B. Zhao, L. Huang, A. Joseph, and J. Kubiatowicz, *Approximate object location and spam filtering on peer-to-peer systems*, Proc. of ACM/IFIP/USENIX International Middleware Conference, 2003.

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

[3] This plug-in is available for download at the URL: http://prag.diee.unica.it/n3ws1t0/?q=node/107

Towards an Intelligent Decision Support System for Intensive Care Units

Pedro Gago[1] and **Manuel Filipe Santos** [2]

Abstract. Intensive Care Units (ICUs) are an attractive field for data analysis as they provide huge amounts of data related to the patient's condition. However, effective decision support systems operating in such an environment should not only be accurate but also as autonomous as possible, being capable of maintaining good performance levels without human intervention. Moreover, the complexity of an ICU setting is such that available data only manages to cover a limited part of the feature space. Such characteristics led us to investigate the development of ensemble update techniques capable of improving the discriminative power of the ensemble. Our chosen technique is inspired on the Dynamic Weighted Majority algorithm, an algorithm initially developed for the concept drift problem. In this paper we will show that, in the problem we are addressing, simple weight updates do not improve results, whereas an ensemble where we allow not only weight updates but also the creation and eliminations of models increases significantly the performance.

1 Introduction

Since the 1960's computer applications whose purpose was that of supporting the decision making process have been designed [17]. Even though the first computer applications in business environments were intended to make easier operational activities like order processing, billing or inventory control, the need arose for tools that could ease the tasks related to decision support [1].

In the medical area several expert systems were built and deployed [2, 9, 16]. However, the failure rate was high as the effort required to update the knowledge base was excessive and the scope of the expert systems was very limited.

Researchers started shifting their attention to the automation of the knowledge acquisition process by using methods from several areas of expertise (e.g. machine learning, statistics). Knowledge Discovery from Databases (KDD) [4] is well suited for this task. In fact, given that there is enough data, KDD techniques make knowledge acquisition easier thus simplifying the task of building decision support tools. Despite KDD being a semi-automatic process, the predictive models still need to be re-evaluated on a regular basis to detect any loss of predictive accuracy. In fact, model performance is known to degrade over time as the world does not remain in a stationary state [7] (e.g. in the medical area new drugs and therapeutic procedures are constantly being developed). Whenever performance drops bellow acceptable values it is necessary to repeat the KDD process or, at the very least, retrain the models using the latest data. Thus, an adaptive Decision Support System (DSS) must include mechanisms to detect the degradation in performance and to act accordingly in order to maintain the needed performance levels [15]. Moreover, it is now well established that prediction accuracy can usually be improved by using ensembles of prediction models instead of a single model [3, 10]. Despite ensemble performance being usually better than that of a single model the quality of ensemble predictions also degrades with the passing of time [7].

In this paper we present several experiments aiming at predicting the final outcome for patients staying in an Intensive Care Unit (ICU). Our final goal is that of building a DSS connected to the hospital's computer network allowing for real-time prediction and continuous performance assessment. The prediction is the result of an ensemble of classifiers, composed of both neural networks and decision trees as it has been shown that the different model types contribute to a lower number of coincident failures thus increasing the ensemble performance [19]. Whenever necessary the system automatically alters the ensemble, either by changing the models weights, by deleting poor performing models or by adding new models.

In section 2 we present an overview the overall problem we are trying to solve and also describe the data used in the experiments presented in this paper. Section 3 contains a brief overview of related work. In section 4 we describe the experimental setting and in section 5 we present the experimental results. Finally, section 6 includes the discussion of the results and in section 7 we present conclusions and pointers to future work.

2 Problem description

We are currently developing an intelligent Decision Support System called INTCare [5]. Operating in an ICU setting, INTCare uses data available in the first 24 hours, after ICU admission, to predict the patient's outcome (the patient status at the time of hospital discharge: dead or alive) and also to predict organ failure for six organ or systems (cardiovascular, respiratory, hepatic, renal, central nervous system and hematologic). Initially, the models included in INTCare were obtained via batch off-line training even though the INTCare's architecture allows for integration with the Hospital's Electronic Patient Records. Such integration will allow INTCare to be semi-autonomous as it will be possible to automatically collect data both for making predictions and to the appraisal of its predictive performance. This autonomous behavior demands the inclusion of adjustment mechanisms into INTCare making it capable of maintaining an acceptable performance as time passes.

In this paper we present the results of several experiments on the use of this information to build a predictive model that maintains an interesting predictive performance in the ICU. In particular we are going to address the problem of creating a system capable of predicting hospital mortality for ICU patients using data collected during

[1] Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, email: pgago@estg.ipleiria.pt
[2] Universidade do Minho, Portugal, email: mfs@dsi.uminho.pt

the first 24 hours after ICU admission. Moreover, such system must be able to function without human intervention, i.e. it must automatically adjust to new data. We are interested in comparing the performance of a static ensemble system with that of several dynamic ensemble systems. It is our belief that the dynamic systems better suit the problem at hand as the data available is never enough to cover the diversity of patients and the related clinical phenomena that occurs in an ICU and thus the training set does not contain enough examples to completely describe the concept being investigated.

2.1 Data Description

The available data is composed of data collected during the first day of ICU stay for approximately 13000 patients. Outcome information was added to each record, indicating the status of the patient at the time of hospital discharge (dead or alive).

Four variables contain the information that remains unchanged during the patient's stay, including the site where the patient came from, the type of admission, the patient's age and the Simplified Acute Physiology Score (SAPS II) score [12] (SAPS II is a severity of disease classification system). The remaining variables (except for the outcome) contain values collected during the inpatient first day in the ICU. The remaining attributes were derived from the information available on the intermediate outcomes, which are defined from four monitored biometrics: the systolic blood pressure, the heart rate, the oxygen saturation and the urine output (UR). The information regarding these monitored biometrics was condensed into 12 variables (3 for each of the biometrics) indicating the existence and duration of what was defined as relevant clinical events. Information regarding the definition of events and critical events may be found in [14]. Finally, the last attribute denotes the patients' final outcome (status at the time of hospital discharge).

3 Related Work

The weight update procedure we use to supervise the ensemble is inspired both on the on the Weighted Majority algorithm [13] and the Dynamic Weighted Majority (DWM) [8]. Even though DWM is intended to track concept drift we found the general idea appealing and decided to investigate the use of a similar algorithm in a more stable task of predicting the outcome of patients in a ICU.

Even if inspired by DWM, our implementation is different in several details. One of the most relevant is that we do not use incremental learning algorithms. After being included in our ensemble the models (or experts) are not modified in any way, only their weights are changed as determined by the algorithm. Unlike DWM the creation and elimination of experts is not directly dependent on any individual prediction made. Rather it is the result of the overall prediction results over the entire batch of records being processed.

4 Experimental Setting

Two different ensemble evolution strategies were evaluated and compared to a "'traditional"' ensemble (Configuration A). In the first one (Configuration B) the model weights are changed after the evaluation of each batch of records. The weights update procedure is also included in the second configuration (Configuration C). Also, in this configuration new models are created using the records in each batch and the poor performing models (those with negative weights) are removed from the ensemble.

In order to investigate the effect of evaluating batches of records of different sizes we tested batches of 10, 20, 50 , 100 and 200 records. The rationale behind this is that bigger batches may reduce the responsiveness of the system and thus lower its tendency for over training.

The same initial ensemble was used for each configuration. This ensemble was created using a process similar to the Random Subspace Method [6]. In our approach all the available training records are used when creating a new model but each attribute has only a 50% chance of being selected. Our initial ensemble is composed of 50 models, 25 of which are neural networks with the other 25 being decision trees. The algorithms used for model creation were those present in the Weka tool [20]. We used j48 for decision trees and the multilayer perceptron for neural networks. Both algorithms were run with the default parameter values. In our base ensemble all the models are assigned the same weight ($\frac{1}{50}$).

In configurations B and C, in order to evaluate the effects of allowing changes in the model's weights we decided to adjust the weights after each prediction. The models that had made a correct prediction had their weights increased. Those who failed had their weights reduced. We started by doing this after each prediction, but then investigated whether evaluating batches of records before making the weights updates would lead to better results. We tested updating the weights after each batch of 10, 20, 50, 100 and 200 records. After each batch of predictions (of size N) we had the number of correct predictions (C). First we calculated the fraction of the increment for each weight:

$$P = \frac{C - \frac{N}{2}}{\frac{N}{2}} \tag{1}$$

where P is positive if the model is correct in more than half of the records in the batch being considered. It is zero for those cases where exactly half of the answers are correct and it is negative for the remaining cases. If all the predictions are correct, the value for P is 1, if all are wrong the value is -1. The weight update is then the result of equation 2 where w_i stands for the weight of model *i*.

$$w_i = w_i + P * \frac{1}{10 * number\ of\ models} \tag{2}$$

In configuration C new models were added to the ensemble after the evaluation of each batch of records. Two new models are created: one decision tree and one neural network. Both are trained on that batch of records using the same method described above for the initial training of the ensemble. The correspondent weight is equal to the average of the weights of the other models in the ensemble. The algorithm is as follows (as in DWM, we used the term "'expert"' instead of "'model"'):

Weight Updates Algorithm

```
{$\vec{x}, y$} : training data
{$e, w$}$^1_m$ : experts and their weights
$p$ : number of records in each batch
$\delta_i$ : fraction of weight increment for expert $_i$

$n\vec{u}m = 0$
for i = 1,...,n
    $answer \leftarrow 0$
    for j = 1, ..., m
        predicted $\leftarrow$ Classify(e $_j, x_i$ )
        **if** (predicted = $y_i$)
            num $_j$ $\leftarrow$ num $_j$ + 1
        **end if**
        answer $\leftarrow$ answer + predicted * $w_j$
    **end for**
    **output** answer
    **if** ($i$ mod $p$ = 0)
        $\vec{\delta} \leftarrow$ Increment(num, p)
        $\vec{w} \leftarrow$ UpdateWeights($\vec{w}, \vec{\delta}$)
        $\{e, w\} \leftarrow$ DeleteNeg($\{e, w\}$)
        $m \leftarrow m + 1$
        $e_m \leftarrow$ Create-DecTree(precords)
        $w_m \leftarrow \bar{w}$
        $m \leftarrow m + 1$
        $e_m \leftarrow$ Create-NNetwork(precords)
        $w_m \leftarrow \bar{w}$
        $\vec{w} \leftarrow$ Normalize($\vec{w}$)
    **end if**
**end for**
```

To evaluate the results we used the average of the values of the area under the Receiver Operating Characteristic curve (AUC ROC) obtained after 30 runs of each experiment. The ROC curves are often used in the medical area to evaluate computational models for decision support, diagnosis and prognosis [11, 21]. A model presenting an AUC of 1 has perfect discriminative power (perfect predictive ability) while a value of 0.5 corresponds to random guessing.

5 Results

We divided the available data into two mutually exclusive datasets. Models were created using the first dataset. Those models were then evaluated using the second dataset. Several experiments were conducted with different parameter settings with regard to the frequency of weights updates and the possibility of creating of new models.
We started by evaluating the performance of the static ensemble(no changes are made to the ensemble during the evaluation of the records). With no changes in the ensemble composition the AUC ROC was 78.80% ±0.93 %.

Next we tested configurations B and C investigating the effect of using different intervals for evaluation before the weights were changed. Different configurations include weight changes after every record was evaluated or after batches of 10, 20 , 50, 100 or 200 records. In Table 1 we present the AUC under the ROC curve for each of the configurations considered, allowing us to see the ensembles with better discrimination capabilities (better able to distinguish between the patients with outcome 0 or 1).

Table 1. Results for the ensemble evolution (% of AUC ROC).

Config.	B	C
1	78.79±0.93	–
10	78.72±0.93	85.05±0.07
20	78.61±0.93	84.83±0.10
50	78.44±0.92	84.58±0.15
100	78.03±0.92	84.41±0.21
200	77.45±0.91	82.94±0.33

B - Weight updates; C - Weight updates and model creation and elimination

These results seem to indicate that it is best to update the models weights immediately after the evaluation of each record. In order to further clarify this point we decided to evaluate the ROCs in a segmented manner. Considering batches of 200 records and computing the AUC ROC for each of the batches we got the results that are show in Figure 1. The graph shows an example of the evolution of the partial AUC ROC values for one of the experiment runs.

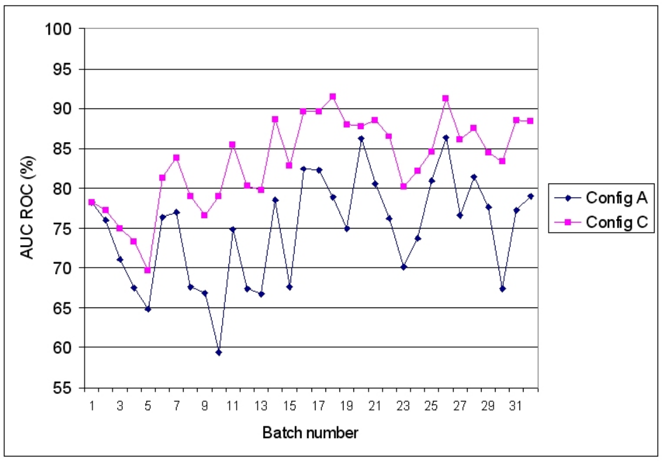


Figure 1. Evolution of AUC ROC values over time

We can clearly see that after the first weight update, the performance of the ensemble using configuration C is significantly better than that of the static ensemble (configuration A). However, even if the overall trend is positive there are some batches of records where the AUC ROC value drops. That may be explained by the variables we used and by the composition of the batches. Indeed, there are some medical conditions that cannot be detected by analysing the available variables (e.g. a patient with head trauma often has normal values for the four biometrics parameters included in this work). Batches with an higher number of such records are likely to lead to lower AUC ROC values. The increase in discriminative power may be better perceived in Figure 2. Here we show the increase in AUC ROC after each batch of 200 records is processed.

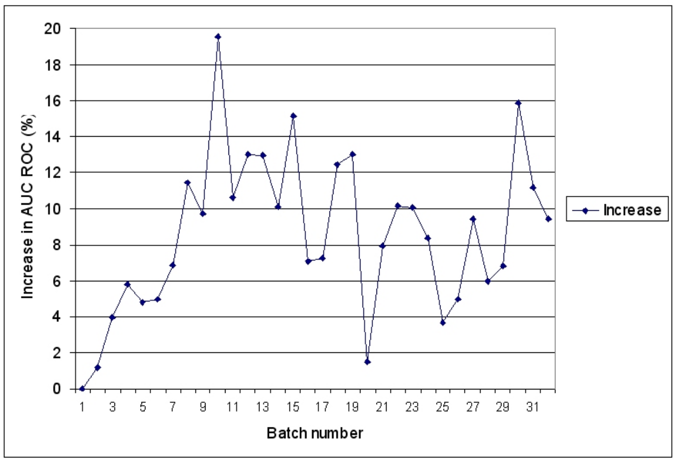


Figure 2. AUC ROC increase.

It is clear that there is a real gain in using the evolution algorithm (configuration C) as opposed to a mere static ensemble (configuration A). Indeed, after the first two batches the increase in AUC ROC is allways greater than 4% except for one batch.

Finally Figure 3 shows the differences in performance for configuration C when different batch sizes are considered. In order to allow a clear understanding we plotted only those values for batches of 50 and batches of 200 records.

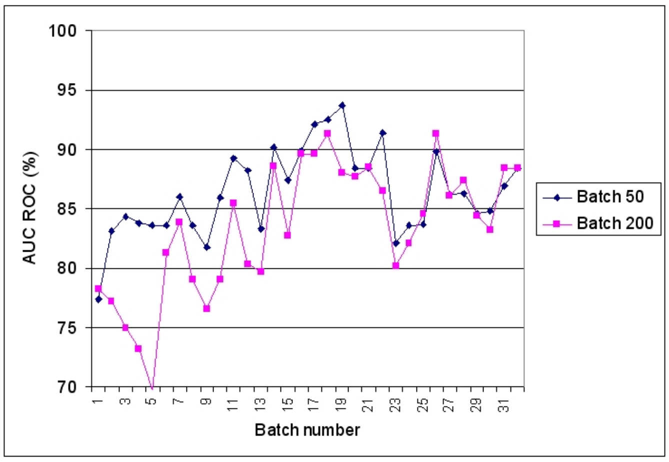


Figure 3. AUC ROC variation with weights updated after 50 or after 200 records.

6 Discussion

Ensemble methods are known to improve results when compared to those from single classifiers. Our tests show that allowing for dynamic adjustments ensemble (both in terms of models' weights and in terms of number of models) leads to overall better discriminative power of our ensemble classifier. Moreover, the intervals at which such changes are made seems to be an issue worthy of further study as different values were obtained with different intervals. Finally, the issue of whether or not to change the ensemble composition seems to point to a solution where the creation and elimination of models is encouraged.

The size of the batches used is still an open issue as smaller batch sizes increase the number of models in the ensemble and may prove impractical in a real setting. Indeed, figure 3 seems to suggest that in the short term smaller batches are recommended. However, if we examine more closely the right side of that figure (after allowing the ensemble to evolve over several batches of records) it seems that the performance of the second ensemble is converging to (if not surpassing) that of the first ensemble. This suggests investigating update procedures with batches of increasing size.

7 Conclusion

Future decision supports systems must be capable of adapting to changes in their environment [15, 18]. In the medical area this will allow for an easier integration of these tools in everyday use as its reliability tends to increase. In this paper we presented the results of a set of experiments in building the adaptive module of a decision support system (INTCare). Considering the available data, both in the present time and in the foreseeable future we tested dynamic hybrid ensemble architectures allowing for unassisted operation while maintaining acceptable performance. We concluded that, for our problem one should allow for dynamic ensembles with the addition of new models after each batch of records is examined and the elimination from the ensemble of those models that have negative weights.

Future work includes extending the architecture to include organ failure prediction. Other necessary developments regard the need to incorporate all the available data as it is being registered. For patients staying several days in the ICU the prediction models must take into account not only data from the initial 24 hours but also all the data stored since then. Moreover, relevant clinical information is possibly hidden in the sequence in witch clinical adverse events occur. As the INTCare system has access to data collected via bed side monitoring this seems to be an interesting path to be explored.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments which helped improve this paper.
Part of this work was supported by the Fundação para a Ciência e Tecnologia (grant SFRH BD 28840 2006). Financial support for this study was received from the FCT project PTDC/EIA/72819/2006 and from the Algoritmi research center.

REFERENCES

[1] David Arnott and Graham Pervan, 'A critical analysis of decision support systems research', *Journal of Information Technology*, **20**(2), 67–87, (June 2005).

[2] B. Buchanan and E. Shortliffe, *Rule-Based Expert Systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, 1984.

[3] Thomas G. Dietterich, 'Ensemble methods in machine learning', *Lecture Notes in Computer Science*, **1857**, 1–15, (2000).

[4] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth, *Advances in knowledge discovery and data mining*, chapter From data mining to knowledge discovery: an overview, 1–34, American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[5] Pedro Gago, Manuel Filipe Santos, Àlvaro Silva, Paulo Cortez, José Neves, and Lopes Gomes, 'Intcare : a knowledge discovery based intelligent decision support system for intensive care medicine', *Journal of Decision Systems*, **14**(3), 241–259, (2005).

[6] Tin Kam Ho, 'The random subspace method for constructing decision forests', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832–844, (1998).

[7] Ralf Klinkenberg and Stefan Ruping, 'Concept drift and the importance of examples', in *Text Mining – Theoretical Aspects and Applications*, Physica-Verlag, (2003).

[8] Jeremy Z. Kolter and Marcus A. Maloof, 'Dynamic weighted majority: A new ensemble method for tracking concept drift', in *Third IEEE International Conference on Data Mining*, (2003).

[9] C. A. Kulikowski and S. M. Weis, *Artificial Intelligence in Medicine*, chapter Representation of expert knowledge for consultation: the CASNET and EXPERT projects, 21–56, Westview Press, Boulder, 1982.

[10] Ludmila I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, 2004.

[11] Thomas A. Lasko, Jui G. Bhagwat, Kelly H. Zou, and Lucila Ohno-Machado, 'The use of receiver operating characteristic curves in biomedical informatics', *J. of Biomedical Informatics*, **38**(5), 404–415, (2005).

[12] J. R. Le Gall, S. Lemeshow, and F. Saulnier, 'A new simplified acute physiology score (saps ii) based on a european/north american multicenter study', *JAMA*, **270**(24), 2957–2963, (1993).

[13] N. Littlestone and M. Warmuth, 'The weighted majority algorithm', *Information and Computation*, **108**, 212–261, (1994).

[14] Álvaro Silva, Paulo Cortez, Manuel Filipe Santos, Lopes Gomes, and José Neves, 'Mortality assessment in intensive care units via adverse events using artificial neural networks', *Artificial Intelligence in Medicine*, **36**(3), 223–234, (2006).

[15] Zbigniew Michalewicz, Martin Schmidt, Matthew Michalewicz, and Constantin Chiriac, *Adaptive Business Intelligence*, Springer, 2006.

[16] H. E. Pople, *Artificial Intelligence in Medicine*, chapter Evolution of an Expert System: from INTERNIST to CADUCEUS, 179–208, Elsevier Science Publisher, Amsterdam, 1985.

[17] J. P. Shim, Merrill Warkentin, James F. Courtney, Daniel J. Power, Ramesh Sharda, and Christer Carlsson, 'Past, present, and future of decision support technology', *Decis. Support Syst.*, **33**(2), 111–126, (2002).

[18] Rustam Vahidov and Gregory E. Kersten, 'Decision station: situating decision support systems', *Decision Support Systems*, **38**, 283–303, (2004).

[19] Wenjia Wang, Derek Partridge, and John Etherington, 'Hybrid ensembles and coincident-failure diversity', in *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pp. 2376–2381, Washington, USA, (July 2001). IEEE Press.

[20] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools and techniques - 2nd Edition*, Morgan Kaufmann, Morgan Kaufmann, 2nd edn., 2005.

[21] MH Zweig and G Campbell, 'Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine', *Clin Chem*, **39**(4), 561–577, (1993).

Weighted Decoding ECOC for Facial Action Unit Classification

Terry Windeatt, Raymond S. Smith and Kaushala Dias [1]

Abstract. There are two approaches to automating the task of facial expression recognition, the first concentrating on what meaning is conveyed by facial expression and the second on categorising deformation and motion into visual classes. The latter approach has the advantage that the interpretation of facial expression is decoupled from individual actions as in FACS (Facial Action Coding System). In this paper, upper face action units (*aus*) are classified using an ensemble of MLP base classifiers with feature ranking based on PCA components. When posed as a multi-class problem using Error-Correcting-Output-Coding (ECOC), experimental results on Cohn-Kanade database demonstrate that error rates comparable to two-class problems (one-versus-rest) may be obtained. Weighted decoding is shown to outperform conventional ECOC decoding. The error rates obtained for six upper face *aus* around the eyes are believed to be among the best for this database.

1 INTRODUCTION

The problem of face expression recognition is difficult because facial expression depends on age, ethnicity, gender, occlusions as well as pose and lighting variation. Facial action unit (*au*) classification is an approach to face expression recognition that decouples the recognition of expression from individual actions. In FACS (facial action coding system) [1] the problem is decomposed into facial action units, that includes six upper face *aus* around the eyes. It has the potential of being applied to a much richer set of applications than an approach that targets facial expression directly. However, the coding process requires skilled practitioners and is time-consuming so that typically there are a limited number of training patterns.

There are various approaches to determining features for discriminating between *aus*. Originally, features were based on geometric measurements of the face that were involved in the *au* of interest [1]. More recently, holistic approaches based on PCA, Gabor [2] and Haar wavelets represent a more general approach to extracting features [3], and have been shown to give comparable results. The difficulty with these latter approaches is the large number of features. When combined with the limited number of patterns, this can lead to the small sample-size problem, that is when the number of patterns is less than or comparable to the number of features. A method of eliminating irrelevant features is therefore required [4] [5]. In this paper the Out-of-Bag error estimate is used to optimise the number of features.

In previous work [6] [7] five feature ranking schemes were compared using Gabor features in an MLP ensemble. The schemes were Recursive Feature Elimination (RFE) [9] combined with MLP weights and noisy bootstrap, boosting (single feature selected each round), one-dimensional class-separability measure and Sequential Floating Forward Search (SFFS). It was shown that ensemble performance is relatively insensitive to the feature-ranking method with simple one-dimensional performing at least as well as multi-dimensional schemes. It was also shown that the ensemble using PCA features with its own inherent ranking outperformed Gabor.

In this paper, PCA features are used with Error-Correcting Output Coding (ECOC) and a weighted decoding strategy based on bootstrapping individual base classifiers is proposed. The principle behind weighted decoding is to reward classifiers that perform well. The weights in this study are fixed in the sense that none change as a function of the particular pattern being classified. Sometimes this is referred to as implicit data-dependence or constant weighting. It is generally recognized that a weighed combination may in principle be superior, but it is not easy to estimate the weights.

The main contribution in this paper is to apply a weighted ECOC decoding strategy to the problem of facial action unit classification. Section 2 discusses ensemble techniques, Bootstrapping and ECOC for weighted decoding. Section 3 describes the database and design decisions for *au* classification, and compares 2-class classification with weighted and conventional ECOC decoding.

2 ENSEMBLES, BOOTSTRAPPING AND ECOC

We assume a simple parallel Multiple Classifier System (MCS) architecture with homogenous MLP base classifiers. A good strategy for improving generalisation performance in MCS is to inject randomness, the most popular strategy being Bootstrapping. An advantage of Bootstrapping is that the Out-of-Bootstrap (OOB) error estimate may be used to tune base classifier parameters, and

[1] University of Surrey, UK email: t.windeatt@surrey.ac.uk

furthermore, the OOB is a good estimator of when to stop eliminating features [8]. Normally, deciding when to stop eliminating irrelevant features is difficult and requires a validation set or cross-validation techniques.

Bootstrapping is an ensemble technique which implies that if μ training patterns are randomly sampled with replacement, $(1-1/\mu))^{\mu} \cong 37\%$ are removed with remaining patterns occurring one or more times. The base classifier OOB estimate uses the patterns left out of training, and should be distinguished from the ensemble OOB. For the ensemble OOB, all training patterns contribute to the estimate, but the only participating classifiers for each pattern are those that have not been used with that pattern for training (that is, approximately thirty-seven percent of classifiers). Note that OOB gives a biased estimate of the absolute value of generalisation error, but for tuning purposes the estimate of the absolute value is not important.

Error-Correcting Output Coding (ECOC) is a well-established method [10] [11] for solving multi-class problems by decomposition into complementary two-class problems. It is a two-stage process, coding followed by decoding. The coding step is defined by the binary *k x B* code word matrix Z that has one row (code word) for each of *k* classes, with each column defining one of B sub-problems that use a different labeling. Assuming each element of Z is a binary variable z, a training pattern with target class ω_l $(l = 1 \ldots k)$ is re-labeled as class Ω_1 if $Z_{ij} = z$ and as class Ω_2 if $Z_{ij} = \bar{z}$. The two super-classes Ω_1 and Ω_2 represent, for each column, a different decomposition of the original problem. For example, if a column of Z is given by $[0\ 1\ 0\ 0\ 1]^T$, this would naturally be interpreted as patterns from class 2 and 5 being assigned to Ω_1 with remaining patterns assigned to Ω_2. This is in contrast to the conventional One-versus-rest code, which can be defined by the diagonal *k x k* code matrix

Many types of coding are possible, but theoretical and experimental evidence indicates that, providing a problem-independent code is long enough and base classifier is powerful enough, performance is not much affected. In this paper, a random code with near equal split of labels in each column is used with *B=200 and k=12*. It has been shown theoretically and experimentally that a long random code performs almost as well as a pre-defined code, optimised for its error-correcting properties [11].

In the test phase, the *jth* classifier produces an estimated probability $\hat{q}_j$ that a test pattern comes from the super-class defined by the *jth* decomposition. The *pth* test pattern is assigned to the class that is represented by the closest code word, where distance of the *pth* pattern to the *ith* code word is defined as

$$D_{pi} = \sum_{j=1}^{B} \alpha_{jl} \left| Z_{ij} - \hat{q}_{pj} \right| \qquad l = 1, \ldots k \qquad (1)$$

where α_{jl} allows for *lth* class and *jth* classifier to be assigned a different weight. If $\alpha=1$ in equ. (1), Hamming decoding uses hard decision and L^1 *norm* decoding uses soft decision.

To obtain the OOB estimate, the *pth* pattern is classified using only those classifiers that are in the set OOB_m, defined as the set of classifiers for which the *pth* pattern is OOB. For the OOB estimate, the summation in equ. (1) is therefore modified to

$\sum_{j \in OOB_m}$ In other words columns of Z are removed if they correspond to classifiers that used the *pth* pattern for training.

In this paper we introduce a different weighted decoding scheme, that treats the outputs of the base classifiers as binary features. By using the diagonal matrix $\{Z_{ij} = 1$ if and only if $i = j\}$ the problem is recoded as *k* 2-class problems where each problem is defined by a different binary-to-binary mapping. There are many strategies that may be used to learn this mapping, but we use a weighted vote with weights set by class-separability measure applied to the training data, defined in [12].

Let y_{mj} indicate the binary output of the *jth* classifier applied to the *mth* training pattern, so that the output of base classifiers for the *mth* pattern is given by

$$y_{mj} = (y_{m1}, y_{m2} \cdots y_{mB}) \qquad (2)$$

Assuming in equ. (2) that a value of 1 indicates agreement of the output with target label and 0 disagreement, we can define counts for *jth* classifier as follows

$$N_j^{11} = y_{mj} \wedge y_{nj} \text{ and } N_j^{00} = \bar{y}_{mj} \wedge \bar{y}_{nj} \qquad (3)$$

where the *mth* and *nth* pattern are chosen from different classes.

The weight for the *jth* output is then defined as

$$w_j = \frac{1}{K} \left(\sum_{allpairs} N_j^{11} - \sum_{allpairs} N_j^{00} \right) \qquad (4)$$

where K is a normalization constant and the summation is over all pairs of patterns from different class.

The motivation behind equ. (4) is that the weight is computed as the difference between positive and negative correlation with respect to target class. In [12] this is shown to be a measure of class separability.

3 DATASET & EXPERIMENTAL EVIDENCE

The Cohn-Kanade database [13] contains posed expression sequences from a frontal camera from 97 university students. Each sequence goes from neutral to target display but only the last image is *au* coded. Facial expressions in general contain combinations of action units (*aus*), and in some cases *aus* are non-additive (one action unit is dependent on another). To automate the task of *au* classification, a number of design decisions need to be made, which relate to the following 1) subset of image sequences chosen from the database 2) whether or not the neutral image is included in training 3)image resolution 4)normalisation procedure 5)size of window extracted from the image, if at all 6) features chosen for discrimination. Furthermore classifier type/parameters, and training/testing protocol need to be chosen. Researchers choose different decisions in these areas, and in some cases are not explicit about which choice has been made. Therefore it is difficult to make a fair comparison with previous results.

We concentrate on the upper face around the eyes, involving *au1(inner brow raised), au2(outer brow raised), au4(brow lowered), au5(upper eyelid raised), au6(cheek raised),* and *au7(lower eyelid tightened)*. We chose an MLP ensemble and random training/test split of 90/10 repeated twenty times and averaged. Other design decisions we made were:

1) All image sequences of size 640 x 480 chosen
2) Last image in sequence (no neutral) chosen giving 424 images, 115 containing *au1*
3) Full image resolution, no compression
4) Manually located eye centres plus rotation/scaling into 2 common eye coordinates
5) Window extracted of size 150 x 75 pixels centred on eye coordinates
6) PCA applied to raw image with PCA ordering

With reference to 2), some studies use only the last image in the sequence but others use the neutral image to increase the numbers of *non-aus*. Furthermore, some researchers consider only images with single *au*, while others use combinations of *aus*. We consider the more difficult problem, in which neutral images are excluded and images contain combinations of *aus*. With reference to 4) there are different approaches to normalisation and extraction of the relevant facial region. To ensure that our results are independent of any eye detection software, we manually annotate the eye centres of all images, and subsequently rotate and scale the images to align the eye centres horizontally. A further problem is that some papers only report overall error rate. This may be mis-leading since class distributions are unequal, and it is possible to get an apparently low error rate by a simplistic classifier that classifies all images as *non-au*. For the reason we report area under ROC curve, similar to [5].

There are two sets of experiments aimed at 2-class and multi-class formulations of *au* classification. In both sets of experiments, the MLP ensemble uses two hundred single hidden-layer MLP base classifiers, with Levenberg-Marquardt training algorithm and default parameters. Random perturbation of the MLP base classifiers is caused by different starting weights on each run, combined with bootstrapped training patterns. In our framework, we vary the number of hidden nodes, with a single node for linear perceptron, and keep the number of training epochs fixed at 20.

The ultimate goal in *au* classification is to detect combination of *aus*. In the ECOC approach, a random *200x12* code matrix is used to consider each *au* combination as a different class. After removing classes with less than four patterns this gives a 12-class problem with *au* combinations as shown in Table 1. To compare the results with 2-class classification, we compute test error by interpreting super-classes as 2-class problems, defined as either containing or not containing respective *au*. For example, *sc2, sc3, sc6, sc11, sc12* in Table 1 are interpreted as *au1*, and remaining super-classes as *non-au1*

The first set of experiments detects *au1, au2, au4, au5, au6, au7* using six different 2-class classification problems, where the second class contains all patterns not containing respective *au*. The MLP ensemble uses majority vote combining rule. The best error rate of 9.4% for *au1* was obtained with 16 nodes and 28 features. The 9.4% error rate for *au1* is equivalent to 73% of *au1s* correctly recognised. However, by changing the threshold for calculating the ROC, it is clearly possible to increase the true positive rate at the expense of overall error rate. The best ensemble error rate, area under ROC with number of features and number of nodes for all upper face *aus* are shown in the first two columns of Table 2. Note that number of nodes for best area under ROC is generally higher than for best error rate, indicating that error rate is more likely to be susceptible to over-fitting.

The second set of experiments uses ECOC method described in Section 2, and figure 1 shows area under ROC for the six *aus*, as number of PCA features is reduced. Columns 3 and 4 in Table 2 show best L^1 *norm* decoding classification error and area under ROC, while last 2 columns show respective weighted decoding. It may be seen that weighted outperforms L^1 *norm* decoding. Also it may be seen from Table 2 that 2-class classification with optimized PCA features (columns 1 and 2) on average slightly outperforms ECOC. However, the advantage of ECOC is that all problems are solved simultaneously with 200 classifiers, and furthermore the combination of *aus* is recognized. As a 12-class problem, the mean best error rate over the twelve classes defined in Table 1 is 38.2 %, showing that recognition of combination of *aus* is a difficult problem.

4 DISCUSSION

The results for upper face *aus*, shown in Table 2, are believed to be among the best on this database (recognising the difficulty of making fair comparison as explained in Section 3).There are two possible reasons why the ECOC decoding strategy works well. Firstly, the data is projected into a high-dimensional space and therefore more likely to be linearly separable [14]. Secondly, although the full training set is used to estimate the weights, each base classifier is bootstrapped and therefore is trained on a subset of the data, which guards against over-fitting. As indicated in Section 2, bootstrapping also facilitates the OOB estimate for removing irrelevant features without validation. The effect of bootstrapping can be understood using bias/variance of 0/1 loss function [15]. In [6] it is shown that a bootstrapped ensemble benefits from reduced bias at the expense of increased variance.

Some preliminary results on other techniques to learn the binary-to-binary mappings defined in Section 2, indicate that the decoding strategy is fairly insensitive to the method of setting the weights. For example, similar results were obtained by using Adaboost logarithmic formula [16].

5 CONCLUSION

For upper face *au* classification, weighted decoding ECOC achieves comparable performance to optimized 2-class classifiers. However, ECOC has the advantage that all *aus* are detected simultaneously, and further work is aimed at determining whether problem-dependent rather than random codes can improve results.

References

[1] Y. Tian, T. Kanade and J. F. Cohn, Recognising action units for facial expression analysis, IEEE Trans. PAMI 23(2), 2001, 97-115.

[2] G Donato, M S Bartlett, J C Hager, P Ekman and T J Sejnowski, Classifying facial actions, IEEE Trans. PAMI 21(10), 1999, 974-989.

[3] Bartlett, M.S. Littlewort, G. Lainscsek, C. Fasel, I. Movellan, J. Machine learning methods for fully automatic recognition of facial expressions and facial actions, IEEE Conf. **Systems, Man and Cybernetics**, Oct 2004, Vol. 1, 592- 597.

[4] P. Silapachote, D. R. Karuppiah, and A. R. Hanson, Feature Selection using Adaboost for Face Expression Recognition, Proc. Conf. on Visualisation, Imaging and Image Processing, Marbella, Spain, Sept. 2004, 84-89.

[5] M S Bartlett, G Littlewort, M Frank, C Lainscsek, I Fasel and J Movellan, Fully automatic facial action recognition in spontaneous behavior, Proc 7th Conf. On Automatic Face and Gesture Recognition, 2006, ISBN 0-7695-2503-2, 223-238.

[6] T Windeatt, K Dias, Feature-ranking ensembles for facial action unit classification, IAPR Third Int. Workshop on artificial neural networks in pattern recognition, Paris, July, 2008, accepted.

[7] T. Windeatt., M. Prior, N. Effron, N. Intrator, Ensemble-based Feature Selection Criteria, Proc. Conference on Machine Learning Data Mining MLDM2007, Leipzig, July 2007, ISBN 978-3-940501-00-4, pp 168-182

[8] T Windeatt, M Prior, Stopping Criteria for Ensemble-based Feature Selection, Proc. 7th Int. Workshop Multiple Classifier Systems, Prague May 2007, Lecture notes in computer science, Springer-Verlag, 271-281

[9] Guyon I., Weston J., Barnhill S. and Vapnik V., Gene selection for cancer classification using support vector machines, Machine Learning 46(1-3), 2002, 389-422.

[10] T. G. Dietterich ,G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artificial Intelligence Research 2, 1995, 263-286

[11] T Windeatt and R Ghaderi, Coding and Decoding Strategies for multiclass learning problems, Information Fusion, 4(1), 2003, 11-21.

[12] T Windeatt, Accuracy/Diversity and Ensemble Classifier Design, IEEE Trans. Neural Networks 17(5), 2006, 287-297.

[13] T. Kanade, J. F. Cohn and Y. Tian, Comprehensive Database for facial expression analysis, Proc. 4th Int. Conf. automatic face and gesture recognition, Grenoble, France, 2000, 46-53.

[14] T.M. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, IEEE Trans. Information Theory, vol. EC-14, 1965, 326-334.

[15] G. Valentini, T. G. Dietterich, Bias-variance analysis of Support Vector Machines for the development of SVM-based ensemble methods, *Journal of Machine Learning Research*, 5 , 2004, MIT Press, 725-775.

[16] Y. Freund and R.E. Schapire. A decision-theoretic generalisation of on-line learning and an application to boosting, J. of Computer and System Science, 55(1), 1997, 119-139.

Table 1. ECOC super-classes of action units and number of patterns

ID	**sc1**	**sc2**	**sc3**	**sc4**	**sc5**	**sc6**	**sc7**	**sc8**	**sc9**	**sc10**	**sc11**	**sc12**
superclass	{}	1,2	1,2,5	4	6	1,4	1,4,7	4,7	4,6,7	6,7	1	1,2,4
#patterns	149	21	44	26	64	18	10	39	16	7	6	4

Table 2: Mean best test error rates (%) and area under ROC showing #nodes/#features for *au* classification with optimized PCA features and MLP ensemble

	2-class Test Error %	***2-class area under ROC***	***ECOC Test Error %***	***ECOC area under ROC***	***ECOC Weighted Error %***	***ECOC Weighted ROC***
au1	9.4/16/28	0.97/16/36	10.3/1/10	0.92/16/46	9.2/4/36	0.94/16/36
au2	3.5/4/36	0.99/16/22	3.4/1/36	0.96/16/28	2.8/16/22	0.98/1/46
au4	9.1/16/36	0.95/16/46	12.0/16/28	0.92/4/28	9.5/1/28	0.94/4/28
au5	5.5/1/46	0.97/1/46	3.6/16/36	0.99/1/36	3.2/1/36	0.99/1/36
au6	10.5/1/36	0.94/4/28	13.1/1/77	0.88/1/77	12.8/1/77	0.90/1/28
au7	10.3/1/28	0.92/16/60	11.6/1/28	0.89/4/46	10.9/4/46	0.92/1/36
mean	**8.1**	**0.96**	**9.0**	**0.93**	**8.1**	**0.95**

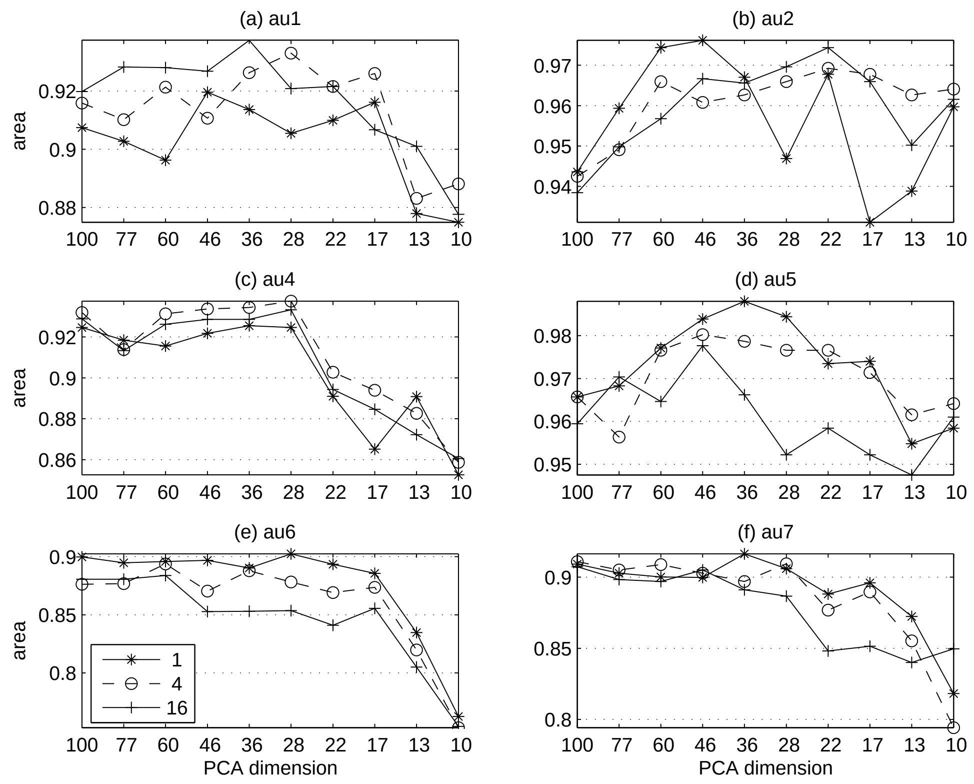


Figure 1: Area under ROC for weighted decoding ECOC MLP ensemble [1,4,16] hidden nodes 20 epochs versus number PCA features (logscale)

The Neighbors Voting Algorithm

Gabriele Lombardi and **Elena Casiraghi** and **Paola Campadelli**
{lombardi,casiraghi,campadelli}@dsi.unimi.it
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Abstract. In the last ten years the tensor voting framework (`TVF`), proposed by Medioni at al., has proved its effectiveness in perceptual grouping of arbitrary dimensional data. In the computer vision and image processing fields, this algorithm has been applied to solve various problems like stereo-matching, 3D reconstruction, and image inpainting.

In this paper we propose a new technique, inspired to the `TVF`, that allows to estimate the dimensionality and normal orientation of the manifolds underlying a given point set. These informations are encoded in tensors that can be considered as weak classifiers for the dimensionality classification problem; their linear combination is then used as a strong manifold dimensionality classifier. To prove the effectiveness of the described algorithm, two problems are faced: clustering by dimensionality estimation, and image inpainting by texture learning.

1 Introduction

The Tensor Voting Framework (`TVF`) proposed in [2] by Medioni at al., and further developed over the past ten years [6, 7], is a computational framework that can address a wide range of computer vision problems in a unified way. The framework has been designed based on perceptual principles, formulated by Gestalt psychologists, to infer salient structures from sparse and noisy data. It has been successfully applied to stereo matching [3, 5], image repairing [10], boundary inference [8], and motion segmentation [11].

The `TVF` is a general methodology suitable for problems of any dimensionality that can be formulated as a perceptual organization problem. In [4] Medioni at al. applied tensor voting to the problem of learning a target function from a set of points. To this aim they proposed a new implementation which can deal efficiently with data in arbitrary dimensional spaces. In [9] a new vote generation algorithm was proposed and a new family of decay functions was defined. In [1] Medioni and Tang described an augmentation of the `TVF` consisting in the estimation of the manifold's curvature and its usage to the aim of improving the manifold's inference precision.

In this paper we present a data analysis technique, the *Neighbors Voting* algorithm (`NV`), inspired to the `TVF`, that allows to estimate the dimensionality and normal orientation of the manifolds underlying a set of given points. The `NV` algorithm is iterative and consists in the generation of tensorial votes between neighboring points. Each pair "point-tensor" can be considered as a (weak) classifier for the dimensionality classification problem; the linear combination of weak classifiers is then used as a strong classifier to infer the dimension of the underlying manifold. In this work we demonstrate the efficiency and effectiveness of our approach with both synthetic and real data.

This paper is organized as follows: Section 2 briefly recalls the `TVF`; Section 3 presents the `NV` theory and describes the developed algorithm; Section 4 presents results and discusses future works.

2 The tensor voting framework

The `TVF` is a mechanism that forces the interaction among input tokens (points) in order to infer salient perceptual structures. Each token is associated to a local potential orientation of the manifold going through it. The orientation information is propagated from each token to its neighbors via a voting operation. Votes are casted from one location to each other, forcing orientation updates. Tokens that receive no vote can be classified as outliers. Tokens that receive votes from almost every direction, identify a locally unoriented manifold. Tokens that receive votes mainly from a well defined direction, describe an oriented manifold.

Every token is a location where an orientation is defined. To manage orientations, symmetric non-negatively defined second order tensors [1] are used. The eigensystem of this structures, in $\Re^n$, consists of n orthonormal eigenvectors, and n non-negative associated eigenvalues. The eigenvectors describe the orientation of the underlying manifold, whilst the eigenvalues give a confidence for each direction; geometrically, each tensor represents an hyper-ellipsoid. Each vote, casted by emitters, is itself a tensor.

Given the eigensystem $(\mathbf{X}, \mathbf{\Lambda})$, where $\mathbf{X}$ is an orthogonal matrix composed column wise by the orthonormal eigenvectors, and $\mathbf{\Lambda}$ is a diagonal matrix containing the corresponding eigenvalues, the associated tensor can be computed as: $\mathbf{T} = \mathbf{X\Lambda X}^T = \sum_{i=1}^{n} \lambda_i \mathbf{e}_i \mathbf{e}_i^T$, where the λ_i are the eigenvalues in non-increasing order, and the $\mathbf{e}_i$ are the corresponding eigenvectors. $\mathbf{T}$ can be decomposed in base tensors, as follows:

$$\mathbf{T} = \sum_{i=1}^{n-1} \left((\lambda_i - \lambda_{i+1}) \sum_{j=1}^{i} \mathbf{e}_j \mathbf{e}_j^T \right) + \lambda_n \sum_{j=1}^{n} \mathbf{e}_j \mathbf{e}_j^T \qquad (1)$$

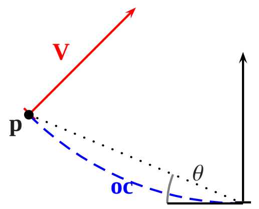
with i non null eigenvalues $(1 \leq i \leq n)$. The last term in Equation (1) is an unoriented (*ball*) tensor, the other $n-1$ terms are oriented tensors. The $(\lambda_i - \lambda_{i+1})$ and the λ_n coefficients are called *saliency* values and denoted by s_i. Each decomposed tensor term is useful to identify a manifold dimensionality: as an example, in $\Re^2$ the ball tensor identifies filled regions, and the oriented tensor (*stick*) describes curves. Tensor composition can be obtained summing tensors component by component, thus allowing to merge uncertainties, and to reinforce coherent orientations.

In the `TVF`, each token casts tensorial votes to neighboring tokens, and it is updated substituting its tensor with the sum of received

[1] In the following they will be simply referred as tensors.

votes. The voting fields, emitted by tokens, play a central role in the data information propagation process, so that the votes generating function must be chosen carefully. The simplest voting field is the stick field in $\Re^2$: given a stick tensor $\mathbf{E}$ (emitter), and a point $\mathbf{p}$ (receiver), the vote $\mathbf{V}$ computed in $\mathbf{p}$, shown in Figure 1, represents the most likely normal, in $\mathbf{p}$, to the curve that we want to infer, according to the curve's normal defined by the voter $\mathbf{E}$.

Figure 1. *The stick vote in $\Re^2$: in $\mathbf{p}$ the best orientation $\mathbf{V}$ is the normal to the osculating circle* $\mathbf{oc}$.



The vote $\mathbf{V}$ can be computed as follows:

$$\mathbf{V}_{stick}(\mathbf{p}) = DF(\mathbf{p}) \cdot \begin{bmatrix} -\sin(2\theta) \\ \cos(2\theta) \end{bmatrix} [\,-\sin(2\theta) \quad \cos(2\theta)\,] \tag{2}$$

where the dependency of $\mathbf{V}_{stick}$ from $\mathbf{E}$ is in the computation of the angle θ, and $DF(\cdot)$ is the *decay function* that controls the vote intensity with respect to the length s and curvature k of the arc $\mathbf{c}$. More precisely:

$$DF(\mathbf{p}; \sigma) = \exp\left(-\frac{s(\mathbf{p})^2 + ck(\mathbf{p})^2}{\sigma^2}\right) \tag{3}$$

The parameter σ controls the scale, whereas the constant c is used to maintain the isotropy. The arc length and the curvature are calculated as follows:

$$s(\mathbf{p}) = \frac{\theta\|\mathbf{p}\|}{\sin(\theta)}; \quad k(\mathbf{p}) = \frac{2\sin(\theta)}{\|\mathbf{p}\|} \tag{4}$$

Moreover, the decay function is forced to zero when $\theta > \frac{\pi}{4}$.

Given the stick field in $\Re^2$, the ball field can be obtained integrating the stick one over all its possible rotations, that is:

$$\mathbf{V}_{ball}(\mathbf{p}) = \int_0^{\pi} \mathcal{R}_\alpha \mathbf{V}_{stick}(\mathcal{R}_\alpha \mathbf{p}) \mathcal{R}_\alpha^T d\alpha \tag{5}$$

where $\mathcal{R}_\alpha$ represents a 2×2 rotation matrix.

The voting algorithm is conceptually a simple task: given a set of points $\{\mathbf{p}_i \in \Re^D\}$, a ball tensor (identity) is generated as initial emitter for each $\mathbf{p}_i$; then voting passes are iterated at least two times to improve the manifolds' orientation estimation encoded in the set of tensors $\mathbf{T}_i$ associated to the points. A voting pass consists of the following steps:

1. each receiver is initialized with a null tensor;
2. for each emitter $(\mathbf{p}_i, \mathbf{T}_i)$:
   (a) determine the set of the receivers $(\mathbf{p}_j, \mathbf{T}_j)$ distant less than 3σ from the voter [2];
   (b) for each receiver j compute the vote $\mathbf{V}_i^j$ casted to it by the emitter i, as follows:
      i. decompose $\mathbf{T}_i$ as defined in Equation (1) obtaining the base tensors $\mathbf{T}_{i,k}$, with $1 \le k \le D$;
      ii. for each base tensor $\mathbf{T}_{i,k}$ compute the tensorial vote $\mathbf{V}_{i,k}^j$ casted according to the associated voting field; as an example, in $\Re^2$ the computed votes are: the stick vote (see Equation (2)), and the ball one (see Equation (5));
      iii. compute the emitted vote as $\mathbf{V}_i^j = \sum_k s_k \mathbf{V}_{i,k}^j$ where s_k are the saliency values of $\mathbf{T}_i$;
   (c) update the receiver's tensor by adding to it $\mathbf{V}_i^j$.

After each voting pass execution, the output tensors accumulated in the receivers can be used as emitters during the next voting pass.

In [4] a direct method to compute tensorial votes, without the usage of integrals like the one reported in Equation (5), is presented; it allows to work in spaces of high dimensionality relaxing the memory requirement constraints and removing the tensorial fields' precomputation step. In [9] a new algorithm to compute tensorial votes is presented, that employs a new family of decay functions to reduce the additive noise effects on the inferred structure. The new decay functions presented there were:

$$NDF(\mathbf{p}; C, \sigma) = \exp\left(-\frac{\|C \cdot \mathbf{p}\|^2}{2\sigma^2}\right) \tag{6}$$

$$C(\mathbf{p}, \mathbf{dir}; per, ha, in) = 1 + in\left|\frac{\mathbf{p} \cdot \mathbf{dir}}{\|\mathbf{p}\|^{2ha}}\right|^{per} \tag{7}$$

The TVF is a general technique that could be applied in arbitrary dimensional spaces; nevertheless, when the space dimensionality is high this method cannot be used in practice, due to its high time and space complexity. Indeed, due to the curse of dimensionality, its time and space complexity grows as $\Theta(N\log(N)D^3)$ and $O(ND^3)$, where N is the number of data points and D is the space dimensionality[3]. The technique proposed in this paper is much faster in high dimensional spaces; it has a lower time complexity ($\Theta(N\log(N)D^2)$), and has less memory requirements ($O(ND^2)$). The drawback is that the NV technique is an approximation of the TVF, thus generating less precise results.

3 The Neighbors Voting algorithm

In this section we describe an algorithm that is a simplified version of the TVF; it obtains similar results with a lower time and space computational cost. To reduce the complexity we simplify the vote generation algorithm by avoiding the base tensors decomposition step, so that each emitter directly generates the vote on each receiver. Moreover, our voting scheme approximates tensors' orientation, thus preventing the onerous tensor rotation operations. To compensate for the missing information we add a nonlinear filtering step that allows to generate a local classification for the manifold dimensionality.

As in the TVF, the initialization of the NV algorithm associates ball tensors to each point $\mathbf{p}_i$ in the D dimensional space, so that "tokens" are defined with the pairs $(\mathbf{p}_i, \mathbf{T}_i)$. With this initialization no information is available about the manifolds' orientations, so that the first voting pass is different from the others; in [4] Medioni et. al. give the following equation to compute the ball tensorial vote casted from the emitter i to the receiver j:

$$\mathbf{T}_i^j = W_i^j \left(\mathbf{I} - \frac{(\mathbf{p}_j - \mathbf{p}_i)(\mathbf{p}_j - \mathbf{p}_i)^T}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}\right) \tag{8}$$

where $W_i^j = DF(\mathbf{p}_j - \mathbf{p}_i)$ is the ball decay function, that is the unnormalized gaussian function. Summing over all the emitters i, we obtain the vote at the receiver j:

$$\mathbf{T}^j = \sum_{\forall i} \mathbf{T}_i^j = W^j\mathbf{I} - \sum_{\forall i} W_i^j \frac{(\mathbf{p}_j - \mathbf{p}_i)(\mathbf{p}_j - \mathbf{p}_i)^T}{\|\mathbf{p}_j - \mathbf{p}_i\|^2} = W^j\mathbf{I} - \mathbf{A}^j \tag{9}$$

[2] The radius depends on the voting field edge size. The value 3σ allows to capture more than the 99% of the decay volume.

[3] The $N\log(N)$ part of the $\Theta(\cdot)$ mean-case complexity notation is due to the (realistic) hypothesis that there are approximately $\log(N)$ neighbors to each data point, and that the used data structure allows to find them in logarithmic time.

where $\|\cdot\|$ is the euclidean norm, $W^j = \sum_{\forall i} W_i^j$, and $\mathbf{A}^j = \sum_{\forall i} W_i^j \frac{(\mathbf{p}_j-\mathbf{p}_i)(\mathbf{p}_j-\mathbf{p}_i)^T}{\|\mathbf{p}_j-\mathbf{p}_i\|^2}$. Representing the spectral norm with $\|\cdot\|_\rho$, and defining $s_i^j = \frac{(\mathbf{p}_j-\mathbf{p}_i)}{\|\mathbf{p}_j-\mathbf{p}_i\|}$, we can demonstrate that $\left\|W^j\mathbf{I}\right\|_\rho = W^j \geq \left\|\mathbf{A}^j\right\|_\rho$:

$$\begin{aligned}\left\|\mathbf{A}^j\right\|_\rho &= \left\|\sum_{\forall i} W_i^j \frac{(\mathbf{p}_j-\mathbf{p}_i)(\mathbf{p}_j-\mathbf{p}_i)^T}{\|\mathbf{p}_j-\mathbf{p}_i\|^2}\right\|_\rho \\ &= \left\|\sum_{\forall i} W_i^j s_i^j (s_i^j)^T\right\|_\rho \leq \sum_{\forall i} W_i^j \underbrace{\left\|s_i^j (s_i^j)^T\right\|_\rho}_{1} = W^j \quad (10)\end{aligned}$$

Next, we show that

$$\frac{\mathbf{T}^j}{\|\mathbf{T}^j\|_\rho} = I - \frac{\mathbf{A}^j}{\|\mathbf{A}^j\|_\rho} \Leftrightarrow \left\|\mathbf{A}^j\right\|_\rho = W^j \quad (11)$$

At first, we note that if $\left\|\mathbf{A}^j\right\|_\rho = W^j$, from Equation (10) it must be:

$$\left\|\sum_{\forall i} W_i^j s_i^j (s_i^j)^T\right\|_\rho = \sum_{\forall i} W_i^j \left\|s_i^j (s_i^j)^T\right\|_\rho \quad (12)$$

Since $\forall i, j$ we have that $W_i^j > 0 \wedge \|s_i^j\| = 1$, from Equation (12) it follows that $\forall i, j, k; |s_i^j \cdot s_k^j| = 1$; therefore, $rank(\mathbf{A}^j) = 1$ and $min\left(\lambda\left(\frac{A^j}{\|A^j\|_\rho}\right)\right) = 0$, where $\lambda(\cdot)$ is the function that computes the eigenvalues of a given matrix. Thanks to this demonstration we can write:

$$\begin{aligned}\mathbf{T}^j &= \mathbf{W}^j I - \mathbf{A}^j = \mathbf{W}^j\left(I - \frac{\mathbf{A}^j}{\mathbf{W}^j}\right) \\ &\Rightarrow \frac{\mathbf{T}^j}{\mathbf{W}^j} = I - \frac{\mathbf{A}^j}{\mathbf{W}^j} \Rightarrow \frac{\mathbf{T}^j}{\|\mathbf{A}^j\|_\rho} = I - \frac{\mathbf{A}^j}{\|\mathbf{A}^j\|_\rho} \quad (13)\end{aligned}$$

and

$$\begin{aligned}\left\|\mathbf{T}^j\right\|_\rho &= \left\|\mathbf{A}^j\right\|_\rho \left\|I - \frac{\mathbf{A}^j}{\|\mathbf{A}^j\|_\rho}\right\|_\rho \\ &= \left\|\mathbf{A}^j\right\|_\rho \left(1 - \underbrace{min\left(\lambda\left(\frac{A^j}{\|A^j\|_\rho}\right)\right)}_{0}\right) = \left\|\mathbf{A}^j\right\|_\rho \quad (14)\end{aligned}$$

Thus, we have demonstrated the "if" part of Equation (11); the "only if" part can be demonstrated in a similar way. Thus the computation of $\mathbf{T}^j$ can be substituted by $\frac{\mathbf{T}^j}{\|\mathbf{T}^j\|_\rho} = I - \frac{\mathbf{A}^j}{\|\mathbf{A}^j\|_\rho}$ when $\left\|\mathbf{A}^j\right\|_\rho = W^j$; nevertheless, since the second order tensor $\hat{\mathbf{T}}^j = I - \frac{\mathbf{A}^j}{\|\mathbf{A}^j\|_\rho}$ is generally positive semi-definite, we always compute $\hat{\mathbf{T}}^j$ as the vote at receiver j, also when the condition in Equation (11) does not hold. This is useful for it enforces the classified dimensionality d to be $0 < d < D$, and $\forall j, \left\|\hat{\mathbf{T}}^j\right\|_\rho \leq 1$.

In the NV algorithm, voting passes executed after the first use the tensorial information computed during the previous voting passes to generate oriented votes on target points. More precisely, each emitter $(\mathbf{p}_i, \mathbf{T}_i)$ generates tensor votes equal to $W^i\mathbf{T}_i$, where W^i is a weight computed according to an unnormalized gaussian decay function centered in $\mathbf{p}_i$. To face additive noise, and to cast strongest votes in the direction tangent to the underlying manifold, the decay function is elongated proportionally to a scale parameter σ on the tangent space, and proportionally to a noise standard deviation parameter γ ($\gamma \leq \sigma$) on the normal space. To build such a gaussian function, we at first classify the dimensionality of the underlying manifold by selecting the maximum saliency value, s_h; then, we synthesize the precision matrix $\mathbf{P}_i = \mathbf{X}_i\mathbf{\Lambda}_i\mathbf{X}_i^T$, describing the gaussian function, by setting the eigenvalues in the diagonal matrix $\mathbf{\Lambda}_i$ to $\lambda_k = \frac{1}{\sigma^2}$ for $1 \leq k \leq h$, and $\lambda_k = \frac{1}{\gamma^2}$ for $h < k \leq D$, while $\mathbf{X}_i$ is the matrix of the eigenvectors of the tensor obtained from the previous voting pass. Once the precision matrix is obtained the scaling values W^i are computed and used to cast tensorial votes.

In this setting, each emitter pair $(\mathbf{p}_i, \mathbf{T}_i)$ can be viewed as a weak classifier trained by iterating voting passes. After the training step, the vote $W^i\mathbf{T}_i^j$ casted from the emitter i to the receiver position $\mathbf{p}_j$, can be considered as the precision matrix of the unnormalized gaussian function $\mathcal{N}\left(\mathbf{x}; \mathbf{p}_j, W^i\mathbf{T}_i^j\right) = e^{-(\mathbf{x}-\mathbf{p}_j)^T W^i \mathbf{T}_i^j (\mathbf{x}-\mathbf{p}_j)/2}$ which is proportional to the generalized probability density function, $P(\mathbf{x} \in \mathcal{M}_{\mathbf{p}_j} | (\mathbf{p}_i, \mathbf{T}_i))$, associating to each point $\mathbf{x} \in \Re^D$ the probability to be on the manifold $\mathcal{M}_{\mathbf{p}_j}$ underlying the point $\mathbf{p}_j$, conditioned to $(\mathbf{p}_i, \mathbf{T}_i)$. Since:

$$\begin{aligned}P\left(\mathbf{x} \in \mathcal{M}_{\mathbf{p}_j}\right) &= \mathcal{N}\left(\mathbf{x}; \mathbf{p}_j, \mathbf{T}^j\right) = \mathcal{N}\left(\mathbf{x}; \mathbf{p}_j, \sum_{\forall i} W^i\mathbf{T}_i^j\right) \\ = \prod_{\forall i} \mathcal{N}\left(\mathbf{x}; \mathbf{p}_j, W^i\mathbf{T}_i^j\right) &= \prod_{\forall i} P\left(\mathbf{x} \in \mathcal{M}_{\mathbf{p}_j} | (\mathbf{p}_i, \mathbf{T}_i)\right) \quad (15)\end{aligned}$$

the accumulated vote $\mathbf{T}^j$ represents the joint distribution in $\mathbf{p}_j$; therefore, the strong dimensionality classifier obtained by employing $P\left(\mathbf{x} \in \mathcal{M}_{\mathbf{p}_j}\right)$ is an ensemble classifier made by combining the weak classifiers obtained by employing the $P\left(\mathbf{x} \in \mathcal{M}_{\mathbf{p}_j} | (\mathbf{p}_i, \mathbf{T}_i)\right)$ for each i.

3.1 Point clustering

Given a set of points $\mathcal{P} = \{\mathbf{p}_i\}$, the NV algorithm can be used to train an ensemble classifier. Each point $\mathbf{p}_i$ can then be classified according to the dimensionality of the inferred manifold going trough it; point clustering is therefore realized by grouping points belonging to inferred manifolds with the same dimensionality.

To the aim of manifold dimensionality inference, the NV algorithm starts from a set of pairs $(\mathbf{p}_i, \mathbf{T}_i)$, and iterates voting passes, to modify (train) the $\mathbf{T}_i$s by means of the information exchanged among neighboring points. This information diffusion process allows to improve the dimensionality and local orientation estimation, encoded in $\mathbf{T}_i$, with respect to the underlying manifold going trough $\mathbf{p}_i$.

After each voting pass the inferred dimensionality information is reinforced by applying a nonlinear filtering, as follows:

1. the eigensystem $\mathbf{T}_i = \mathbf{X}_i\mathbf{\Lambda}_i\mathbf{X}_i^T$ is computed;
2. the saliency values are computed, as described in Section 2;
3. the maximum saliency value, s_h, is found to select the most likely dimensionality of the underlying manifold;
4. a new eigenvalue matrix $\bar{\mathbf{\Lambda}}_i$ is generated by setting to zero the eigenvalues corresponding to the tangent directions, and to one the others;
5. the filtered tensor is $\bar{\mathbf{T}}_i = \mathbf{X}_i\bar{\mathbf{\Lambda}}_i\mathbf{X}_i^T$.

The described algorithm allows to efficiently classify the dimensionality of the underlying manifold going trough each input data point, so that points related to manifolds with different dimensionality can be separated. In section Section 4 results are shown in a synthetic case.

3.2 Image Inpainting with NV

In [10] a TVF-based image inpainting technique was proposed. In this section we describe how to solve this problem in a similar way, by using a NV-based technique.

Given an image region, a set Ω of $(2r+1 \times 2r+1)$-pixels sub-images (training patches) can be extracted; each patch can be seen as a vector in $\Re^{(2r+1)^2}$. The NV algorithm is then applied to Ω to obtain a set of trained tensors $\mathcal{T}$, that describe an estimate of the underlying manifold; therefore, they can be used to recover missing information.

A "patch vector" $\mathbf{p}$ containing unspecified coefficients identifies the submanifold $\mathcal{M} \subset \Re^{(2r+1)^2}$. To recover the missing information in $\mathbf{p}$, we select from the set of tensors $\mathcal{T} = \{(\mathbf{p}_i, \mathbf{T}_i)\}$ the pair $(\mathbf{p}_h, \mathbf{T}_h)$ so that the distance $\|\tilde{\mathbf{p}} - \tilde{\mathbf{p}}_h\|$ is minimum with respect to all the training patches[4]. Being d the estimated dimensionality of the manifold underlying $\mathbf{p}_h$, **TS** the tangent space identified by the tensor $\mathbf{T}_h$, and $\mathbf{e}_j$ $(1 \leq j \leq d)$ its column vectors, the inferred patch $\bar{\mathbf{p}}$ can be computed as follows:

$$\bar{\mathbf{p}} = \frac{(\tilde{\mathbf{p}}_h - \mathbf{p}_h) \cdot \bar{\mathbf{p}}'}{\|\tilde{\mathbf{p}}_h - \mathbf{p}_h\|^2} \bar{\mathbf{p}}' + \mathbf{p}_h \tag{16}$$

where

$$\bar{\mathbf{p}}' = \sum_{l=1}^{d} \mathbf{e}_l \cdot (\tilde{\mathbf{p}}_h - \mathbf{p}_h)\mathbf{e}_l \tag{17}$$

These equations compute $\bar{\mathbf{p}}$ as the point, nearest to $\tilde{\mathbf{p}}_h$, and constrained on the linear space $\mathrm{span}\left\langle \{\mathbf{e}_j\}_{j=1}^{d} \right\rangle \cap \mathcal{M}$.

Every patch $\bar{\mathbf{p}}$, inferred for a partially specified patch $\mathbf{p}$, is an estimate of the unknown real patch and contains the estimated pixel gray levels $\bar{\mathbf{p}}_{x,y}$. Inferring overlapping patches allows to generate different gray level estimates for the same pixel; their mean value is the maximum likelihood estimation of the unknown pixel gray level. Given an unknown image region, its boundary patches are the easiest to be recovered since they have less unknown coordinates.

Based on these considerations, the inpainting algorithm proceeds by iterating the following steps:

1. the external edge pixels, p_i, of the unknown region are identified morphologically;
2. for each p_i:
   (a) a partially specified patch is centered on p_i and its unknown pixel values are inferred;
   (b) the inferred pixel values are accumulated in a working image and a counter image is used to count the number of contributions for each pixel;
3. the maximum value mc in the counter image is found and the gray levels of pixels p_k with at least $\left\lceil \frac{8}{9} mc \right\rceil$ contributions are estimated by averaging;
4. the pixels p_k are removed from the unknown region.

These steps are iterated until the unknown region becomes empty. This algorithm stops after few steps and has proved to produce promising results, that are described in Section 4.

4 Results

In this section we report qualitative results obtained both by clustering through dimensionality classification, and by inpainting.

4.1 Clustering

To test our point clustering algorithm we create two manifolds, $\mathcal{M}$ and $\mathcal{N}$, of dimensionality d and e, respectively (with $d \neq e$); $\mathcal{M}$ and $\mathcal{N}$ are imbedded in $\Re^D$, and $\mathcal{M} \cap \mathcal{N} \neq \emptyset$. Two sets of points $\mathcal{P}_\mathcal{M}$ and $\mathcal{P}_\mathcal{N}$ are drawn from $\mathcal{M}$ and $\mathcal{N}$, respectively, and $\mathcal{P} = \mathcal{P}_\mathcal{M} \cup \mathcal{P}_\mathcal{N}$ is used as input to the NV algorithm, whose aim is to classify each point as belonging to either $\mathcal{M}$ or $\mathcal{N}$.

As an example, consider the dataset $\mathcal{P}$ depicted in Figure 2: it contains a curve and a surface that intersect in $\Re^3$. After executing the NV algorithm with 5 voting passes, tensors are oriented normally with respect to the underlying manifolds; therefore, we compute the saliency values of each tensor, and use them to infer the dimensionality of the manifold from which each point is drawn.

In Figure 3 the three saliency values for each data point are shown after the first and the last voting passes, respectively; note that after the last voting pass the saliency values are either ≈ 1, or ≈ 0. Therefore, they clearly identify the dimensionality of the manifold from which each point is drawn. In Figure 4 the clustered geometrical structures are shown; for this test we used $\sigma = 0.1$ and $\gamma = \sigma/10$.

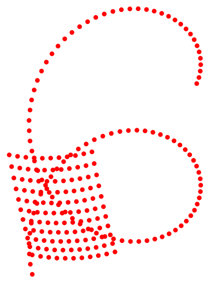


Figure 2. *The input dataset.*

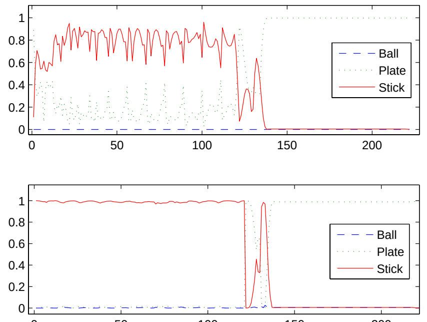


Figure 3. *Saliency values after 1 and 5 votings.*

4.2 Inpainting

We have tested our image inpainting algorithm on a set of 50 generic images, where the unknown region and the training region have been manually chosen. We have used 7×7 square patches, $\sigma = 0.2$, and $\gamma = \sigma/10$; experiments have proved that good inpainting results can be obtained by employing the first voting pass only, thus increasing the efficiency of the overall algorithm. In Figure 5 examples of inpainting results are shown. The inpainting algorithm

[4] We represent with the notation $\tilde{\mathbf{x}}$ the projection of a point $\mathbf{x} \in \Re^{(2r+1)^2}$ on $\mathcal{M}$.

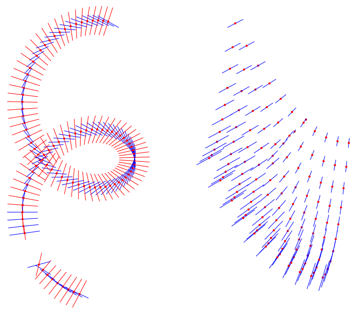


Figure 4. *The clustered geometrical structures and their normal directions.*

has demonstrated to be very efficient during the reconstruction step; the time complexity is indeed dominated by the training step. Using an Intel Centrino Duo 2.0GHz CPU with 2.0GB RAM, our Matlab implementation has taken on average 18.156s to infer, on average, 1171 pixel values per image, starting from a training set of about 550 training patches.

4.3 Conclusions and future works

In this work we have described a new normal space inference technique, inspired to the tensor voting framework, that employs an ensemble classifier. Our technique is less precise but more efficient, and its effectiveness has been proved by experimental results.

In the future we aim to improve the point clustering algorithm by adding spatial information, to separate structures of the same dimensionality, and by using a distance function between orientations, to separate intersecting structures of the same dimensionality. The inpainting algorithm is promising; its extension to color images is straight forward, and it can be made more efficient by computing the tensors only when needed, and storing them in a lazy-evaluation fashion.

Finally, considering that the NV algorithm trains weak classifiers by allowing them to interact through an information diffusion process, we plan to further explore ensemble methods where weak classifiers interact also during the training phase.

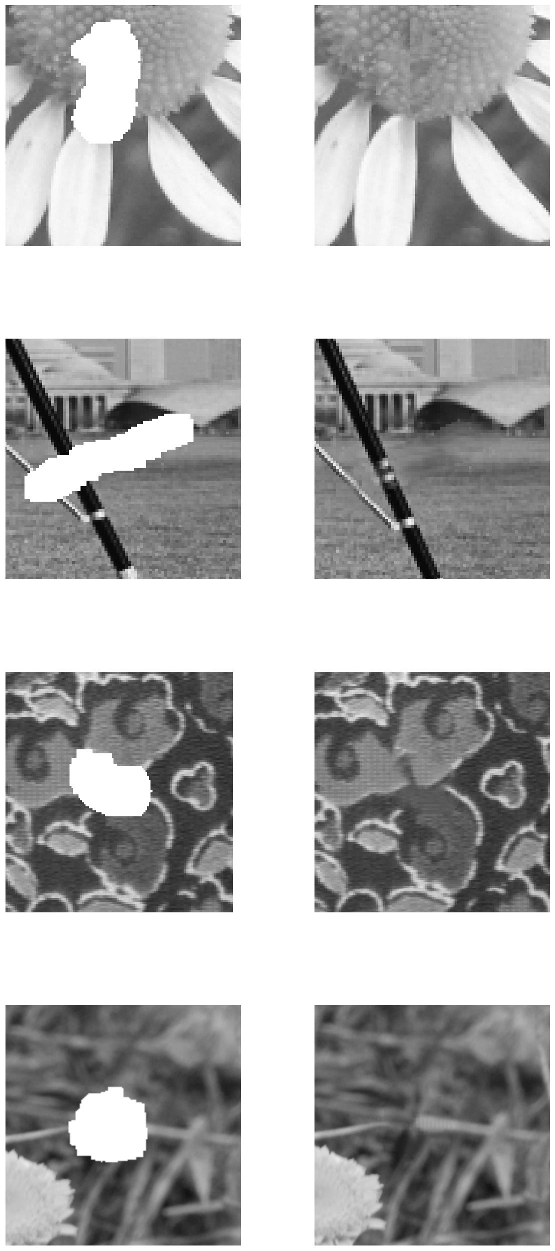


Figure 5. *Left: images with unknown regions. Right: inpaintin results.*

References

[1] C. K. Tang G. Medioni, 'Curvature-augmented tensor voting for shape inference from noisy 3d data', *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, No. 6*, 858–863, (June 2002).

[2] G. Guy G. Medioni, 'Inferring global perceptual contours from local features', *International journal of Computer Vision*, 113–133, (1996).

[3] P. Mordohai G. Medioni, 'Dense multiple view stereo with general camera placement using tensor voting', *3D Data Processing, Visualization and Transmission*, 725–732, (September 2004).

[4] P. Mordohai G. Medioni, 'Dimensionality estimation and manifold learning using tensor voting', Technical report, Institute for Robotics and Intelligent Systems, University of Southern California, (2005).

[5] P. Mordohai G. Medioni, 'Stereo using monocular cues within the tensor voting framework', *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, No. 6*, 968–982, (June 2006).

[6] P. Mordohai G. Medioni, *Tensor Voting: A Perceptual Organization Approach to Computer Vision and Machine Learning*, Morgan and Claypool Publishers, first edn., 2006.

[7] G. Medioni S. B. Kang, *Emerging topics in Computer Vision*, Prentice Hall, first edn., 2004.

[8] W. S. Tong at al., 'First order augmentation to tensor voting for boundary inference and multiscale analysis in 3d', *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, No. 5*, 858–863, (May 2004).

[9] Paola Campadelli and Gabriele Lombardi, 'Tensor voting fields: Direct votes computation and new saliency functions', in *ICIAP*, ed., Rita Cucchiara, pp. 677–684. IEEE Computer Society, (2007).

[10] Jiaya Jia and Chi-Keung Tang, 'Image repairing: Robust image synthesis by adaptive nd tensor voting', *CVPR*, **01**, 643, (2003).

[11] Mircea Nicolescu and Gérard G. Medioni, 'Motion segmentation with accurate boundaries - a tensor voting approach.', in *CVPR (1)*, pp. 382–389. IEEE Computer Society, (2003).

Multi-Class Modeling with Ensembles of Local Models for Imbalanced Misclassification Costs

Sebastian Nusser [1, 2] and **Clemens Otte** [1] and **Werner Hauptmann** [1]

Abstract. In this paper, we will discuss different strategies of extending an ensemble approach based on local binary classifiers to solve multi-class problems. The ensembles of binary classifiers were developed with the objective of providing interpretable local models for use in safety-related application domains. The ensembles assume highly imbalanced misclassification costs between the two classes. The extension to multi-class problems is not straightforward because common multi-class extensions might induce inconsistent decisions. We propose an approach that avoids such inconsistencies by introducing a hierarchy of misclassification costs. We will show that by following such a hierarchy it becomes feasible to extend the binary ensemble and to achieve a good predictive performance.

1 Introduction

Safety-related systems are systems whose malfunction or failure may lead to death or serious injury of people, loss or severe damage of equipment, or environmental harm. They are deployed, for instance, in aviation, automotive industry, medical systems and process control. In [13] we proposed a binary ensemble framework for use in safety-related domains. The main design criterion of this approach is to provide an ensemble of binary classification models that use small subspaces of the complete input space enabling the visual interpretation of the models. Because machine learning approaches are regarded with suspiciousness in the field of safety-related domains, the possibility to visualize each local model greatly facilitates the domain experts' acceptance of the data-driven generated models. An interpretable solution is often required for applications where the available training data is too sparse and the number of input dimensions is too large to sufficiently apply statistical risk estimation methods in practical application tasks. In most cases, high-dimensional models are required to solve a given problem. Unfortunately, such high-dimensional models are hard to verify (*curse of dimensionality*), may tend to overfitting, and the interpolation and extrapolation behavior is often unclear. An example of such counterintuitive and unintended behavior is illustrated in Fig. 1, where the prediction of the model changes in a region not covered by the given data set. Such behavior becomes even more likely and much more difficult to discover in the high-dimensional case. Our ensemble approach provides an insight into each local model, which can be evaluated according domain knowledge and, thus, the correct interpolation and extrapolation behavior of the model can be guaranteed. The extension of our binary classification ensemble to multi-class problems is

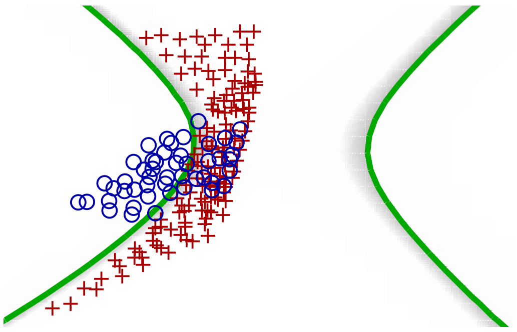


Figure 1. Counterintuitive extrapolation behavior in a region not covered by the given data set. This two-class problem is solved by a support vector machine (SVM) with an acceptable classification performance on the given data. However, in a region not covered by any data the decision of the SVM changes arbitrarily.

not straightforward, since commonly used methods like one-against-one or one-against-rest voting [5, 10] may introduce inconsistencies. We will show that such inconsistencies can be avoided by introducing a hierarchy of misclassification costs. The crucial aspect is to find a suitable trade-off between the generation of an interpretable and verifiable model and the realization of a high predictive accuracy. In most situations, more complex models will be able to achieve a better predictive performance on the available data compared to simpler models. However, a higher complexity of the model will usually lead to an increased effort for model verification.

This contribution is organized as follows: in Section 2 we recall our binary ensemble approaches for use in safety-related domains. In Section 3, two commonly used approaches of extending binary classifiers to multi-class problems are briefly discussed and the inconsistencies are illustrated that might arise when applying them. Section 4 extends the binary classification framework to also solve multi-class problems. Experiments on well-known benchmark problems are discussed in Section 5 and Section 6 concludes.

2 The Binary Ensemble Framework

This section introduces the basic concepts of our ensemble approach. In Section 4, these algorithms are extended to solve multi-class problems. The algorithms are designed to solve binary classification problems. The task is to find an estimate of the unknown function $f : V^n \to Y$, where $V^n = \prod_{i=1}^{n} X_i$ with $X_i \subseteq \mathbb{R}$ is the input space and Y is the target value, given an observed data set: $\mathfrak{D} = \{(\vec{v}_1, y_1), ..., (\vec{v}_m, y_m)\} \subset V^n \times Y$.

Basic Idea. Our ensemble framework, which was introduced in [13], is motivated by Generalized Additive Models [14, 9] and

[1] Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81730 Munich, Germany, email: {sebastian.nusser.ext, clemens.otte, werner.hauptmann} @siemens.com

[2] School of Computer Science, Otto-von-Guericke-University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

separate-and-conquer approaches [7]. It can be interpreted as a variant of the projection pursuit [6, 11]. Both approaches are designed to find an estimate of the unknown function $f : V^n \longrightarrow \mathbb{K}$, where $\mathbb{K} \subset \mathbb{N}$ is the set of class labels. Our approaches are based on the projections of the high-dimensional data to low-dimensional subspaces. *Local models* g_j are trained on these subspaces. By regarding only low-dimensional subspaces a visual interpretation becomes feasible and, thus, the avoidance of unintended extrapolation behavior is possible. The ensemble of local models boosts the overall predictive accuracy and overcomes the limited predictive performance of each single local model, while the global model remains interpretable.

Projection of High-Dimensional Data Set. The projection π maps the n-dimensional input space V^n to an arbitrary subspace of V^n. This mapping is determined by a given index set $\beta \subset \{1, ..., n\}$. The index set defines the dimensions of V^n that will be included in the subspace V_β. Thus, the projection π on the input space V^n given the index set β is defined as:

$$\pi_\beta(V^n) = V_\beta = \prod_{\imath \in \beta} X_\imath \,. \tag{1}$$

Local Models. The j-th local model is defined as:

$$g_j : \pi_{\beta_j}(V^n) \rightarrow \mathbb{K}, \tag{2}$$

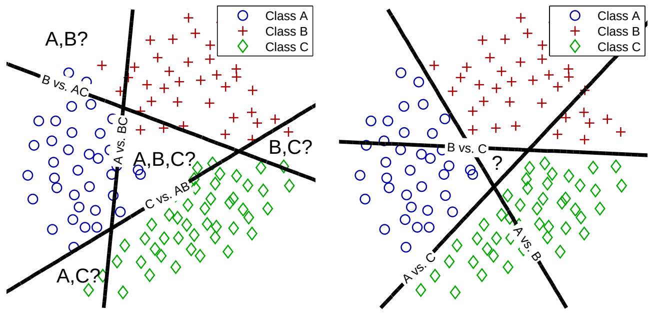
where β_j denotes the index set of the subspace where the classification error of the local model g_j is minimal. The best projections are determined by a wrapper method for feature selection [12]. The final function estimate $\hat{f}$ of the global model is determined by the aggregation of the results of all local models $g_j(\pi_{\beta_j}(\vec{v}))$.

DecisionTree-like Ensemble. This approach replaces the classification nodes in a common decision tree by strong classifiers. The classification nodes in the tree are restricted to two input dimensions. This facilitates the visualization of the relevant decision region and avoids overfitting. The best local model g_j is used to divide the training set into new subsets $\mathfrak{D}_\theta^{new} := \{(\vec{v}, y) | g_j(\pi_{\beta_j}(\vec{v})) = \theta\}$, where $\theta \in \mathbb{K}$. The local models for these subsets are built recursively, until an appropriate termination criterion is fulfilled. The leaf nodes of the tree represent the final classification labels.

Non-hierarchical Ensemble. This method incorporates prior knowledge about the subgroups of the given problem and avoids hierarchical dependencies of the local models as in the DecisionTree-like Ensemble approach. It is required that the so-called *preferred class* c_{pref} must not be misclassified by any of the trained local models: $\forall y = c_{\text{pref}} : |y - g(\pi_\beta(\vec{v}))| \overset{\text{def}}{=} 0$. This requirement typically leads to imbalanced misclassification costs. The local models are trained on low-dimensional projections of the high-dimensional input space with the objective to avoid the misclassification of the preferred class. The local models greedily separate the samples of the other class from the preferred class samples. Missed samples of the other class are used to build further sub-experts. This yields the following final function estimate: $\hat{f}(\vec{v}) = \bigvee_{g_j \in Exp} g_j(\pi_{\beta_j}(\vec{v}))$, where Exp is the set of all local models. For the sake of simplicity it is defined that the preferred class c_{pref} is always encoded as 0 and the other class is always encoded as 1 by the local models g_j.

3 Multi-Class Extensions of Binary Classifiers

There are two commonly used approaches to extend binary classifiers to solve multi-class problems: (1) a one-against-one extension and (2) a one-against-rest extension. A detailed comparison of these methods and an experimental evaluation for support vector machines is given in [10]. Figure 2 illustrates both approaches.



(a) One-against-rest extension. (b) One-against-one extension.

Figure 2. Illustration of multi-class extensions based on binary classifiers. There are three classes: A, B, C. The discriminant functions are given as black lines. Regions with possible inconsistent decisions are labeled with question marks.

One-against-rest multi-class extension. This method constructs k classifiers, where $k = |\mathbb{K}|$ is the number of classes. The model $f_{c_\mathfrak{k}}$ for class $c_\mathfrak{k} \in \mathbb{K}$ is trained on all samples of class $c_\mathfrak{k}$ against all samples from the remaining classes which are combined to a new class $c_\mathfrak{k}^* = \mathbb{K} \setminus c_\mathfrak{k}$, for the sake of simplicity the class label of $c_\mathfrak{k}^*$ is set to -1. A new data point $\vec{v}$ is assigned to: $f(\vec{v}) = \arg\max_{c \in \mathbb{K}} f_c(\vec{v})$.

One-against-one multi-class extension. This method builds $k(k-1)/2$ classifiers, each for the pair-wise combination of the classes $c_\mathfrak{k}, c_\mathfrak{l} \in \mathbb{K}$, $\mathfrak{k} \neq \mathfrak{l}$. The final classification is performed by majority voting – that is the most frequent predicted class label is returned as prediction of the multi-class model.

Risk of inconsistent decisions. The issue of inconsistent decisions of combining binary classifiers to multi-class classifiers is addressed in [16], for instance. As illustrated in Figure 2, there can be regions of the input space where the decision of the multi-class models might be inconsistent. Those regions are marked with question marks in each figure. For the *one-against-rest method*, there are two possibilities of an inconsistent decision: (1) there are several binary classifiers predicting different class labels for one given data point. Such regions are (A,B ?), (A,C ?), (B,C ?). (2) there are regions, where all classifiers are predicting the "rest" class, (A,B,C ?). For the *one-against-one method*, there is only one kind of inconsistent decisions possible: several binary classifiers are predicting different class label for one given data point. The problem of several classifiers predicting different class labels can be solved by assigning the class label at random [10] or to assign the data point to the class with the highest posterior probability [16]. The second kind of inconsistent decisions of the one-against-rest method can be acceptable for some problems, where "no decision" might be better than a "wrong decision". Otherwise, one can use the same strategy as for the other kind of inconsistent decisions.

4 The Multi-Class Ensemble Framework

Our ensemble framework leads to the following two levels where the binary classification approaches can be extended to solve multi-class problems:

1. The multi-class decision is made on the level of the local models (Local Multi-Class Ensemble).

Table 1. Confusion Matrix for multi-class local models in a Non-hierarchical Ensemble. The following hierarchy of misclassification costs is assumed: $penalty(c_1) > penalty(c_2) > penalty(c_3) > penalty(c_4) > penalty(...)$

	predicted class				
true class	c_1	c_2	c_3	c_4	...
c_1	$n_{1,1}$	0	0	0	...
c_2	$n_{2,1}$	$n_{2,2}$	0	0	...
c_3	$n_{3,1}$	$n_{3,2}$	$n_{3,3}$	0	...
c_4	$n_{4,1}$	$n_{4,2}$	$n_{4,3}$	$n_{4,4}$	...
...	...	...	...	...	...

2. Local models are binary classifiers, the multi-class classification task is performed by the ensemble. There are two variants:
   (a) the One-versus-Rest Ensemble and
   (b) the Hierarchical Separate-and-Conquer Ensemble .

Another algorithm based on one-against-one classifier combination is given in [15]. This algorithm uses a one-against-one approach to extend binary classifiers to solve multi-class problems. The important similarity of this approach to our framework is that it is also based on a dimensionality-reduction on the level of local models. In contrast to our approach, the number of dimensions of the local models is not limited – all input dimensions that provide statistically sufficient information are included in the training set to build a single local model to separate the pair of classes. Our approach may use several local models with limited dimensionality to solve the same subproblem, while each local model remains visually interpretable.

For safety-related problems it is important to take into account that the commonly used strategies of extending binary classifiers to multi-class classifiers, which are illustrated in Figure 2, may lead to regions with inconsistent decisions. In order to avoid an unintended labeling the inconsistent decisions are solved according to a hierarchy of misclassification costs: for a given new data point $\vec{v}$ that class label of all predicted class labels is chosen which has the largest misclassification penalty.

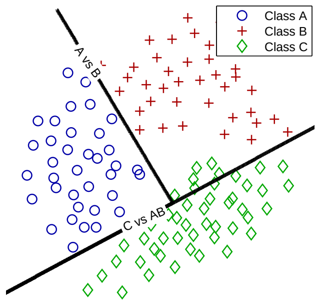
Local Multi-Class Ensemble. Using local multi-class models in a *Non-hierarchical Ensemble* requires a hierarchy of misclassification costs, i.e. it is assumed that there exists an *ordering* of the class labels, which allows statements like: *"class c_1 samples should never be misclassified, class c_2 samples might be misclassified only as class c_1 samples, class c_3 might be classified as class c_1 or c_2 samples, ..."*

$$penalty(c_1) > penalty(c_2) > penalty(c_3) > ... \qquad (3)$$

Such a hierarchy of misclassification costs leads to a confusion matrix as depicted in Table 1. This issue is closely related to ordinal classification problems. An SVM-based approach for ordinal classification can be found in [2].

Combining several local multi-class models becomes difficult because one can only rely on the prediction of the class $c_\mathfrak{k}$, which has the minimal misclassification cost – all other class label predictions might be false positives. Thus, it is necessary to include all samples that are not predicted as class $c_\mathfrak{k}$ in the training for the next local model. This fact leads directly to the Hierarchical Separate-and-Conquer Ensemble approach, which is described in the following paragraph.

The extension of the DecisionTree-like Ensemble approach to solve a multi-class problem is straightforward – a novel subtree is generated for each class predicted by the local model of the current node. The final classification decision is determined by the leaf node

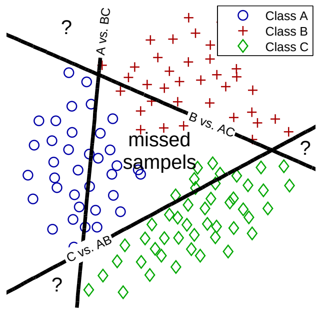


(a) Discriminant functions.

	predicted class		
true class	A	B	C
A	36	0	0
B	0	38	0
C	0	6	41

(b) Confusion matrix.

Figure 3. Hierarchical Separate-and-Conquer Ensemble trained on the data from Figure 2. The following hierarchy of misclassification costs is assumed: $penalty(A) > penalty(B) > penalty(C)$.



(a) Discriminant functions. Ambiguous regions are labeled with '?'.

	predicted class			
true class	A	B	C	?
A	22	0	0	14
B	0	31	0	7
C	0	0	41	6

(b) Confusion matrix. The last column denotes missed samples.

Figure 4. One-versus-Rest Ensemble trained on the data from Figure 2. Each model for class $c_\mathfrak{k}$ is trained with the objective to avoid the misclassification of the samples belonging to $c^*_\mathfrak{k} = \mathbb{K} \setminus c_\mathfrak{k}$.

of the learned tree – similar to standard decision tree approaches. To avoid inconsistent decisions it is encouraged to also use a hierarchy of misclassification costs in this approach.

Hierarchical Separate-and-Conquer Ensemble. This approach requires a hierarchy of the misclassification costs as already introduced for the Local Multi-Class Ensemble approach. It is related to the commonly used one-against-rest approach. Instead of building all one-against-rest combinations of models, the class with the minimal classification costs is separated from all samples of the other classes via binary local models. This approach is illustrated in Figure 3. The procedure is the same as for the Non-hierarchical Ensemble, which is described in Section 2. If the problem is solved for this class or there are no further improvements possible, all samples of this class are removed from the training data set and the procedure is repeated for the class which has now the smallest misclassification costs. This procedure is repeated until the data set of the next iteration has only a single class label. The resulting binary classifiers are evaluated according to the misclassification hierarchy, that is in the first step all local models of the class with minimal misclassification costs are evaluated. If the novel sample cannot be assigned to the class with minimal misclassification costs, the procedure is repeated for the next class in the hierarchy of misclassification costs. If no local model predicts the novel sample the sample is assigned to the class with maximal misclassification costs.

One-versus-Rest Ensemble. This approach follows the one-against-rest multi-class classification approach. It is illustrated in Figure 4. For every class $c_\mathfrak{k} \in \mathbb{K}$ versus $c^*_\mathfrak{k} = \mathbb{K} \setminus c_\mathfrak{k}$ a complete binary Non-hierarchical Ensemble $\hat{f}_{c_{\mathfrak{k}}}(\vec{v})$ is trained. The class $c^*_{\mathfrak{k}}$ is chosen as the preferred class c_{pref} to avoid the misclassification of any sample belonging to $\mathbb{K} \setminus c_{\mathfrak{k}}$. For the sake of simplicity $c^*_{\mathfrak{k}}$ is encoded as -1. The resulting binary models can be combined by determining the maximum: $\hat{f}(\vec{v}) = \arg\max_{c_{\mathfrak{k}} \in \mathbb{K}} \hat{f}_{c_{\mathfrak{k}}}(\vec{v})$.

This approach is the easiest way to extend the binary local modeling approach to multi-class modeling, but it shows a lack of performance for overlapping data sets: it is possible that certain data points will be assigned to the class $c^*_{\mathfrak{k}}$ by every local model and some classes cannot be separated from the other classes due to overlapping of the classes in all projections. This approach still yields ambiguous decisions within the input space, as shown in Figure 4. Such ambiguities can be resolved by the hierarchy of misclassification costs.

5 Experiments

Table 2 summarizes the experiments performed on several benchmark data sets. All data sets can be obtained from [1] – except for the CUBESMULT data set. The predictive error is estimated by a 10-fold-crossvalidation procedure where the error is computed on the complete data set. For every data set we used 10 different fold initializations. Two error types are determined: (1) the relative misclassification error, which represents the number of all misclassified samples, and (2) the critical error, which represents the relative number of samples that do not satisfy the hierarchy of misclassification costs. The critical error is the more important error measurement because it corresponds to a violation of given domain knowledge.

Our ensemble methods are compared with a standard SVM implementation (libSVM) from [3] and a CART classification tree (`treefit` in Matlab). In Table 2 the Hierarchical Separate-and-Conquer Ensemble is abbreviated as HSCE, the Local Multi-Class Ensemble is abbreviated as LMCE, and the One-versus-Rest Ensemble is abbreviated as OvRE. The local models of our ensemble models are SVMs with Gaussian kernels. The parameter sets of the SVMs are chosen manually in order to obtain smooth decision surfaces in the local models. The same parameter sets are used for the high-dimensional SVM. For feature selection, our ensemble performs a search through all possible pairs of features.

CUBESMULT data set. The task of this artificial data set is to solve a four-class problem. The samples of CLASS 1 are drawn from $N(e_n + i \cdot 0.5, 0.2 \cdot \mathbf{I})$, $i \in \{0,1,2\}$, where e_i is a unit vector in $\mathbb{R}^3$ and $\mathbf{I}$ is the identity matrix. The samples of CLASS 2, CLASS 3, and CLASS 4 are drawn from $N((0.0,0.0,0.0)^T, 0.2 \cdot \mathbf{I})$, $N((0.5,0.5,0.5)^T, 0.2 \cdot \mathbf{I})$, and $N((1.0,1.0,1.0)^T, 0.2 \cdot \mathbf{I})$, respectively. For this problem the following hierarchy of misclassification costs is assumed: $\mathit{penalty}(\text{CLASS 4}) > \mathit{penalty}(\text{CLASS 3}) > \mathit{penalty}(\text{CLASS 2}) > \mathit{penalty}(\text{CLASS 1})$.

FISHER'S IRIS data set. This well-known data set from [4] contains three classes (IRIS SETOSA – CLASS 1, IRIS VERSICOLOR – CLASS 2, and IRIS VIRGINICA – CLASS 3) of 50 instances each. The input space consists of four numeric attributes. CLASS 1 is linearly separable from the other two classes; CLASS 2 and CLASS 3 are not linearly separable from each other. The following hierarchy of misclassification costs is assumed: $\mathit{penalty}(\text{CLASS 3}) > \mathit{penalty}(\text{CLASS 1}) > \mathit{penalty}(\text{CLASS 2})$.

WINE data set. This data set consists of the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. $\mathit{penalty}(\text{CLASS 3}) > \mathit{penalty}(\text{CLASS 2}) > \mathit{penalty}(\text{CLASS 1})$ is assumed as hierarchy of misclassification costs.

Table 2. 10-fold crossvalidation on multi-class problems. The models are trained on 90% of the given data set and the error is estimated on the whole data set. The error is averaged over 10 random fold initializations. #M denotes the number of all models within the ensemble or decision nodes within the classification tree. #D denotes the dimensionality of each model or decision node. The error represents the relative number of misclassified samples. The critical error represents the number of samples that violate the hierarchy of misclassification costs.

Method	#M	#D	Error		Critical Error	
			mean (std)	min	mean (std)	min
			CUBESMULT data set			
HSCE	4	2	4.05% (0.33)	2.93%	**0.02**% (0.05)	0.00%
LMCE	4	2	3.21% (0.86)	2.00%	1.77% (1.11)	0.53%
OvRE	2	2	43.32% (0.35)	42.13%	**0.02**% (0.04)	0.00%
libSVM	1	3	**2.96**% (0.24)	2.40%	**0.02**% (0.05)	0.00%
`treefit`	23	1	4.05% (1.31)	1.60%	0.39% (0.22)	0.00%
			FISHER'S IRIS data set			
HSCE	2	2	6.37% (1.56)	2.67%	0.06% (0.21)	0.00%
LMCE	1	2	3.50% (0.47)	2.67%	1.92% (0.61)	0.67%
OvRE	3	2	12.32% (2.21)	7.33%	0.06% (0.21)	0.00%
libSVM	1	4	**3.04**% (0.35)	2.67%	**0.05**% (0.18)	0.00%
`treefit`	6	1	6.37% (1.05)	4.00%	0.18% (0.35)	0.00%
			WINE data set			
HSCE	4	2	4.20% (1.00)	1.69%	0.03% (0.13)	0.00%
LMCE	4	2	1.62% (0.61)	0.56%	0.95% (0.39)	0.00%
OvRE	7	2	10.34% (1.78)	6.18%	**0.01**% (0.06)	0.00%
libSVM	1	13	**0.19**% (0.31)	0.00%	0.08% (0.20)	0.00%
`treefit`	7	1	2.75% (1.19)	0.56%	0.25% (0.42)	0.00%
			DERMATOLOGY data set			
HSCE	7	2	6.77% (0.66)	4.92%	0.23% (0.20)	0.00%
LMCE	10	2	1.39% (0.62)	0.55%	1.08% (0.45)	0.55%
OvRE	12	2	20.14% (2.30)	15.30%	**0.03**% (0.09)	0.00%
libSVM	1	33	**0.30**% (0.26)	0.00%	0.17% (0.21)	0.00%
`treefit`	19	1	3.14% (0.93)	1.37%	0.49% (0.31)	0.00%
			SEGMENTATION data set			
HSCE	11	2	3.83% (0.69)	2.38%	0.55% (0.46)	0.00%
LMCE	8	2	3.35% (1.55)	0.48%	2.43% (1.40)	0.48%
OvRE	14	2	18.97% (2.22)	13.81%	**0.34**% (0.39)	0.00%
libSVM	1	18	**1.12**% (0.74)	0.00%	0.58% (0.54)	0.00%
`treefit`	16	1	11.02% (2.79)	4.29%	0.72% (0.70)	0.00%
			POST-OP data set			
HSCE	4	2	25.13% (1.21)	23.33%	0.27% (0.50)	0.00%
LMCE	1	2	25.74% (0.53)	25.56%	2.36% (0.40)	2.22%
OvRE	4	2	82.07% (3.78)	71.11%	**0.04**% (0.27)	0.00%
libSVM	1	7	**23.23**% (0.78)	22.22%	0.34% (0.56)	0.00%
`treefit`	16	1	73.67% (9.20)	52.22%	0.82% (0.92)	0.00%

DERMATOLOGY data set. This example is an challenging problem in dermatology [8]. The task is to discriminate six differential diagnostics of erythemato-squamous diseases. The data set consists of 366 records and each record has 33 attributes. The age attribute of the original data set from the UCI Machine Learning Repository [1] is omitted here, because it has some missing values. The following hierarchy of misclassification costs is assumed: $\mathit{penalty}(\text{CLASS 6}) > \mathit{penalty}(\text{CLASS 5}) > \mathit{penalty}(\text{CLASS 4}) > \mathit{penalty}(\text{CLASS 3}) > \mathit{penalty}(\text{CLASS 2}) > \mathit{penalty}(\text{CLASS 1})$.

SEGMENTATION data set. The 2310 instances of this data set are drawn randomly from a database of seven outdoor images. Each instance is described by 19 continuous attributes. The third attribute is ignored because it does not change for all instances. The task is to distinguish six different surface textures. The following hierarchy of misclassification costs is assumed: $\mathit{penalty}(\text{CLASS 6}) > \mathit{penalty}(\text{CLASS 5}) > \mathit{penalty}(\text{CLASS 4}) > \mathit{penalty}(\text{CLASS 3}) >$

$penalty(\text{CLASS 2}) > penalty(\text{CLASS 1})$.

POST-OP data set. The classification task of this database is to determine where patients in a postoperative recovery area should be sent to next: PATIENT SENT TO INTENSIVE CARE UNIT – CLASS 1, PATIENT SENT TO GENERAL HOSPITAL FLOOR – CLASS 2, and PATIENT PREPARED TO GO HOME – CLASS 3. There are 90 instances and eight attributes. The 'comfort' attribute is ignored within the experiments because it has missing values. The following hierarchy of misclassification costs is assumed: $penalty(\text{CLASS 1}) > penalty(\text{CLASS 2}) > penalty(\text{CLASS 3})$.

Discussion. For almost all data sets the One-versus-Rest Ensemble achieves the best performance by regarding the critical error. On the other hand, the overall error of this approach is worse compared to all other methods. This poor performance is due to a large number of samples that are missed by the local models and are assigned to the "other" class. Nevertheless, samples that are labeled as "unrecognized" might be acceptable for some application problems. The Hierarchical Separate-and-Conquer Ensemble approach provides a good trade-off between the predictive performance – on both, the overall error and the critical error – and the interpretation of the models compared to the SVM solution that achieves the least overall error on all data sets but incorporates always the complete input space. The critical error of the Local Multi-Class Ensemble approach is quite large in all experiments because the hierarchy of misclassification costs is ignored within our experiments. On the other hand, this ensemble approach is the only variant that does not require a hierarchy of misclassification costs to build a multi-class model. While interpreting the decision boundaries of a high-dimensional SVM is quite infeasible, all ensemble approaches allow a visualization of the local models and, thus, facilitate the incorporation of domain knowledge via an interactive model selection process. The incorporation of domain knowledge can yield a higher accuracy of the solution as shown in Table 2 with the columns that show the minimal predictive errors of each approach.

6 Conclusions

To apply machine learning approaches in the field of safety-related problems it is crucial to provide interpretable and verifiable models. Unfortunately, it is infeasible to interpret high-dimensional models sufficiently. Therefore, for safety-related problems, high-dimensional models are not to be applied. On the other hand, simple models which are easier to interpret show a lack of predictive performance. The framework proposed in this paper provides a good trade-off between the interpretation and verification of the learned (local) models, avoiding an unintended extrapolation behavior, and the achievement of a high predictive accuracy. Each local model can be interpreted visually and the ensemble of the local models compensates for the limited predictive performance of each single local model. In contrast to dimensionality reduction methods, which combine several dimensions of the input space, the local models are trained on the original dimensions, allowing domain experts to evaluate the trained models directly. In our experiments, the data sets from the UCI Machine Learning Repository were successfully tested, providing good results.

REFERENCES

[1] Arthur Asuncion and David J. Newman. UCI Machine Learning Repository, 2007. Available at *http://www.ics.uci.edu/~mlearn/MLRepository.html.*

[2] Jaime S. Cardoso, Joaquim F. Pinto da Costa, and Maria J. Cardoso, 'Modelling ordinal relations with SVMs: An application to objective aesthetic evaluation of breast cancer conservative treatment', *IEEE Transactions on Neural Networks*, **18**(5-6), 808–817, (2005).

[3] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at *http://www.csie.ntu.edu.tw/~cjlin/libsvm.*

[4] Ronald A. Fisher, 'The use of multiple measurements in taxonomic problems', *Annals of Eugenics*, **7**, 179–188, (1936).

[5] Jerome H. Friedman, 'Another approach to polychotomous classification', Technical report, Department of Statistics, Stanford University, (1996).

[6] Jerome H. Friedman and John W. Tukey, 'A projection pursuit algorithm for exploratory data analysis', *IEEE Transactions on Computers*, **23**(9), 881–890, (1974).

[7] Johannes Fürnkranz, 'Separate-and-conquer rule learning', *Artificial Intelligence Review*, **13**(1), 3–54, (1999).

[8] H. Altay Güvenir, Gülsen Demiröz, and Nilsel Ilter, 'Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals', *Artificial Intelligence in Medicine*, **13**(3), 147–165, (1998).

[9] Trevor Hastie and Robert Tibshirani, *Generalized Additive Models*, Chapman&Hall, 1990.

[10] Chih-Wei Hsu and Chih-Jen Lin, 'A comparison of methods for multiclass support vector machines', *IEEE Transactions on Neural Networks*, **13**(2), 415–425, (2002).

[11] Peter J. Huber, 'Projection pursuit', *The Annals of Statistics*, **13**(2), 435–475, (1985).

[12] Ron Kohavi and George H. John, 'Wrappers for feature subset selection', *Artificial Intelligence*, **97**(1-2), 273–324, (1997).

[13] Sebastian Nusser, Clemens Otte, and Werner Hauptmann, 'Learning binary classifiers for applications in safety-related domains', in *Proceedings of 17th Workshop Computational Intelligence*, pp. 139–151. Universitätsverlag Karlsruhe, (2007).

[14] Charles J. Stone, 'Additive regression and other nonparametric models', *The Annals of Statistics*, **13**(2), 689–705, (1985).

[15] Gero Szepannek and Claus Weihs, 'Local modelling in classification on different feature subspaces', in *Industrial Conference on Data Mining*, pp. 226–238, Leipzig, Germany, (2006).

[16] David M. J. Tax and Robert P. W. Duin, 'Using two-class classifiers for multiclass classification', in *Proceedings of the 16th International Conference on Pattern Recognition*, volume 2, pp. 124–127, Quebec, Canada, (2002).

A Taxonomy and Short Review of Ensemble Selection

Grigorios Tsoumakas and **Ioannis Partalas** and **Ioannis Vlahavas**[1]

Abstract. Ensemble selection deals with the reduction of an ensemble of predictive models in order to improve its efficiency and predictive performance. The last 10 years a large number of very diverse ensemble selection methods have been proposed. In this paper we make a first approach to categorize them into a taxonomy. We also present a short review of some of these methods. We particularly focus on a category of methods that are based on greedy search of the space of all possible ensemble subsets. Such methods use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. This paper abstracts the key points of these methods and offers a general framework of the greedy ensemble selection algorithm, discussing its important parameters and the different options for instantiating these parameters.

1 Introduction

Ensemble methods [5] have been a very popular research topic during the last decade. They have attracted scientists from several fields including Statistics, Machine Learning, Pattern Recognition and Knowledge Discovery in Databases. Their popularity arises largely from the fact that they offer an appealing solution to several interesting learning problems of the past and the present.

First of all, ensembles *lead to improved accuracy* compared to a single classification or regression mode. This was the main motivation that led to the development of the ensemble methods area. Ensembles achieve higher accuracy than individual models, mainly through the correction of their uncorrelated errors. Secondly, ensembles *solve the problem of scaling inductive algorithms to large databases*. Most inductive algorithms are too computationally complex and suffer from memory problems when applied to very large databases. A solution to this problem is to horizontally partition the database into smaller parts, train a predictive model in each of the smaller manageable part and combine the predictive models. Thirdly, ensembles can *learn from multiple physically distributed data sets*. Often such data can't be collected to a single site due to privacy or size reasons. This problem can be overcome through the combination of multiple predictive models, each trained on a different distributed data set. Finally, ensembles are useful for *learning from concept-drifting data streams*. The main idea here is to maintain an ensemble of classifiers that are trained from different batches of the data stream. Combining these classifiers with a proper methodology can solve the problem of data expiration that occurs when the learning concept drifts.

Typically, ensemble methods comprise two phases: the *production* of multiple predictive models and their *combination*. Recent work [13, 9, 12, 7, 21, 4, 14, 1, 16, 24, 17], has considered an additional intermediate phase that deals with the reduction of the ensemble size prior to combination. This phase is commonly called *ensemble pruning*, *selective ensemble*, *ensemble thinning* and *ensemble selection*, of which we shall use the last one within this paper.

Ensemble selection is important for two reasons: *efficiency* and *predictive performance*. Having a very large number of models in an ensemble adds a lot of computational overhead. For example, decision tree models may have large memory requirements [13] and lazy learning methods have a considerable computational cost during execution. The minimization of run-time overhead is crucial in certain applications, such as in stream mining. Equally important is the second reason, predictive performance. An ensemble may consist of both high and low predictive performance models. The latter may negatively affect the overall performance of the ensemble. Pruning these models while maintaining a high diversity among the remaining members of the ensemble is typically considered a proper recipe for an effective ensemble.

The last 10 years a large number of very diverse ensemble selection methods have been proposed. In this paper we make a first approach to categorize them into a taxonomy. We hope that community feedback will help fine-tuning this taxonomy and shape it into a proper starting place for researchers designing new methods. In addition, we delve a little deeper into a specific category in this taxonomy: greedy search-based methods.

A number of ensemble selection methods that are based on a greedy search of the space of all possible ensemble subsets, have recently been proposed [13, 7, 4, 14, 1]. They use different directions for searching this space and different measures for evaluating the available actions at each state. Some use the training set for subset evaluation, while others a separate validation set. In this paper we attempt to highlight the salient parameters of greedy ensemble selection algorithms, offer a critical discussion of the different options for instantiating these parameters and mention the particular choices of existing approaches. The paper steers clear of a mere enumeration of particular approaches in the related literature, by generalizing their key aspects and providing comments, categorizations and complexity analysis wherever possible.

The remainder of this paper is structured as follows. Section 2 contains background material on ensemble production and combination. Section 3 presents the proposed taxonomy including a short account of methods in each category. The category of clustering-based methods is discussed at a greater detail, from a more critical point of view. Section 4 discusses extensively the category of greedy search-based ensemble selection algorithms. Finally Section 5 concludes this work.

[1] Dept. of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece, email: {greg,partalas,vlahavas}@csd.auth.gr

2 Background

This section provides background material on ensemble methods. More specifically, information about the different ways of producing models are presented as well as different methods for combining the decisions of the models.

2.1 Producing the Models

An ensemble can be composed of either *homogeneous* or *heterogeneous models*. Homogeneous models derive from different executions of the same learning algorithm. Such models can be produced by using different values for the parameters of the learning algorithm, injecting randomness into the learning algorithm or through the manipulation of the training instances, the input attributes and the model outputs [6]. Popular methods for producing homogeneous models are *bagging* [2] and *boosting* [18].

Heterogeneous models derive from running different learning algorithms on the same data set. Such models have different views about the data, as they make different assumptions about it. For example, a neural network is robust to noise in contrast with a k-nearest neighbor classifier.

2.2 Combining the Models

Common methods for combining an ensemble of predictive models include *voting*, *stacked generalization* and *mixture of experts*.

In voting, each model outputs a class value (or ranking, or probability distribution) and the class with the most votes is the one proposed by the ensemble. When the class with the maximum number of votes is the winner, the rule is called *plurality voting* and when the class with more than half of the votes is the winner, the rule is called *majority voting*. A variant of voting is weighted voting where the models are not treated equally as each of them is associated with a coefficient (weight), usually proportional to its classification accuracy.

Let x be an instance and m_i, $i = 1..k$ a set of models that output a probability distribution $m_i(x, c_j)$ for each class c_j, $j = 1..n$. The output of the (weighted) voting method $y(x)$ for instance x is given by the following mathematical expression:

$$y(x) = \arg\max_{c_j} \sum_{i=1}^{k} w_i m_i(x, c_j),$$

where w_i is the weight of model i. In the simple case of voting (unweighted), the weights are all equal to one, that is, $w_i = 1, i = 1..k$.

Stacked generalization [23], also known as *stacking* is a method that combines models by learning a meta-level (or level-1) model that predicts the correct class based on the decisions of the base level (or level-0) models. This model is induced on a set of meta-level training data that are typically produced by applying a procedure similar to k-fold cross validation on the training data. The outputs of the base-learners for each instance along with the true class of that instance form a meta-instance. A meta-classifier is then trained on the meta-instances. When a new instance appears for classification, the output of the all base-learners is first calculated and then propagated to the meta-classifier, which outputs the final result.

The mixture of experts architecture [10] is similar to the weighted voting method except that the weights are not constant over the input space. Instead there is a gating network which takes as input an instance and outputs the weights that will be used in the weighted voting method for that specific instance. Each expert makes a decision and the output is averaged as in the method of voting.

3 A Taxonomy of Ensemble Selection Algorithms

We propose the organization of the various ensemble selection methods into the following categories: a) Search-based, b) Clustering-based c) Ranking-based and d) Other.

3.1 Search Based Methods

The most direct approach for pruning an ensemble of predictive models is to perform a heuristic search in the space of the possible different model subsets, guided by some metric for the evaluation of each candidate subset. We further divide this category into two subcategories, based on the search paradigm: a) greedy search, and b) stochastic search. The former is among the most popular categories of ensemble pruning algorithms and is investigated at depth in Section 4. Stochastic search allows randomness in the selection of the next candidate subset and thus can avoid getting stuck in local optima.

3.1.1 Stochastic Search

Gasen-b [25] performs stochastic search in the space of model subsets using a standard genetic algorithm. The ensemble is represented as a bit string, using one bit for each model. Models are included or excluded from the ensemble depending on the value of the corresponding bit. Gasen-b performs standard genetic operations such as mutations and crossovers and uses default values for the parameters of the genetic algorithm. The performance of the ensemble is used as a function for evaluating the fitness of individuals in the population.

Partalas et al. [16] search the space of model subsets using a reinforcement learning approach. We categorize this approach into the stochastic search algorithms, as the exploration of the state space includes a (progressively reducing) stochastic element. The problem of pruning an ensemble of n classifiers has been transformed into the reinforcement learning task of letting an agent learn an optimal policy of taking n actions in order to include or exclude each classifier from the ensemble. The method uses the Q-learning [22] algorithm to approximate an optimal policy.

3.2 Clustering-based methods

The methods of this category comprise two stages. Firstly, they employ a clustering algorithm in order to discover groups of models that make similar predictions. Subsequently, each cluster is separately pruned in order to increase the overall diversity of the ensemble.

3.2.1 Giacinto et al., 2000

Giacinto et al. [9] employ Hierarchical Agglomerative Clustering (HAC) for classifier pruning. This type of clustering requires the definition of a distance metric between two data points (here classifiers). The authors defined this metric as the probability that the classifiers don't make coincident errors and estimate it from a validation set in order to avoid overfitting problems. The authors also defined the distance between two clusters as the maximum distance between two classifiers belonging to these clusters. This way they implicitly used the *complete link* method for inter-cluster distance computation. Pruning is accomplished by selecting a single representative classifier from each cluster. The representative classifier is the one exhibiting the maximum average distance from all other clusters.

HAC returns a hierarchy of different clustering results starting from as many clusters as the data points and ending at a single cluster containing all data points. This raises the problem of how to chose the best clustering from this hierarchy. They solve this problem as follows: For each clustering result they evaluate the performance of the pruned ensemble on a validation set using majority voting as the combination method. The final pruned ensemble is the one that achieves the highest classification accuracy.

They experimented on a single data set, using heterogeneous classifiers derived by running different learning algorithms with different configurations. They compared their approach with *overproduce and choose* strategies and found that their approach exhibits better classification accuracy.

This approach is generally guided by the notion of diversity. Diversity guides both the clustering process and the subsequent pruning process. However, the authors use the classification accuracy with a specific combination method (majority voting) to select among the different clustering results. This reduces the generality of the method, as the selection is optimized towards majority voting. Of course this could be easily alleviated by using at that stage the method that will be later used for combining the ensemble.

In addition, the authors used a specific distance metric to guide the clustering process, while it would be interesting to evaluate the performance of other pairwise diversity metrics, like the ones proposed by Kuncheva [11]. Their limited (datasets) experimental results however does not guarantee the general utility of their method.

3.2.2 Lazarevic and Obradovic, 2001

Lazarevic and Obradovic [12] use the k-means algorithm to perform the clustering of classifiers. The k-means algorithm is applied to a table of data with as many rows as the classifiers and as many columns as the instances of the training set. The table contains the predictions of each classifier on each instance. Similar to HAC, the k-means algorithm suffers from the problem of selecting the number of clusters (k). The authors solve this problem, by considering iteratively a larger number of clusters until the diversity between them starts to decrease.

Subsequently, the authors prune the classifiers of each cluster using the following approach until the accuracy of the ensemble is decreased. They consider the classifiers in turn from the least accurate to the most accurate. A classifier is kept in the ensemble if its disagreement with the most accurate classifier is more than a predefined threshold and is sufficiently accurate. In addition to simple elimination of classifiers a method for distributing their voting weights is also implemented.

They experimented on four different data sets, using neural network ensembles produced with bagging and boosting. They compare the performance of their pruning method with that of unpruned ensembles and another ad-hoc method that they propose (see other methods) and find that their clustering-based approach offers the highest classification accuracy.

Their method suffers from the fact of parameter setting. How does one set the threshold for pruning models? In addition, the method is not compared to any other pruning methods and sufficient data sets, so its utility cannot be determined. It is very heuristic and ad-hoc.

3.2.3 Fu, Hu and Zhao, 2005

The work of [8] is largely based on the two previous methods. Similarly to [12] it uses the k-means algorithm for clustering the models of an ensemble. Similarly to [9] it prunes each cluster by selecting the single best performing model and uses the accuracy of the pruned ensemble to select the number of clusters.

The difference of this work with the other two clustering-based methods, is merely that the experiments are performed on regression data sets. However, both previous methods could be relatively easily extended to handle the pruning of regression models. The experiments of this work are performed on four data sets using an ensemble of neural networks produced with bagging and boosting, similar to [12].

3.3 Ranking-based

Ranking-based methods order the classifiers in the ensemble *once* according to some evaluation metric and select the classifiers in this order. They differ mainly in the criterion used for ordering the members of the ensemble.

A key concept in *Orientation Ordering* [15] is the *signature vector*. The signature vector of a classifier c is a $|D|$-dimensional vector with elements taking the value +1 if $c(x_i) = y_i$ and -1 if $c(x_i) \neq y_i$. The average signature vector of all classifiers in an ensemble is called the *ensemble signature vector* and is indicative of the ability of the Voting ensemble combination method to correctly classify each of the training examples. The *reference vector* is a vector perpendicular to the ensemble signature vector that corresponds to the projection of the first quadrant diagonal onto the hyper-plane defined by the ensemble signature vector.

In Orientation Ordering the classifiers are ordered by increasing values of the angle between their signature vector and the reference vector. Only the classifiers whose angle is less than $\pi/2$ are included in the final ensemble. Essentially this ordering gives preference to classifiers, which correctly classify those examples that are incorrectly classified by the full ensemble.

3.4 Other

This category includes two approaches that don't belong to any of the previous categories. The first one is based on statistical procedures for directly selecting a subset of classifiers, while the second is based on semi-definite programming.

Tsoumakas et al. [21, 20] prune an ensemble of heterogeneous classifiers using statistical procedures that determine whether the differences in predictive performance among the classifiers of the ensemble are significant. Only the classifiers with significantly better performance than the rest are retained and subsequently combined with the methods of (weighted) voting. The obtained results are better than those of state-of-the-art ensemble methods.

Zhang et al. [24] formulate the ensemble pruning problem as a mathematical problem and apply semi-definite programming (SDP) techniques. In specific, the authors initially formulated the ensemble pruning problem as a quadratic integer programming problem that looks for a fixed-size subset of k classifiers with minimum misclassification and maximum divergence. They subsequently found that this quadratic integer programming problem is similar to the "max cut with size k" problem, which can be approximately solved using an algorithm based on SDP. Their algorithm requires the number of classifiers to retain as a parameter and runs in polynomial time.

4 Greedy Ensemble Selection

Greedy ensemble selection algorithms attempt to find the globally best subset of classifiers by taking local greedy decisions for changing the current subset. An example of the search space for an ensemble of four models is presented in Figure 1.

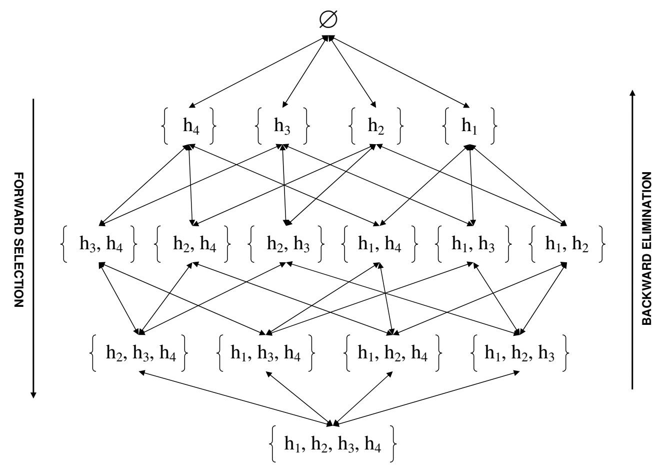

∅
{ h4 } { h3 } { h2 } { h1 }
{ h3, h4 } { h2, h4 } { h2, h3 } { h1, h4 } { h1, h3 } { h1, h2 }
{ h2, h3, h4 } { h1, h3, h4 } { h1, h2, h4 } { h1, h2, h3 }
{ h1, h2, h3, h4 }
FORWARD SELECTION
BACKWARD ELIMINATION

Figure 1. An example of the search space of greedy ensemble selection algorithms for an ensemble of four models.

In the following subsections we present and discuss on what we consider to be the main aspects of greedy ensemble selection algorithms: the direction of search, the measure and dataset used for evaluating the different branches of the search and the size of the final subensemble. The notation that will be used is the following.

- $D = \{(x_i, y_i), i = 1, 2, \ldots, N\}$ is an evaluation set of labelled training examples where each example consists of a feature vector x_i and a class label y_i.
- $H = \{h_t, t = 1, 2, \ldots, T\}$ is the set of classifiers or hypotheses of an ensemble, where each classifier h_t maps an instance x to a class label y, $h_t(x) = y$.
- $S \subseteq H$, is the current subensemble during the search in the space of subensembles.

4.1 Direction of Search

Based on the direction of search, there are two main categories of greedy ensemble selection algorithms: *forward selection* and *backward elimination*.

In forward selection, the current classifier subset S is initialized to the empty set. The algorithm continues by iteratively adding to S the classifier $h_t \in H \backslash S$ that optimizes an evaluation function $f_{FS}(S, h_t, D)$. This function evaluates the addition of classifier h_t in the current subset S based on the labelled data of D. For example, f_{FS} could return the accuracy of the ensemble $S \cup h_t$ on the data set D by combining the decisions of the classifiers with the method of voting. Algorithm 1 shows the pseudocode of the forward selection ensemble selection algorithm. In the past, this approach has been used in [7, 14, 4] and in the Reduce-Error Pruning with Backfitting (REPwB) method in [13].

In backward elimination, the current classifier subset S is initialized to the complete ensemble H and the algorithm continues by iteratively removing from S the classifier $h_t \in S$ that optimizes the evaluation function $f_{BE}(S, h_t, D)$. This function evaluates the removal of classifier h from the current subset S based on the labelled data of D. For example, f_{BE} could return a measure of diversity for the ensemble $S \setminus \{h_t\}$, calculated on the data of D. Algorithm 2 shows the pseudocode of the backward elimination ensemble selection algorithm. In the past, this approach has been used in the *AID thinning* and *concurrency thinning* algorithms [1].

Algorithm 1 The forward selection method in pseudocode

```
Require: Ensemble of classifiers $H$, evaluation function $f_{FS}$, evaluation set D
1: $S = \emptyset$
2: while $S \neq H$ do
3:     $h_t = \arg\max_{h \in H \backslash S} f_{FS}(S, h, D)$
4:     $S = S \cup \{h_t\}$
5: end while
```

Algorithm 2 The backward elimination method in pseudocode

```
Require: Ensemble of classifiers $H$, evaluation function $f_{BE}$, evaluation set D
1: $S = H$
2: while $S \neq \emptyset$ do
3:     $h_t = \arg\max_{h \in S} f_{BE}(S, h, D)$
4:     $S = S \setminus \{h_t\}$
5: end while
```

The time complexity of greedy ensemble selection algorithms for traversing the space of subensembles is $O(t^2 g(T, N))$. The term $g(T, N)$ concerns the complexity of the evaluation function, which is linear with respect to N and ranges from constant to quadratic with respect to T, as we shall see in the following subsections.

4.2 Evaluation Function

One of the main components of greedy ensemble selection algorithms is the function that evaluates the alternative branches during the search in the space of subensembles. Given a subensemble S and a model h_t the evaluation function estimates the utility of inserting (deleting) h_t into (from) S using an appropriate *evaluation measure*, which is calculated on an *evaluation dataset*. Both the measure and the dataset used for evaluation are very important, as their choice affects the quality of the evaluation function and as a result the quality of the selected ensemble.

4.2.1 Evaluation Dataset

One approach is to use the training dataset for evaluation, as in [14]. This approach offers the benefit that plenty of data will be available for evaluation and training, but is susceptible to the danger of overfitting.

Another approach is to withhold a part of the training set for evaluation, as in [4, 1] and in the REPwB method in [13]. This approach is less prone to overfitting, but reduces the amount of data that are available for training and evaluation compared to the previous approach. It sacrifices both the predictive performance of the ensemble's members and the quantity of the evaluation data for the sake of using unseen data in the evaluation. This method should probably be preferred over the previous one, when there is abundance of training data.

An alternative approach that has been used in [3], is based on k-fold cross-validation. For each fold an ensemble is created using the remaining folds as the training set. The same fold is used as the evaluation dataset for models and subensembles of this ensemble. Finally, the evaluations are averaged across all folds. This approach is less prone to overfitting as the evaluation of models is based on data that were not used for their training and at the same time, the complete training dataset is used for evaluation.

During testing the above approach works as follows: the k models that where trained using the same procedure (same algorithm, same subset, etc.) form a cross-validated model. When the cross-validated model makes a prediction for an instance, it averages the predictions of the individuals models. An alternative testing strategy that we suggest for the above approach is to train an additional single model from the complete training set and use this single model during testing.

4.2.2 Evaluation Measure

The evaluation measures can be grouped into two major categories: those that are based on *performance* and those on *diversity*.

The goal of performance-based measures is to find the model that maximizes the performance of the ensemble produced by adding (removing) a model to (from) the current ensemble. Their calculation depends on the method used for ensemble combination, which usually is voting. Accuracy was used as an evaluation measure in [13, 7], while [4] experimented with several metrics, including accuracy, root-mean-squared-error, mean cross-entropy, lift, precision/recall break-even point, precision/recall F-score, average precision and ROC area. Another measure is *benefit* which is based on a cost model and has been used in [7].

The calculation of performance-based metrics requires the decision of the ensemble on all examples of the pruning dataset. Therefore, the complexity of these measures is $O(|S|N)$. However, this complexity can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a classifier is added to/removed from it.

It is generally accepted that an ensemble should contain diverse models in order to achieve high predictive performance. However, there is no clear definition of diversity, neither a single measure to calculate it. In their interesting study, [11], could not reach into a solid conclusion on how to utilize diversity for the production of effective classifier ensembles. In a more recent theoretical and experimental study on diversity measures [19], the authors reached to the conclusion that diversity cannot be explicitly used for guiding the process of greedy ensemble selection. Yet, certain approaches have reported promising results [14, 1].

One issue that worths mentioning here is how to calculate the diversity during the search in the space of ensemble subsets. For simplicity we consider the case of forward selection only. Let S be the current ensemble and $h_t \in H \setminus S$ a candidate classifier to add to the ensemble.

One could compare the diversities of subensembles $S' = S \cup h_t$ for all candidate $h_t \in H \setminus S$ and select the ensemble with the highest diversity. Any pairwise and non-pairwise diversity measure can be used for this purpose. The time complexity of most non-pairwise diversity measures is $O(|S'|N)$, while that of pairwise diversity measures is $O(|S'|^2N)$. However, a straightforward optimization can be performed in the case of pairwise diversity measures. Instead of calculating the sum of the pairwise diversity for every pair of classifiers in each candidate ensemble S', one can simply calculate the sum of the pairwise diversities only for the pairs that include the candidate classifier h_t. The sum of the rest of the pairs is equal for all candidate ensembles. The same optimization can be achieved in backward elimination too. This reduces their time complexity to $O(|S|N)$.

Existing methods [14, 1, 19] use a different approach to calculate diversity during the search. They use pairwise measures to compare the candidate classifier h_t with the current ensemble S, which is viewed as a single classifier that combines the decisions of its members with voting. This way they calculate the diversity between the current ensemble as a whole and the candidate classifier. Such an approach has time complexity $O(|S|N)$, which can be optimized to $O(N)$, if the predictions of the current ensemble are updated incrementally each time a classifier is added to/removed from it. However, these calculations do not take into account the decisions of individual models.

In the past, the widely known diversity measures *disagreement*, *double fault*, *Kohavi-Wolpert variance*, *inter-rater agreement*, *generalized diversity* and *difficulty* were used for greedy ensemble selection in [19]. *Concurrency* [1], *margin distance minimization*, *Complementariness* [14] and *Focused Selection Diversity* are four diversity measures designed specifically for greedy ensemble selection. We next present these measures using a common notation. We can distinguish 4 events concerning the decision of the current ensemble and the candidate classifier:

$$\begin{aligned} e_1 &: \ y = h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \\ e_2 &: \ y \neq h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_3 &: \ y = h_t(\boldsymbol{x_i}) \wedge y = S(\boldsymbol{x_i}) \\ e_4 &: \ y \neq h_t(\boldsymbol{x_i}) \wedge y \neq S(\boldsymbol{x_i}) \end{aligned}$$

The *complementariness* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$COM_D(h_k, S) = \sum_{i=1}^{N} I(e_1),$$

where $I(true) = 1$, $I(false) = 0$ and $S(x_i)$ is the classification of instance x_i from the subensemble S. This classification is derived from the application of an ensemble combination method to S, which usually is voting. The complementariness of a model with respect to a subensemble is actually the number of examples of D that are classified correctly by the model and incorrectly by the subensemble. A selection algorithm that uses the above measure, tries to add (remove) at each step the model that helps the subensemble classify correctly the examples it gets wrong.

The *concurrency* of a model h_k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$CON_D(h_k, S) = \sum_{i=1}^{N} \Big(2 * I(e_1) + I(e_3) - 2 * I(e_4)\Big)$$

This measure is very similar to complementariness with the difference that it takes into account two extra cases.

The *focused ensemble selection* method [17] uses all the events and also takes into account the strength of the current ensemble's decision. Focused ensemble selection is calculated with the following form:

$$FES(h_k, S) = \sum_{i=1}^{N} \Big(NT_i * I(e_1) - NF_i * I(e_2) + \\ +NF_i * I(e_3) - NT_i * I(e_4) \Big),$$

where NT_i denotes the proportion of models in the current ensemble S that classify example $(\boldsymbol{x_i}, y_i)$ correctly, and $NF_i = 1 - NT_i$ denotes the number of models in S that classify it incorrectly.

The *margin distance minimization* method [14] follows a different approach for calculating the diversity. For each classifier h_t an N-dimensional vector, c_t, is defined where each element $c_t(i)$ is equal to 1 if the t^{th} classifier classifies correctly instance i, and -1 otherwise. The vector, C_S of the subensemble S is the average of the individual vectors c_t, $C_S = \frac{1}{|S|}\sum_{t=1}^{|S|} c_t$. When S classifies correctly all the instances the corresponding vector is in the first quadrant of the N-dimensional hyperplane. The objective is to reduce the distance, $d(o, C)$, where d is the Euclidean distance and o a predefined vector placed in the first quadrant. The margin, $MAR_D(h_k, S)$, of a classifier k with respect to a subensemble S and a set of examples $D = (x_i, y_i), i = 1, 2, \ldots, N$ is calculated as follows:

$$MAR_D(h_k, S) = d\left(o, \frac{1}{|S|+1}\Big(c_k + C_S\Big)\right)$$

4.3 Size of Final Ensemble

Another issue that concerns greedy ensemble selection algorithms, is when to stop the search process, or in other words how many models should the final ensemble include.

One solution is to perform the search until all models have been added into (removed from) the ensemble and select the subensemble with the highest accuracy on the evaluation set. This approach has been used in [4]. Others prefer to select a predefined number of models, expressed as a percentage of the original ensemble [13, 7, 14, 1].

5 Conclusions

This works was a first attempt towards a taxonomy of ensemble selection methods. We believe that such a taxonomy is necessary for researchers working on new methods. It will help them identify the main categories of methods and their key points, and avoid duplication of work. Due to the large amount of existing methods and the different parameters of an ensemble selection framework (heterogeneous/homogeneous ensemble, algorithms used, size of ensemble, etc), it is possible to devise a new method, which may only differ in small, perhaps unimportant, details from existing methods. A generalized view of the methods, as offered from a taxonomy, will help avoid work towards such small differences, and perhaps may lead to more novel methods.

Of course, we do not argue that the proposed taxonomy is perfect. On the contrary, it is just a first and limited step in abstracting and categorizing the different methods. Much more elaborate study has to be made, to properly account for the different aspects of existing methods. No doubt, some high quality methods may have been left outside this study. We hope that through a discussion and the criticism of this work within the ensemble methods community, and especially people working on ensemble selection, a much improved version of it will arise.

REFERENCES

[1] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, 'Ensemble diversity measures and their application to thinning.', *Information Fusion*, **6**(1), 49–62, (2005).

[2] L. Breiman, 'Bagging Predictors', *Machine Learning*, **24**(2), 123–40, (1996).

[3] R. Caruana, A. Munson, and A. Niculescu-Mizil, 'Getting the most out of ensemble selection', in *Sixth International Conference in Data Mining (ICDM '06)*, (2006).

[4] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, 'Ensemble selection from libraries of models', in *Proceedings of the 21st International Conference on Machine Learning*, p. 18, (2004).

[5] T. G. Dietterich, 'Machine-learning research: Four current directions', *AI Magazine*, **18**(4), 97–136, (1997).

[6] T. G. Dietterich, 'Ensemble Methods in Machine Learning', in *Proceedings of the 1st International Workshop in Multiple Classifier Systems*, pp. 1–15, (2000).

[7] W. Fan, F. Chu, H. Wang, and P. S. Yu, 'Pruning and dynamic scheduling of cost-sensitive ensembles', in *Eighteenth national conference on Artificial intelligence*, pp. 146–151. American Association for Artificial Intelligence, (2002).

[8] Qiang Fu, Shang-Xu Hu, and Sheng-Ying Zhao, 'Clusterin-based selective neural network ensemble', *Journal of Zhejiang University SCIENCE*, **6A**(5), 387–392, (2005).

[9] Giorgio Giacinto, Fabio Roli, and Giorgio Fumera, 'Design of effective multiple classifier systems by clustering of classifiers', in *15th International Conference on Pattern Recognition, ICPR 2000*, pp. 160–163, (3–8 September 2000).

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, 'Adaptive mixtures of local experts', *Neural Computation*, **3**, 79–87, (1991).

[11] L.I. Kuncheva and C.J. Whitaker, 'Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy', *Machine Learning*, **51**, 181–207, (2003).

[12] Aleksandar Lazarevic and Zoran Obradovic, 'Effective pruning of neural network classifiers', in *2001 IEEE/INNS International Conference on Neural Networks, IJCNN 2001*, pp. 796–801, (15–19 July 2001).

[13] D. Margineantu and T. Dietterich, 'Pruning adaptive boosting', in *Proceedings of the 14th International Conference on Machine Learning*, pp. 211–218, (1997).

[14] G. Martinez-Munoz and A. Suarez, 'Aggregation ordering in bagging', in *International Conference on Artificial Intelligence and Applications (IASTED)*, pp. 258–263. Acta Press, (2004).

[15] G. Martinez-Munoz and A. Suarez, 'Pruning in ordered bagging ensembles', in *23rd International Conference in Machine Learning (ICML-2006)*, pp. 609–616. ACM Press, (2006).

[16] I. Partalas, G. Tsoumakas, I. Katakis, and I. Vlahavas, 'Ensemble pruning via reinforcement learning', in *4th Hellenic Conference on Artificial Intelligence (SETN 2006)*, pp. 301–310, (May 18–20 2006).

[17] I. Partalas, G. Tsoumakas, and I. Vlahavas, 'Focused ensemble selection: A diversity-based method for greedy ensemble selection', in *18th European Conference on Artificial Intelligence*, (2008).

[18] Robert E. Schapire, 'The strength of weak learnability', *Machine Learning*, **5**, 197–227, (1990).

[19] E. K. Tang, P. N. Suganthan, and X. Yao, 'An analysis of diversity measures', *Machine Learning*, **65**(1), 247–271, (2006).

[20] G. Tsoumakas, L. Angelis, and I. Vlahavas, 'Selective fusion of heterogeneous classifiers', *Intelligent Data Analysis*, **9**(6), 511–525, (2005).

[21] G. Tsoumakas, I. Katakis, and I. Vlahavas, 'Effective Voting of Heterogeneous Classifiers', in *Proceedings of the 15th European Conference on Machine Learning, ECML2004*, pp. 465–476, (2004).

[22] C.J. Watkins and P. Dayan, 'Q-learning', *Machine Learning*, **8**, 279–292, (1992).

[23] D. Wolpert, 'Stacked generalization', *Neural Networks*, **5**, 241–259, (1992).

[24] Yi Zhang, Samuel Burer, and W. Nick Street, 'Ensemble pruning via semi-definite programming', *Journal of Machine Learning Research*, **7**, 1315–1338, (2006).

[25] Zhi-Hua Zhou and Wei Tang, 'Selective ensemble of decision trees', in *9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, RSFDGrC 2003*, pp. 476–483, Chongqing, China, (May 2003).

Penta-Training: Clustering Ensembles with Bootstrapping of Constraints

Carlotta Domeniconi[1] and **Muna Al-Razgan**[1]

Abstract.
In this paper we combine clustering ensembles and semi-supervised clustering to address the ill-posed nature of clustering. We introduce a mechanism which leverages the ensemble framework to bootstrap informative constraints directly from the data and from the various clusterings, without intervention from the user. Our approach is well suited for problems where the information available from an external source is very limited. We demonstrate the effectiveness of our proposed technique with experiments using real datasets and other state-of-the-art semi-supervised techniques.

1 Introduction

Clustering is the process of discovering homogeneous groups of data according to a given similarity measure. Clustering is well suited for data analysis. However, clustering is susceptible to several difficulties. It is well known that off-the-shelf clustering methods may discover very different structures in a given set of data. This is because each clustering algorithm has its own bias resulting from the optimization of different criteria. Furthermore, there is no ground truth against which the clustering result can be validated. Thus, no cross-validation technique can be carried out to tune input parameters involved in the clustering process.

Recently, cluster ensembles have emerged as a technique for overcoming problems with clustering algorithms. A cluster ensemble consists of different partitions. These partitions can be obtained from multiple applications of any single algorithm with different initializations, from various bootstrapped samples of the available data, or from the application of different algorithms to the same dataset. Cluster ensembles offer a solution to challenges inherent to clustering arising from its ill-posed nature: they can provide more robust and stable solutions by making use of the consensus across multiple clustering results, while averaging out emergent spurious structures that arise due to the various biases to which each participating algorithm is tuned.

To address the ill-posed nature of clustering, semi-supervised clustering has also emerged. Semi-supervised clustering uses prior knowledge to guide the clustering process, thus providing results that adhere to the user's preference. Prior knowledge is often expressed in terms of pairwise constraints (must-link or cannot-link) on the data. Semi-supervised clustering has its own challenges. Often, the number of constraints available is very limited. Furthermore, the given constraints may not be effective for the improvement of clustering results. Active learning strategies to identify informative constraints have been developed. These techniques select pairs of points which maximize some measure of "relevance". The user (or oracle) is then queried on the nature of the relationship existing between these points.

In this work, we combine clustering ensembles with semi-supervised clustering. We use the ensemble framework to bootstrap informative constraints directly from the data, and from the different clustering components. Our approach is well suited for problems where the information available from an external source (e.g., domain expert) is very limited. We also demonstrate the feasibility of our technique to situations where prior knowledge is absent. Our work is motivated by co-training [5] and tri-training [13]. As co-training and tri-training, we leverage the ensemble methodology to perform semi-supervised learning. While co-training and tri-training use classifiers as learning components, and propagate labels among them, our technique uses a collection of clusterings (five, hence the name *Penta-Training*) to derive constraints. To the best of our knowledge, this is the first attempt of its kind.

We design a constrained version of subspace clustering, and use it as the basic component of our penta-training framework. To discover subspace clusters, we use a Locally Adaptive Clustering (or LAC) algorithm which has been proven to be effective [8, 7]. LAC is an iterative algorithm that assigns weights to features according to the local variance of data along each dimension. Dimensions along which data are loosely clustered receive a small weight, which has the effect of elongating distances along that dimension. Features along which data manifest a small variance receive a large weight, which has the effect of constricting distances along that dimension. Thus, the learned weights perform a directional local reshaping of distances which allows a better separation of clusters, and therefore the discovery of different patterns in different subspaces of the original input space. LAC depends on an input parameter (called h) that controls the strength of the incentive to cluster on more features. Diverse clusterings can be generated using different h values.

We modify the dynamic of the LAC algorithm to embed the given constraints in the cluster assignment and weight computation processes. We call the resulting algorithm CLAC (Constrained-LAC). The individual clusterings are iteratively refined using constraints generated during the penta-training process. In particular, in each iteration of penta-training, constraints are generated for a clustering if the other four clusterings agree on them. The process is repeated until convergence, i.e., until no change in all five clusterings is observed.

2 Related Work

Our approach is related to co-training [5] and tri-training [13]. Co-training [5] assumes that the given features can be split into two sets, each of which is sufficient to train a classifier that will produce accu-

[1] Department of Computer Science, George Mason University, USA, email: carlotta@cs.gmu.edu, malrazga@gmu.edu

rate results. Each resulting classifier makes predictions on unlabeled data, and then provides new training data (those labeled with highest confidence) to its counterpart, for iterating rounds of training. Although this method broke ground in first designing collaborative classifiers to perform semi-supervised learning, its requirement for two sufficient feature sets severely limits its applicability.

Tri-training [13] does not assume redundant feature sets. Diverse classifiers (three) are generated via bootstrap sampling of the original labeled data. An unlabeled example is labeled for a classifier if the other two classifiers agree on the labeling, under certain conditions. Both co-training and tri-training require initial labeled data to train the component classifiers.

Recently, several semi-supervised clustering algorithms have been proposed. The authors in [12] propose the COP-Kmeans algorithm as a variation of k-means, where constraints are embedded during the clustering process: each point is assigned to the closest cluster, which will enact the least violation of constraints. The algorithm will not assign the point if no such cluster can be found. Two additional constraint-based variants of k-means are Seeded-KMeans and Constrained-KMeans [4]. In both algorithms, the given labeled data are used to initialize a seeded set; the constraints obtained from this labeled set are then used to guide the k-means algorithm. Seeded-KMeans allows its constraints to be violated in successive iterations, while Constrained-KMeans enforces the constraints in each iteration.

Most semi-supervised learning approaches have focused on developing new algorithms. Only recently, more attention has been given to the nature of the available knowledge, and strategies to active learn relevant side-information have been developed. In [9], constraints are imputed from the information provided by the co-association values between pairs of points in a clustering ensemble. In [6], the authors define two measures to characterize the informativeness and coherence of a set of constraints. All these methods require the existence of a domain expert. Our approach, instead, leverages the ensemble methodology to derive constraints which are completely data-driven.

3 Penta-Training

When building ensembles, we are faced with a difficult dilemma: accuracy versus diversity. We are in need of diverse, and yet accurate, components. In our work, to construct effective clustering ensembles, we rely on the sensitivity of the underlying subspace clustering algorithm (LAC) on its input parameter. Thus, we run the LAC algorithm multiple times with different input parameter values. (In our experiments, we observed that five components provide a sufficient range of values for the input parameter.) Furthermore, our objective is to improve the quality of the individual clusterings using a collaborative approach among the components. To achieve this goal, we need to ensure that the information shared across the components is accurate and useful. To this end, we adopt the following heuristic: we look for the pairs of points on which four (out of the five) clusterings agree, i.e., all four clusterings group the two points together, or separately. In the first case, a must-link constraint is generated for the fifth component; in the second case, a cannot-link constraint is generated. The process is iterated until no further constraints can be generated. The requirement for an agreement across four components ensures a high level of accuracy of the derived constraints. To ensure constraint relevance, we introduce a ranking mechanism among the generated constraints, and distribute only the top ranked ones to the fifth component.

The following Sections describe the details of our approach. In order to embed constraints into subspace clustering, we organize the constraints into a graph called *Chunklet Graph*. In the following, we describe the procedure to construct such graph.

3.1 Chunklet Graph

We assume that two sets of constraints are given: M is the set of must-link constraints, and C is the set of cannot-link constraints. Such constraints are either provided by a domain expert, or they are bootstrapped from the data (as explained in Section 3.4).

A chunklet is a group of points that belong to the same cluster, although the identity of the cluster is unknown [3]. The size of the chunklet is equal to the number of points it contains, e.g., the chunklet $\triangle = (\mathbf{x}_1, \mathbf{x}_2)$ has size $|\triangle| =$2.

Each chunklet is formed by must-link constraints. For example, if a must-link between points $\mathbf{x}_1$ and $\mathbf{x}_2$ exist, then the chunklet $\triangle_1 = (\mathbf{x}_1, \mathbf{x}_2)$ is formed. Following the formation of chunklets through must-link constraints, a transitive closure process, in which chunklets are merged, is initiated. For example, if there is a must-link constraint between $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{x}_1, \mathbf{x}_3)$, then by transitive closure the chunklet $\triangle_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is formed.

Once chunklets are formed and all transitive closures completed, a graph is created using the cannot-link constraints. These constraints prevent the assignment of some chunklets to the same cluster. If, for example, there is a cannot-link constraint between the pair $(\mathbf{x}_3, \mathbf{x}_5)$ and we have the chunklet $\triangle_3 = (\mathbf{x}_4, \mathbf{x}_5)$, then our previously cited chunklet $\triangle_2 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ will be prevented from the assignment to the same cluster as chunklet $\triangle_3$.

A cannot-link constraint is represented in the graph as an edge between two vertices, where each vertex corresponds to one chunklet. In this way the entire chunklet graph $G_{ch} = (V, E)$ is constructed, where V is a set of vertices (or chunklets) constructed from the must-link constraints, and $|V|$ is the total number of chunklets. E is the set of edges, and an edge E_{ij} exists between vertices (chunklets) v_i and v_j iff there exist $\mathbf{x}_i \in v_i, \mathbf{x}_j \in v_j$ such that $(\mathbf{x}_i, \mathbf{x}_j) \in C$.

3.2 Chunklet Initialization

Similar to k-means, the subspace clustering algorithm LAC depends on the initial choice of centroids. Thus, we make use of the given constraints to achieve a good initialization.

Once we have constructed the chunklet graph, we select the vertices (or chunklets) with cannot-link constraints between them, i.e., the vertices v_i and v_j such that $E_{ij} = 1$. We then compute the mean vectors of the points contained in each corresponding chunklet. These mean vectors become the initial centroids. If the number of selected chunklets is less than k (the number of desired clusters), we choose as additional initial centroids the points that are the farthest from the already chosen ones. The selection is iterated until we reach k initial centroids.

3.3 Constrained Locally Adaptive Clustering (CLAC)

The chunklet initialization procedure provides k initial centroids. We use the subspace clustering algorithm LAC to generate the clustering components of our ensemble. A full derivation of the LAC algorithm is given in [7]. Here it suffices to know that LAC is an iterative algorithm that assigns local weights to features according to the local variance of data along each dimension. Thus, at each iteration, LAC provides a set of k centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, and a set of k weight vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$, where each weight vector reflects the relevance of features within the corresponding cluster.

We modify the dynamic of the LAC algorithm to embed the given constraints into the cluster assignment process, which in turn affects the computation of weights. We call the resulting algorithm CLAC (Constrained LAC). We first present the cluster assignment strategy for points in the chunklet graph.

Chunklet assignment is the process of assigning vertices (chunklets) in the graph to the appropriate centroid without violating any of the must-link or cannot-link constraints. We consider each chunklet as a group of points, and assign all points in the chunklet to the closest centroid which does not violate any constraint. Given a vertex (chunklet) v_i, we calculate the weighted Euclidean distances between all the points $\mathbf{x}_j \in v_i$ and each centroid $\mathbf{c}_l$, where $l = 1, \ldots, k$, and look for the centroid $\mathbf{c}_t$ that satisfies $\mathbf{c}_t = \arg\min_l(d(v_i, \mathbf{c}_l))$, where $d(v_i, \mathbf{c}_l) = \sum_{j=1}^{|v_i|} \sqrt{\sum_{s=1}^{D} w_{ls}(x_{js} - c_{ls})^2}$, D is the dimensionality of the data, and $\mathbf{w}_l$ is the weight vector associated with centroid $\mathbf{c}_l$. To assign a chunklet to a centroid, we need to consider possible cannot-link constraints involving the chunklet. Three cases which require different centroid assignment strategies are given below.

Case 1 v_i is an isolated chunklet that does not have an edge in the graph G_{ch}. We assign this chunklet to the centroid $\mathbf{c}_t = \arg\min_l(d(v_i, \mathbf{c}_l))$.

Case 2 v_i is a chunklet that has at least one neighbor in the graph G_{ch}, and none of its neighbors has been assigned to a centroid. As before, we assign v_i to the centroid $\mathbf{c}_t = \arg\min_l(d(v_i, \mathbf{c}_l))$.

Case 3 v_i ia a chunklet that has at least one neighbor in the graph G_{ch} that has been assigned to a centroid. We construct the set of centroids R_i to which the neighboring nodes of v_i have been assigned. We then assign v_i to the centroid $\mathbf{c}_t = \arg\min_l(d(v_i, \mathbf{c}_l))$ such that $\mathbf{c}_t \notin R_i$.

This procedure assigns chunklets to centroids according to the local similarities being discovered by the subspace clustering algorithm.

The overall CLAC algorithm is summarized in Algorithm 1. CLAC computes a partition of the data that satisfies the given constraints (under the assumption that the satisfaction of all constraints is feasible). The inputs of the algorithm are the number of desired clusters k, the value of the h parameter (which controls the strength of the incentive to cluster on more features [7]), and the sets of constraints M and C. Centroids are initialized according to the procedure described in Section 3.2. Equal weights are assigned initially to all features. The chunklet assignment procedure is used to compute the first partition. Points which are not contained in any chunklet are assigned to the closest centroid (according to the weighted Euclidean distance). The weights are updated, according to an exponential scheme which assigns larger weights to features where points have low variance (the parameter h is used here). Thus, a new partition and new centroids are computed. The procedure is iterated until convergence, i.e., until clusters do not change.

Algorithm 1 CLAC Algorithm

```
Input: n points $\mathbf{x} \in \Re^D$, number of clusters $k$, $h$, set $M$ of must-link constraints, set $C$ of cannot-link constraints.

1. $S_1 = \emptyset, \ldots, S_k = \emptyset$
2. Let $(G_{ch})$ be the chunklet graph constructed from $M$ and $C$;
   Chunklet Initialization
   $\forall v_i \in G_{ch}$ such that $E_{ij} = 1$ for some $j$
       Compute the mean of points in $v_i$ and assign it as initial centroid;
   Set $z$ = number of selected centroids;
   while $(z < k)$
       Select the farthest point from the already selected centroids, and assign it as initial centroid;
   Let $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ be the resulting centroids;
3. Set $w_{sj} = 1/D$, for each centroid $\mathbf{c}_j$, $j = 1, ..., k$ and each feature $s = 1, ..., D$;
4. For each $v_i \in G_{ch}$, let $t_i$ be the assigned cluster centroid (as determined by the chunklet assignment procedure)
     $\forall \mathbf{x} \in v_i$, $S_{t_i} = S_{t_i} \bigcup \{\mathbf{x}\}$;
5. For each centroids $\mathbf{c}_j$, and for each point $\mathbf{x} \notin v_i, \forall_i$
       $S_t = S_t \bigcup \{\mathbf{x} | t = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$
       where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^{D} w_{ls}(c_{ls} - x_s)^2)^{1/2}$;
6. Compute new weights
   For each centroid $c_j$, and for each feature $s$:
   Set $X_{js} = \sum_{x \in S_j} (c_{js} - x_s)^2 / |S_j|$;
   Set $w_{js} = \frac{exp(-X_{js}/h)}{\sum_{s=1}^{D} exp(-X_{js}/h)}$;
7. For each $v_i \in G_{ch}$, let $t_i$ be the assigned cluster centroid (as determined by the chunklet assignment procedure)
     $\forall \mathbf{x} \in v_i$, $S_{t_i} = S_{t_i} \bigcup \{\mathbf{x}\}$
8. For each centroids $\mathbf{c}_j$, and for each point $(\mathbf{x} \notin v_i, \forall_i)$
       $S_t = S_t \bigcup \{\mathbf{x} | t = argmin_l L_w(\mathbf{c}_l, \mathbf{x})\}$
       where $L_w(\mathbf{c}_l, \mathbf{x}) = (\sum_{s=1}^{D} w_{ls}(c_{ls} - x_s)^2)^{1/2}$;
9. Compute new centroids
   Set $c_j = \frac{\sum_x x 1_{S_j}(x)}{\sum_x 1_{S_j}(x)}$, for each $j = 1, ..., k$, where $1_S(.)$ is the indicator function of set $S$;
10. Iterate 5-10 until convergence (no change in cluster assignment).

Output: The final partition $S = \{S_1, \ldots, S_k\}$, $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$, $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$
```

3.4 The Penta-training Algorithm

Penta-Training assembles multiple (five) clusterings obtained by CLAC, and bootstraps constraints to improve the quality of the components, and ultimately of the ensemble.

Given an initial collection of constraints (M, C), we run CLAC five times with different values of the h parameter. Each run of CLAC is provided with the entire data set and the entire constraint set. We obtain five clusterings of the data. Penta-training leverages the consensus achieved across such partitions to bootstrap and propagate constraints: we look for pairs of points on which four (out of the five) clusterings agree (and the fifth disagrees), i.e., all four clusterings group the two points together, or separately. In the first case, a must-link constraint is generated for the fifth component; in the second case, a cannot-link constraint is generated. Once all constraints for a given component have been generated, they are added to the current set of constraints of that component, and CLAC is re-run. The process is iterated for all combinations of four components, until no change in all five clusterings is observed. To ensure that only relevant constraints are propagated, we rank the candidate constraints, and use only the top ranked ones. In particular, for each candidate must-link constraint $(\mathbf{x}_n, \mathbf{x}_m)$, we compute the four weighted Euclidean distances, using the corresponding weights of the clusters the two points

$\mathbf{x}_n$ and $\mathbf{x}_m$ are assigned to, and compute their average. The average distances are then sorted in ascending order. We select the top ranked pairs (with smallest distances) as must-link constraints for the fifth component. We proceed similarly for the cannot-link contraints. For each candidate cannot-link constraint we compute their Euclidean distance (note that in this case, four clusterings place the points in different clusters, and therefore there is no single weight vector associated with them). We then sort the distances in descending order. We select the top ranked pairs (with largest distances) as cannot-link constraints for the fifth component. In our experiments, at each iteration of penta-training, we select the top five ranked must-link and the top five ranked cannot-link constraints.

We observe that penta-training can also be applied when no constraints are initially available. In this case, we start building the ensemble by simply running the original LAC algorithm (with no side-information) using different values of h. As constraints are bootstrapped during the rounds of penta-training, LAC is substituted by CLAC. We test this scenario as well in our experiments.

At convergence, we have available five partitions (precisely, each partition corresponds to k centroids, and corresponding weight vectors). We map the problem of finding a consensus function to a graph partitioning problem, by applying the WBPA (Weighted Bipartite Partitioning) algorithm [1], which has been demonstrated to be effective. The WBPA algorithm takes into account not only how often points are grouped together across the clusterings, but also the degree of confidence of the groupings (by means of the weights). The Penta-Training algorithm is summarized in Algorithm 2.

Algorithm 2 Penta-Training Algorithm

```
Input: $n$ points $\mathbf{x} \in \Re^D$, number of clusters $k$, $h$, set $M$ of must-link constraints, set $C$ of cannot-link constraints.
Let $(G_{ch})$ be the chunklet graph constructed from $M$ and $C$;
$S^{(h_i)}$= CLAC($\{\mathbf{x}\}$, $k$, $h_i$, $M$, $C$), for $i = 1, \ldots, 5$;
Let $T_n^{(h_i)} \in S^{(h_i)}$, be the set in partition $S^{(h_i)}$ to which $\mathbf{x}_n$ is assigned;
[Initialization of constraints for each component]
for $i = 1, \ldots, 5$
    $M^{(h_i)} = M, C^{(h_i)} = C$;
repeat
    for $l = 1, \ldots, 5$
    [Bootstrapping of must-link constraints]
        for every pair $(\mathbf{x}_n, \mathbf{x}_m) \notin M$
            if ( $(T_n^{(h_l)} \neq T_m^{(h_l)})$ and $(\forall i \neq l, T_n^{(h_i)} = T_m^{(h_i)})$)
                Calculate the average weighted distance:
                $d(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{4} \sum_{i \neq l} (\sum_{s=1}^{D} w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$;
                [$\mathbf{w_t}$ is the weight vector of the cluster the points $\mathbf{x}_n$ and $\mathbf{x}_m$ are assigned to]
            end if
        end for
        Sort above distances in ascending order;
        Select a percentage of top ranked pairs (with smallest distances), and add them to $M_{h_l}$;
    [Bootstrapping of cannot-link constraints]
        for every pair $(\mathbf{x}_n, \mathbf{x}_m) \notin C$
            if ( $(T_n^{(h_l)} = T_m^{(h_l)})$ and $(\forall i \neq l, T_n^{(h_i)} \neq T_m^{(h_i)})$)
                Calculate the Euclidean distance:
                $d(\mathbf{x}_n, \mathbf{x}_m) = \sum_{s=1}^{D} w_{ts}(x_{n_s} - x_{m_s})^2)^{1/2}$;
            end if
        end for
        Sort above distances in descending order;
        Select a percentage of top ranked pairs (with largest distances), and add them to $C_{h_l}$;
        Run CLAC($\{\mathbf{x}\}$, $k$, $h_l$, $M_{h_l}$, $C_{h_l}$);
    end for
until convergence [all five clusterings do not change]
Input the obtained five partitions (and corresponding weights) to WBPA [1];
Output: Partition of the $n$ data points into $k$ clusters.
```

4 Empirical Evaluation

4.1 Experimental Design

Table 1. Characteristics of the datasets

Dataset	k	D	n (points-per-class)
Iris	3	4	150 (50-50-50)
WDBC	2	31	424 (212-212)
Breast	2	9	478 (239-239)
Wine	3	13	144 (48-48-48)
Ionosphere	2	33	239 (126-113)

In our experiments, we used five real datasets. The characteristics of all datasets are given in Table 1. Iris, Breast, Wine and Ionosphere are from the UCI Machine Learning Repository [2]. WDBC is the Wisconsin Diagnostic Breast Cancer dataset [11].

The clustering ensemble algorithm WBPA uses METIS [10] to compute the k-way partitioning of a graph. Since METIS requires balanced datasets, we performed random sampling on Breast, WDBC, Wine, and Ionosphere. In each case, we sub-sampled the most populated class: from 357 to 212 for WDBC, from 444 to 239 for Breast, from 59 to 48 and 71 to 48 for Wine, and from 225 to 113 for Ionosphere.

We tested our penta-training framework with two scenarios: in one case, a limited number of constraints is initially available; in the second case, no constraints are available. For the first scenario, to generate the initial set of constraints, we follow the procedure introduced in [9]. We run the LAC algorithm 10 times, for $h = 1, \ldots, 10$. We then build the co-association matrix resulting from the 10 partitions. We select all pairs of points $(\mathbf{x}_n, \mathbf{x}_m)$ with a co-association value in the interval $[0.45, 0.55]$. This means that, roughly, half of the components clusters the two points together, and the other half places them in separate groups. Thus, it is unclear whether the two points should be clustered together or not. Therefore, the user feedback is most valuable for such points. In our experiments, we use the ground truth (class labels) to generate constraints for the selected points. For each dataset, the number of constraints generated is equal to 20% the number of data available. (If a larger number of pairs have a co-association value in the range $[0.45, 0.55]$, we random sample a subset.) The obtained constraints are given in input to all five CLAC components. (The same set of constraints is also given in input to the competitive techniques.) Each run of CLAC uses a different values of the h parameter. In our experiments we use the values $\{1, 3, 5, 7, 10\}$. According to our experience, this range of values provides in general diverse and accurate components. In the second case, when no initial constraints are available, we build the ensemble by running the LAC algorithm with the five values of h. As constraints are bootstrapped during the iterations of penta-training, LAC is substituted by CLAC.

At each iteration of penta-training, for each component, we bootstrap the top five ranked must-link constraints, and the top five ranked cannot-link constraints. We compare the following techniques:

Table 2. Error Rates and Standard Deviations

Methods	Iris	Breast	WDBC	Ionosphere	Wine
LAC	14.13±2.18	15.9±11.78	19.76±15.75	34.06±4.20	15.16±11.15
CLAC (20%)	13.06±1.74	16.32±11.41	16.69±8.92	34.33±6.88	15.69±11.59
COP-Kmeans (20%)	11.73±0.78	4.68±0.17	48.34± 0.5	29.87±9.17	56.46±2.17
Seeded-COP-Kmeans (20%)	**9.33**	4.39	48.34	**25.52**	55.74
Penta-Training (20%)	10.67	3.56	9.19	32.21	11.11
Penta-Training (w/o const.)	14	**3.14**	**8.73**	31.38	**9.03**

- **LAC** [7]. We run the LAC algorithm five times for $h \in \{1, 3, 5, 7, 10\}$, and report average error rates and standard deviations.
- **CLAC**. We run the CLAC algorithm five times for $h \in \{1, 3, 5, 7, 10\}$, and report average error rates and standard deviations. We provide CLAC the set of constraints generated according to the procedure described above.
- **COP-Kmeans** [12]. We run COP-Kmeans 10 times using random initialization of centroids, and report average error rates and standard deviations. Again, we provide COP-Kmeans the same set of constraints generated according to the procedure described above.
- **Seeded-COP-Kmeans** [4]. In this case centroids are initialized using the given constraints, according to the technique described in Section 3.2.
- **Penta-Training** *with initial constraints*. We start penta-training with the same initial set of constraints we feed all competitive semi-supervised techniques.
- **Penta-Training** *without initial constraints*. In this case, no external constraints are used. The ensemble starts off in an unsupervised mode (LAC components), and constraints are bootstrapped in successive iterations from the data.

4.2 Analysis of the Results

Error rates and standard deviations are reported in Table 2. In four problems, penta-training provided the lowest error rate, or an error rate very close to the minimum. For Ionosphere, Seeded-COP-Kmeans gives the lowest error. In some cases, penta-training provides huge improvements with respect to LAC, CLAC, and COP-Kmeans. This indicates that the collaborative approach adopted by penta-training allows the bootstrapping of accurate and relevant constraints for the clustering process. In particular, as expected, the largest improvements are achieved when LAC and CLAC have large standard deviations (i.e., on Breast, WDBC, and Wine). In these cases, the components are diverse, and the ensembles become most effective. Quite interesting is the fact that penta-training without initial constraints in most cases performs better than penta-training with initial constraints. This shows the efficacy of our that data-driven and ensemble-driven constraints. COP-Kmeans and Seeded-COP-Kmeans perform very poorly on WDBC and Wine. These data are sparse and have larger dimensionalities; in such conditions, COP-Kmeans (as k-means) has difficulties in finding the underlying cluster structure.

5 Conclusions and Future Work

We have introduced a semi-supervised framework for clustering ensembles which addresses the ill-posed nature of clustering. We use the ensemble framework to bootstrap informative constraints directly from the data, and from the different clustering components. Our approach is well suited for problems where the information available from an external source is very limited, or not available at all. Furthermore, since our approach builds upon subspace clustering components, it is well suited for high dimensional data.

Several avenues can be taken for future work. As the iterations of penta-training progress, the clustering components may become correlated due to the distribution of constraints across the ensemble. As such, the best ensemble accuracy may be achieved before convergence. A criterion for an early stop of penta-training will be considered. Furthermore, a larger number of components can be used, and alternative majority schemes for the bootstrapping of constraints will be investigated. The use of soft constraints that reflect the uncertainty associated with prior knowledge will also be considered.

ACKNOWLEDGEMENTS

This work was in part supported by NSF CAREER Award IIS-0447814.

REFERENCES

[1] M. Al-Razgan and C. Domeniconi, ‘Weighted clustering ensembles’, in *SIAM International Conference on Data Mining*, pp. 258–269, (2006).

[2] A. Asuncion and D. J. Newman. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2007.

[3] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall., ‘Learning distance functions using equivalence relations’, in *International Conference on Machine Learning*, (2003).

[4] S. Basu, A. Banerjee, and R. Mooney, ‘Semi-supervised clustering by seeding’, in *International Conference on Machine Learning*, (2002).

[5] A. Blum and T. Mitchell, ‘Combining labeled and unlabeled data with co-training’, in *Conference on Computational Learning Theory*, pp. 92–100, (1998).

[6] I. Davidson, K. Wagstaff, and S. Basu, ‘Measuring constraint-set utility for partitional clustering algorithms’, in *European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 115–126, (2006).

[7] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos, ‘Locally adaptive metrics for clustering high dimensional data’, *Data Mining and Knowledge Discovery Jouranal*, **14**(1), 63–97, (2007).

[8] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma, ‘Subspace clustering of high dimensional data.’, in *SIAM International Conference on Data Mining*, pp. 517–520, (2004).

[9] D. Greene and P. Cunningham, ‘An ensemble approach to identifying informative constraints for semi-supervised clustering’, in *European Conference on Machine Learning*, pp. 140–151, (2007).

[10] G. Karypis and V. Kumar, ‘A fast and high quality multilevel scheme for partitioning irregular graphs’, *SIAM Journal on Scientific Computing*, **20**(1), 359–392, (1998).

[11] O. Mangasarian and W. Wolberg, ‘Cancer diagnosis via linear programming’, *SIAM News*, **23**(5), 1–18, (1990).

[12] K. Wagstaff, C. Cardie, S. Rogers, and S. Schröedl, ‘Constrained k-means clustering with background knowledge’, in *International Conference on Machine Learning*, (2001).

[13] Z. Zhou and M. Li, ‘Tri-training: exploiting unlabeled data using three classifiers’, *IEEE Transactions on Knowledge and Data Engineering*, **17**(11), 1529–1541, (2005).

Independent Model Selection for Ensemble Dispersion Forecasting

A. Ciaramella [1], **G. Giunta** [1], **A. Riccio** [1] and **S. Galmarini** [2]

Abstract. This work aims at introducing an approach to analyze the independence between different models in a multi-model ensemble context. The models are operational long-range transport and dispersion models, also used for the real-time simulation of pollutant dispersion or the accidental release of radioactive nuclides in the atmosphere. In order to compare models, an approach based on the hierarchical agglomeration of distributions of predicted radionuclide concentrations is proposed. We use two different similarity measures: Negentropy information and Kullback-Leibler divergence. These approaches are used to analyze the data obtained during the ETEX-1 exercise.

1 Introduction

Standard meteorological/air quality practice, such as the prediction of the future state of the atmosphere, typically proceeds conditionally on one assumed model. The model is the result of the work of many area-expert scientists, e.g. meteorologists, computational scientists, statisticians, and others.

Nowadays, several models are available for the forecast of variables of meteorological and/or air quality interest, but, even when using the same ancillary (e.g. initial and boundary) data, they could give different answers to the scientific question at hand. This is a source of uncertainty in drawing conclusions, and the typical approach, that is of conditioning on a single model deemed to be “the best”, ignores this source of uncertainty and underestimates the possible effects of a false forecast.

Ensemble prediction aims at reducing this uncertainty by means of techniques designed to strategically sample the forecast pdf, e.g. the breeding of growing modes [17] or singular vectors [15] in the weather forecasting field. Recently, the ensemble approach has been extended to multi-model prediction, too [13].

The multimodel approach has been successfully applied to atmospheric dispersion predictions [8, 9, 10] where the uncertainty of weather forecast sums and mixes with that stemming from the description of the dispersion process. The methodology relies on the analysis of the forecasts of several models used operationally by national meteorological services and environmental protection agencies worldwide to forecast the evolution of accidental releases of harmful materials. The objectives are clear: after the release of hazardous material into the atmosphere, it is extremely important to support the decision-making process with any relevant information and to provide a comprehensive analysis of the uncertainties and the confidence that can be put into the the dispersion forecast. Galmarini et al. [9] showed how the intrinsic differences among the models can become a useful asset to be exploited for the sake of a more educated support to decision making by means of the definition of ad-hoc parameters and treatments of model predictions. Among others, they proposed the so called ‘Median Model’, defined as a new set of model results constructed from the median of model predictions. The Median Model was shown to outperform the results of any single deterministic model in reproducing the concentration of atmospheric pollutants measured during the ETEX experiment [11].

Moreover, in [16] an approach for the statistical analysis of multi-model ensemble results is presented. The authors used a well-known statistical approach to multimodel data analysis, i.e. Bayesian Model Averaging (BMA), which is a standard method for combining predictive distributions from different sources. Moreover, similarities and differences between models were explored by means of correlation analysis.

We have to note, however, that if different models are used to simulate the same phenomenon, e.g. weather, climate or the dispersion of radioactive material, they probably will give similar responses. Potentially, model ensemble results may lead to erroneous interpretations, and this is more probable if models are strongly dependent. Models are certainly more or less dependent in the case of ensemble dispersion forecasting, since they often share similar initial/boundary data, numerical methods, parameterizations, and so on.

In this work we use a statistical approach to analyze the independence between model distributions and to select the models that have similar behavior. Substantially we use an agglomerative clustering approach to obtain a dendrogram that describes the relations between models.

To compare models, we propose to use an entropy information approach. From one hand we consider as “distance” the Kullback-Leibler (KL) divergence [14, 4]. This divergence can be considered as a kind of distance between two probability densities, because it is always nonnegative, and zero if and only if the two distributions are equal.

On the other hand Negentropy can be interpreted as a measure of non-Gaussianity and, if we consider the residues between two distributions, we can estimate how these two distributions are different. We use approximations of Negentropy providing a very good compromise between the properties of the two classic non-Gaussianity measures given by kurtosis and skewness [12].

In Section 2 we introduce KL and Negentropy information and successively the agglomerative approach. In Section 3 we show some results obtained from the application of these two approaches to data from the ETEX-1 experiment [11].

[1] Dept. of Applied Sciences, University of Naples “Parthenope”, Isola C4, Centro Direzionale I-80143, Napoli, Italy {angelo.ciaramella,giulio.giunta,angelo.riccio}@uniparthenope.it. Corresponding author: angelo.ciaramella@uniparthenope.it

[2] European Commission - DG Joint Research Centre, Institute for Environment and Sustainability, Ispra, Italy, stefano.galmarini@jrc.it

2 NegEntropy-Based Hierarchical Agglomeration

In this section we describe the hierarchical agglomerative approach. Substantially the aim is to analyze the distributions of the models at a given time and to agglomerate the distributions that have similar behavior.

2.1 Kullback Leibler divergence

The KL divergence is defined between two discrete n-dimensional probability density functions $\mathbf{p} = [p_i \dots p_n]$ and $\mathbf{q} = [q, \dots q_n]$ as

$$KL(\mathbf{p}||\mathbf{q}) = \sum_{i=1}^{n} p_i \log \left(\frac{p_i}{q_i} \right). \tag{1}$$

This is known as the relative entropy. It satisfies the Gibbs' inequality

$$KL(\mathbf{p}||\mathbf{q}) \geq 0 \tag{2}$$

where equality holds only if $\mathbf{p} \equiv \mathbf{q}$. In general $KL(\mathbf{p}||\mathbf{q}) \neq KL(\mathbf{q}||\mathbf{p})$. In our experiments we use the symmetric version [4] that can be defined as

$$KL = \frac{KL(\mathbf{p}||\mathbf{q}) + KL(\mathbf{q}||\mathbf{p})}{2}. \tag{3}$$

The relative entropy is important in pattern recognition as well in information theory. It is also used in Independent Component Analysis to estimate the independence between distributions [12].

2.2 Negentropy Information

The definition of Negentropy J_N is given by

$$J_N(\mathbf{x}) = H(\mathbf{x}_{Gauss}) - H(\mathbf{x}), \tag{4}$$

where $\mathbf{x}_{Gauss}$ is a Gaussian random vector of the same covariance matrix as $\mathbf{x}$ and $H(\cdot)$ is the differential entropy. Negentropy can also be interpreted as a measure of non-Gaussianity [12]. The classic method to approximate Negentropy relies on using higher-cumulants, through polynomial density expansion. However, such cumulant-based methods sometimes provide a rather poor approximation of the entropy. A special approximation is obtained if one uses two functions G^1 and G^2, which are chosen so that G^1 is *odd* and G^2 is *even*. Such a system of two functions can measure the two most important features of non-Gaussian 1-D distributions. The odd function measures the asymmetry, and the even function measures the dimension of bimodality vs. peck at zero, closely related to sub- vs. super-gaussianity. Then the Negentropy approximation of equation 4 is:

$$J_N(\mathbf{x}) \propto k_1 E\{G^1(\mathbf{x})\}^2 + k_2 (E\{G^2(\mathbf{x})\} - E\{G^2(\upsilon)\})^2 \tag{5}$$

where υ is a Gaussian variable of zero mean and unit variance (i.e. standardized), the variable $\mathbf{x}$ is assumed to have also zero mean and unit variance and k_1 and k_2 are positive constants. We note that choosing the functions G^i that do not grow too fast, one obtains more robust estimators.
In this way approximations of Negentropy that give a very good compromise between the properties of the two classic non-Gaussianity measures given by kurtosis and skewness can be obtained [7, 2, 12]. They are conceptually simple, fast to compute, yet have appealing statistical properties, especially robustness.

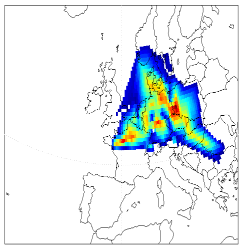


Figure 1. ETEX-1 observations

2.3 Agglomerative approach

We remark that our aim is to agglomerate by an unsupervised method the distributions obtained from the different models of the ensemble. Substantially the aim is to build a hierarchical tree (dendrogram) that permits to cluster models that have similar behavior. To obtain the dendrogram we calculate the dissimilarity matrix between the distributions of the models by using a Negentropy information or a Kullback-Leibler divergence. The densities of the distributions are calculated with a simple non-parametric method. In particular, in our experiments we used a histogram approach. We also observe that in the case of the Negentropy we use the residuum between the distribution to obtain a density.

Using this information we apply the agglomerative hierarchical clustering approach to obtain the dendrogram. In this case we use $complete\ linkage$ or $furthest\ neighbor$

$$d(r, s) = \max(J(X_r, X_s)) \tag{6}$$

where X_r and X_s are two distributions and $J(\cdot, \cdot)$ is one of the two entropy information.

3 Experimental Results

We apply the approach to the analysis of multi-model ensemble results. The ensemble analysed in this work is an extended version of that originally presented in [10]. To summarize, we are looking at 26 simulations [16] of the ETEX-1 experiment [11]. The ETEX-1 experiment concerned the release of pseudo-radioactive material on 23 October 1994 at 16:00 UTC from Monterfil, southeast of Rennes (France). Briefly, a steady westerly flow of unstable air masses was present over central Europe. Such conditions persisted for the 90 h that followed the release with frequent precipitation events over the advection area and a slow movement toward the North Sea region. In figure 1 we show the integrated concentration after 78 hours from release.

Several independent groups worldwide tried to forecast these observations. Each simulation, and therefore each ensemble member, is produced with different atmospheric dispersion models and is based on weather fields generated by (most of the time) different Global Circulation Models (GCM). All the simulations relate to the same release conditions. For details on the groups involved in the exercise and the model characteristics, refer to [10] and [16].

In order to analyze data, we consider the integrated concentrations of the 26 models to calculate the dendrogram and determine the models that have similar distributions.

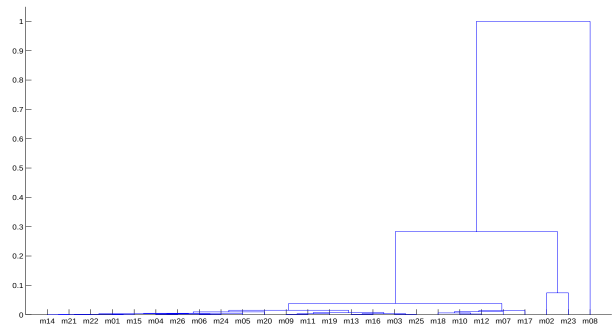


Figure 2. Dendrogram obtained by using the Negentropy information.

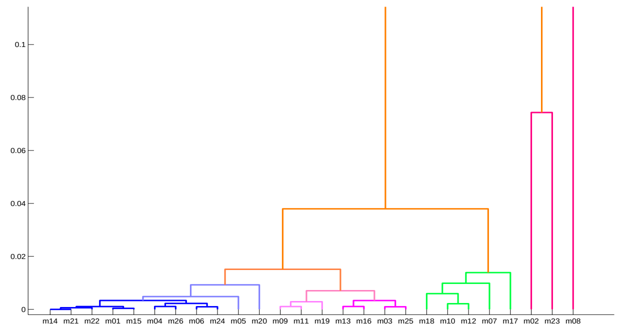


Figure 4. Details of the dendrogram obtained by using the Negentropy information.

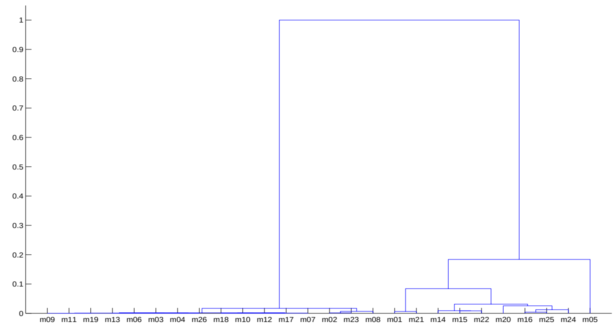


Figure 3. Dendrogram obtained by using KL divergence.

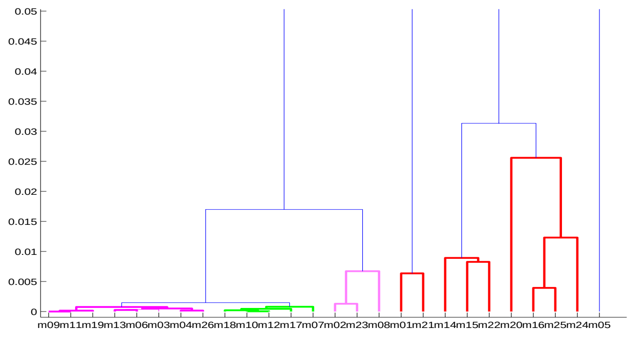


Figure 5. Details of the dendrogram obtained by using the KL divergence.

The dendrogram obtained by using the Negentropy information on the integrated concentrations after 78 hours is plotted in figure 2. We mark that the information on the abscissa are related to the models and those on the ordinate are related to the model similarities obtained by using the Negentropy. In figure 3 we plot the dendrogram obtained by using the KL divergence.

We observe that different clusters of independent models are obtained. In fact, in the Negentropy based dendrogram we can observe 4 mainly independent agglomerations identified in figure 4 with different colors. This clusters are partially confirmed by the KL dendrogram, though in this last dendrogram we obtain more than 4 agglomerations (in figure 5 red links identify clusters that are not partially confirmed in the Negentropy agglomeration).

We can identify the models that have similar behavior by analyzing the different clusters. For example in figure 6 we show the distributions of the models in one of the clusters (blue cluster in the Negentropy dendrogram). The visualized models are $m14$, $m21$, $m01$, $m15$, $m06$ and $m24$, respectively. In figure 7 we show the distributions in the magenta cluster of the Negentropy dendrogram. In this case the models are $m09$, $m11$, $m13$ and $m16$. We can stress that this cluster contains models that have a comparable distribution. Moreover, models $m02$, $m23$ and $m08$ (figure 8) are far from the other clusters. They have a similar distributions but they are more diffusive than the others (see figure 8). We however note that the same three models are agglomerated together in the KL dendrogram but they belong to another extended cluster. We also note from the KL dendrogram that some models probably are erroneously associated to some agglomerations. For example, model $m16$ is associated together with the model $m25$ but we can note that its distribution is closer to that of model $m13$ than $m25$. In fact, in the Negentropy based dendrogram models $m13$ and $m16$ are agglomerated. Moreover model $m13$ in the KL dendrogram is agglomerated with model $m06$, but, as we can see in figures 9c and 9d, they have a rather different distribution. In figure 9 we plot the distributions of all these models.

In order to identify the group of models that more appropriately describe observations, we re-apply the clustering approaches also including the distribution of observed values. This distribution is named $m00$. In figure 10 we plot the two dendrograms. We note that in the Negentropy dendrogram the model $m00$ is agglomerated in the blue cluster, together with models $m01$ and $m15$. Instead, in the KL dendrogram it is associated only with model $m14$.

4 Conclusions

In this work an approach to analyze the independence between different models describing atmospheric dispersion processes has been

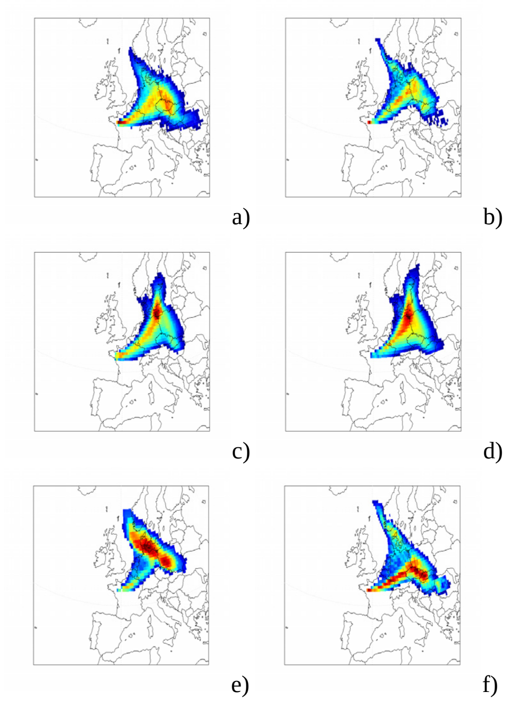


Figure 6. Distributions after 78 hours of the models $m14$ (a), $m21$ (b), $m01$ (c), $m15$ (d), $m06$ (e), $m24$ (f).

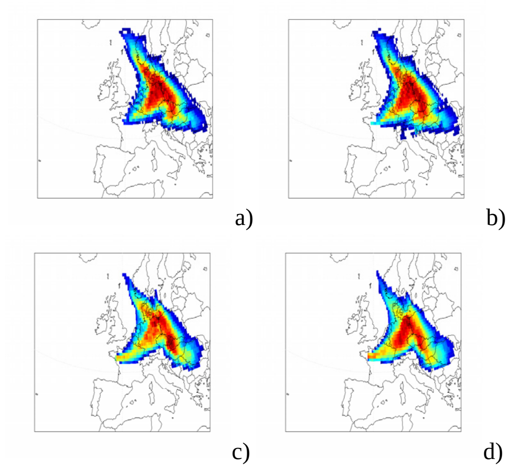


Figure 7. Distributions after 78 hours of the models $m09$ (a), $m11$ (b), $m13$ (c), $m16$ (d).

introduced. The proposed approach is based on a Negentropy information or a Kullback-Leibler divergence. By using this information a hierarchical agglomeration can be obtained. The dendrogram describes the relations between models' distributions. The approach is used to analyze the data obtained during the ETEX-1 exercise. The results show that the approach can be used to analyze the independence between the models. From the presented results the Negentropy information permits to obtain more easily interpretable information than the KL divergence.

In the next future, focus will be devoted mainly on how to extract clusters from the hierarchical agglomeration approaches to obtain

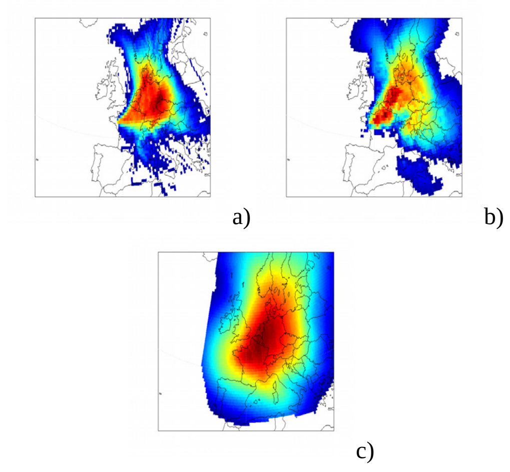


Figure 8. Distributions after 78 hours of the models $m02$ (a), $m23$ (b) and $m08$ (d).

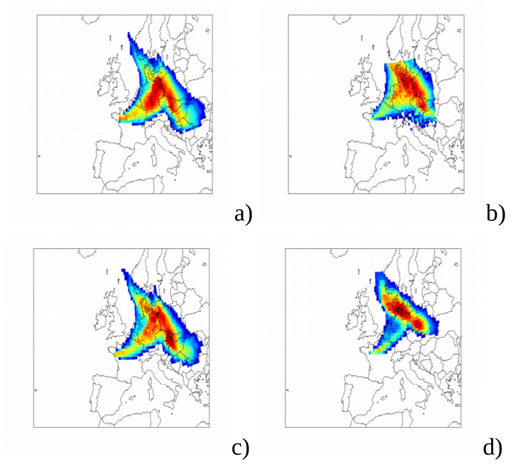


Figure 9. Distributions after 78 hours of the models $m16$ (a), $m25$ (b), $m13$ (c) and $m06$.

more accurate ensemble predictions. Moreover different measures will be used to compare models and by considering their temporal features.

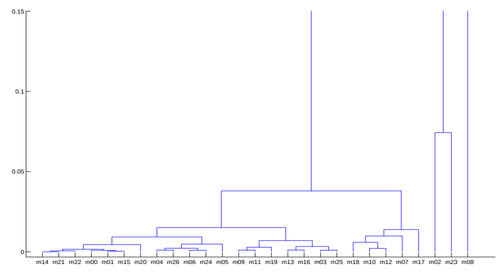
REFERENCES

[1] F. Acernese, A. Ciaramella, S. De Martino, R. De Rosa, M. Falanga, R. Tagliaferri, Neural networks for blind sources separation of Stromboli explosion quakes, *IEEE Trans. on Neural Netw.*, **14**, 1 (2003).

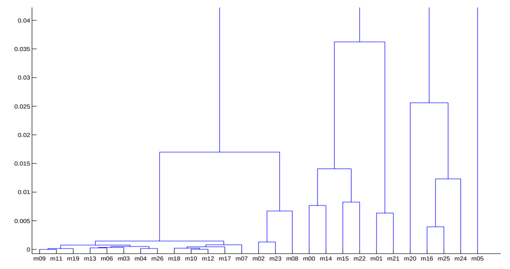
[2] R. Amato, A. Ciaramella, N. Deniskina, C. Del Mondo, D. di Bernardo, C. Donalek, G. Longo, G. Mangano, G. Miele, G. Raiconi, A. Staiano, R. Tagliaferri, A Multi-Step Approach to Time Series Analysis and Gene Expression Clusterings, accepted for publication on Bioinformatics.

[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[4] A. Ciaramella, R. Tagliaferri, Amplitude and Permutation Indeterminacies in Frequency Domain Convolved ICA, Proceedings of the IEEE International Joint Conference on Neural Networks 2003, vol. 1, pp. 708-713, 2003, IEEE PRESS;



a)



b)

Figure 10. Details of the dendrograms: a) by using Negentropy; b) by using KL.

[5] A. Ciaramella, E. de Lauro, S. De Martino, M. Falanga, R. Tagliaferri, Complexity of Time Series Associated to Dynamical Systems Inferred from Independent Component Analysis, Physical Review E. 72, 046712-1/14 (2005);

[6] A. Ciaramella, A. Staiano, R. Tagliaferri, G. Longo, NEC: a Hierarchical Agglomerative Clustering based on Fischer and Negentropy Information, LNCS Vol. 3931/2006, pp. 49-56. WIRN-NAIS 2005.

[7] A. Ciaramella, F. Napolitano, G. Miele, G. Raiconi, A. Staiano, R. Tagliaferri, Clustering and Visualization Approaches for Human Cell Cycle Gene Expression Data Analysis, selected for the special issue of the International Journal of Approximate Reasoning on Approximate Reasoning and Machine Learning for Bioinformatics.

[8] Galmarini, S., Bianconi, R., Bellasio, R., and Graziani, G.: Forecasting consequences of accidental releases from ensemble dispersion modelling, J. Environ. Radioactiv., 57, 203219, 2001.

[9] Galmarini, S., Bianconi, R., Klug,W., Mikkelsen, T., Addis, R., Andronopoulos, S., Astrup, P., Baklanov, A., Bartniki, J., Bartzis, J. C., Bellasio, R., Bompay, F., Buckley, R., Bouzom, M., Champion, H., DAmours, R., Davakis, E., Eleveld, H., Geertsema, G. T., Glaab, H., Kollax, M., Ilvonen, M., Manning, A., Pechinger, U., Persson, C., Polreich, E., Potemski, S., Prodanova, M., Saltbones, J., Slaper, H.,Sofief, M. A., Syrakov, D., Sorensen, J. H., Van der Auwera, L., Valkama, I., and Zelazny, R.: Ensemble dispersion forecastingPart I: concept, approach and indicators, Atmos. Environ., 38, 46074617, 2004a.

[10] Galmarini, S., Bianconi, R., Addis, R., Andronopoulos, S., Astrup, P., Bartzis, J. C., Bellasio, R., Buckley, R.,Champion, H., Chino, M., DAmours, R., Davakis, E., Eleveld, H., Glaab, H., Manning, A., Mikkelsen, T., Pechinger, U., Polreich, E., Prodanova, M., Slaper, H., Syrakov, D., Terada, H., and Van der Auwera, L.: Ensemble dispersion forecastingPart II: application and evaluation, Atmos. Environ., 38, 46194632, 2004b.

[11] Girardi, F., Graziani, G., van Veltzen, D., Galmarini, S., Mosca, S., Bianconi, R., Bellasio, R., and Klug, W.: The ETEX project. EUR Report 181-43 EN. Office for official publications of the European Communities, Luxembourg, 108pp., 1998.

[12] A. Hyvӑrinen, J. Karhunen and E. Oja, *Independent Component Analysis*, (Wiley-Sons, Inc., 2001).

[13] Krishnamurti, T. N., Kishtawal, C. M., Zhang, Z., LaRow, T., Bachiochi, D., Williford, E., Gadgil, S., and Surendran, S.: Multimodel ensemble forecasts for weather and seasonal climate. Mon. Weather Rev., 116, 907920, 2000

[14] D. J. C. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003

[15] Molteni, F., Buizza, R., Palmer, T. N., and Petroliagis, T.: The ECMWF ensemble system: Methodology and validation, Q. J. Roy. Meteor. Soc., 122, 73119, 1996.

[16] A. Riccio, G. Giunta, S: Galmarini, Seeking for the rational basis of the median model: the optimal combination of multi-model ensemble results, Atmos. Chem. Phys., 7, 6085-6098, 2007

[17] Toth, Z. and Kalnay, E.: Ensemble forecasting at the NMC: The generation of perturbations, B. Am. Meteorol. Soc., 74, 2317 2330, 1993

Improving Supervised Learning with Multiple Clusterings

Sébastien Derivaux, Germain Forestier and Cédric Wemmert [1]

Abstract. Classification task involves inducing a predictive model using a set of labeled samples. The more the labeled samples are, the better the model is. When one has only a few samples, the obtained model tends to offer poor result. Even when labeled samples are difficult to get, a lot of unlabeled samples are generally available on which unsupervised learning can be used. In this paper, a way to combine supervised and unsupervised learning in order to use both labeled and unlabeled samples is explored. The efficiency of the method is evaluated on various UCI datasets when the number of labeled samples is very low.

1 INTRODUCTION

The number of labeled samples is a crucial issue for supervised classification. If too few examples are given to a classical algorithm, the induced predictive model will be of poor performance. Sadly, in many real-world applications, labeled samples are difficult to obtain. This is often due to the cost of a human manual labeling. For example, we can cite all the problems where the user only gives few examples, and the system has to find similar objects in a database (content-base image retrieval, online web-page recommendation, . . .). In this cases, only few labeled samples are available although many unlabeled data are present (all the other instances in the database). In web-page recommendation, the user label interesting pages. It is not possible to ask to the user to produce other sample as he may no know another one. The same problem appears with online shop services when a user buys a product and the system wants to propose him other related products. The system only knows which products the user has bought and his notation of the product.

However, if labeled samples are rare, unlabeled samples are generally available in great quantities. Some research works have reveal that these samples can be used to improve supervised classification. The main idea is to partially classify the unlabeled samples using the labeled ones, and to use them to induce a new model [2, 10, 13].

Our method is slightly different of the existing one, as we use many unsupervised clustering to propose new features to describe the labeled samples. Then, a supervised classification is applied in this new data space. As unsupervised clustering creates clusters that tend to maximize intra-cluster similarity and inter-cluster dissimilarity, no labeled sample is needed, but no label is assigned to the different clusters. The clustering may be seen as a way to resume the distribution of the samples. It is sometimes used to reduce data before classification step.

As unsupervised learning can only use data distribution, classes must respect, to some extends, data distribution. If two classes form a cluster highly homogeneous in feature space, one can not expect a clustering to separate them. Even if our approach can weaken this condition, it must be kept in mind.

In Section 2, we first present some related works and situate our method. Then, the algorithm is expressly described in Section 3. In Section 4, we present many experiments made on UCI datasets. The results that we obtain are compared with classical supervised methods to quantify the improvement given by the unsupervised clustering pre-processing.

2 RELATED WORKS

Many past works have shown that unlabeled data can help to improve the quality of the classification, when very few labeled samples are available [6, 3, 7, 11, 4].

A first way to exploit unlabeled objects is the co-training method defined in [3]. The main idea is to use two complementary classification to iteratively label the unlabeled data. This assumes that there exist two independent and complementary feature sets on the data.

To extend this method and avoid the independence and redundancy of the feature sets, which is not realistic in real problems, Goldman and Zhou present in [6] a co-training strategy for using unlabeled data to improve the performance of a supervised classifier. Their method uses two different supervised learners which can select some unlabeled data to label the other learner in a iterative way. Experiments have shown that the method increases the accuracy of the ID3 algorithm. More recently, Raskutti et al. [12] expose a co-training method, that does not necessary need two complementary supervised learning algorithms. The idea is to produce an alternate view on the data by performing an unsupervised classification algorithm on all the dataset (labeled and unlabeled). Then, the original view and the view constructed by the clustering are used to create two independent predictors for co-training. In [14], the authors present a co-training approach which assumes to have two views of the data. The method uses the correlation between the two views to produce extra positive and negative samples in an iterative process. Experiments, where only one labeled sample is available, show that the method overcomes other co-training approaches.

Unlike co-training, ASSEMBLE [1] can build semi-supervised ensembles of any size, and does not require the domain to have multiple views. ASSEMBLE incorporates self-labeled examples in a Boosting framework. In each iteration of ASSEMBLE examples from the unlabeled set are labeled by the current ensemble and added to the training set. In [4], an empirical study of various semi-supervised learning techniques on a variety of datasets is presented. Different experiments are made to evaluate the influence of the size of the labeled and unlabeled sets, or the effect of noise in the samples. The paper

[1] LSIIT - CNRS - University Louis Pasteur - Pôle API, Bd Sébastien Brant - 67412 Illkirch, France, email:derivaux@lsiit.u-strasbg.fr

concludes that the performance of the methods is heavily dependent of the field of application and the nature of the dataset. However, using labeled *and* unlabeled sample improves the accuracy in most of the cases.

The method presented in this paper is slightly different of the ones presented above. All the co-training methods use the labeled and unlabeled samples together in the training step. If more labeled samples are available, all the training step needs to be computed again, which is often costly. In our approach, the unsupervised classification can be seen as a pre-processing step, which is performed only once. Then, depending on the availability of labeled samples, the supervised classification can be calculated. This training part is very quick as the number of samples is very low.

3 DESCRIPTION OF THE METHOD

Let $\mathcal{X}$ denote a set of n data objects $x_j \in \mathcal{X}$. We consider a q-class classification problem with m labeled and l unlabeled objects where m is very low and $l >> m$. Let $\mathcal{L}$ be the set of labeled objects of $\mathcal{X}$.

$$\mathcal{L} = ((x_1, y_1), \ldots, (x_m, y_m)) \tag{1}$$

where $y_i \in \{1, \ldots, q\}$ are the target values of the samples. Let $\mathcal{U}$ be the set of unlabeled objects

$$\mathcal{U} = (x_{m+1}, \ldots, x_{m+l}) \tag{2}$$

The main idea of the method is to improve the classification by first producing a clustering on the dataset. The clustering, computed on all the labeled and unlabeled objects, regroups the similar objects together, maximizing the intracluster similarity and the intercluster dissimilarity. If the classes of the problem are well separated in the feature space, we should be able to associate to each cluster one of the classes, using the class of labeled samples which belong to the cluster.

But in real problem, classes are not generally well separated. It is then possible to have samples from different classes in one cluster, or no sample in others. To avoid this, the proposed method uses a combination of multiple clustering methods.

A clustering is a partition of $\mathcal{X}$ into k clusters, and is represented as an n-dimensional cluster labeling vector

$$C = \left(C^j\right)_{j=1}^n \in \mathcal{C}^n \tag{3}$$

where $\mathcal{C} = \{c_1, \ldots, c_k\}$. We consider here b clusterings of the dataset $\mathcal{X}$, represented as an $n \times b$ matrix of cluster labeling vectors. Let $\mathbf{C}$ denote this set of clusterings, $\mathbf{C} = \{C_1, \ldots, C_b\}$. The idea is to assign to each labeled sample $x_i \in xy^m$, a new features vector

$$v(x_i) = \left(C_1^i, \ldots, C_b^i, y_i\right) \tag{4}$$

where C_j^i is the cluster assigned by the j^{th} clustering method C_j to x_i. Then, a predictive model $\mathbf{P} : \mathcal{X} \to \{1, \ldots, q\}$ can be induced from this new dataset $V = \{v(x_i)\}_{i=1}^m$, using a classical supervised learning method. Finally, the label $\mathbf{P}(x_i)$ is assigned to each unlabeled object x_i of $\mathcal{U}$.

The overall algorithm presenting the complete process of classification is given on Algorithm 1.

4 RESULTS

The method described in the previous section have been evaluated on various datasets of the UCI repository [9]. The Table 1 presents information about these datasets.

Algorithm 1 Classification with few labeled data

```
1: apply $b$ clustering algorithms to all the dataset $\mathcal{X}$
   $\rightsquigarrow$ $\mathbf{C} = \{C_1, \ldots, C_b\}$
2: for all $x_i \in \mathcal{L}$ do
3:     $v(x_i) = \left(C_1^i, \ldots, C_b^i, y_i\right)$ where $C_k^i$ is the cluster assigned to
       $x_i$ by the $k^{\text{th}}$ clustering method $C_k$ and $y_i$ is the label of $x_i$
4: end for
5: apply a supervised learning method to produce a predictive
   model $\mathbf{P}$ from $V = \{v(x_i)\}_{i=1}^m$
6: for all $x_j \in \mathcal{U}$ do
7:     assign $\mathbf{P}(C_1^j, \ldots, C_b^j)$ to $x_j$ where $C_k^j$ is the cluster assigned
       to $x_j$ by the $k^{\text{th}}$ clustering method $C_k$
8: end for
```

To apply the proposed method, we first have to choose how many clusterings will be run on the dataset (i.e. how many attributes will have each object in the new data space), and then, the different clustering methods. We choose four different configurations to study the importance of the number of clusterings. The four configurations are referred as follows:

1. *simple* : one EM (Expectation-Maximization [5])
2. *low* : one EM and one KMeans [8]
3. *medium* : two EM and two KMeans
4. *high* : c EM and c KMeans (with c the number of classes of the datasets).

Table 1. Information about the different datasets

Dataset	Nb. Classes	Nb. attributes	Nb. objects
iris	3	4	150
wine	3	13	178
ionosphere	2	34	351
diabetes	2	8	768
breast-w	2	9	699
anneal	5	38	898

Each method was run with a number of clusters equals to the number of classes actually present in the dataset except for the *high* configuration where the clustering method $k \in \{1, \ldots, c\}$ has k clusters. The four configurations have been compared with other algorithms, taken in different families of learning algorithms: the standard tree inducer C4.5, Naive Bayes and a 1-nearest-neighbor (1-NN) algorithm. Results are presented in Table 2. The number of samples used is indicated at the beginning of each line. We choose to evaluate the methods when $2, 4, 8$ and 16 samples per class were available. For the four configuration of the proposed method, 50% of the remaining data have been used for the unsupervised learning, and the other 50% for the evaluation of the method. We choose to use the Naive Bayes classifier as supervised method in output of the unsupervised learning step. As performance may greatly differ depending on the labeled set, the procedure is performed 20 times and the results are averaged. At each run, the different sets are filled with randomly chosen samples.

The proposed method outperformed the supervised learning when the number of samples was very low (2 or 4 samples per class). For example, it can be noticed on the *breast-w* dataset where, with two samples, the $high$ configuration reaches 95.73 instead of 83.52 for the best supervised approach (1-NN). One can notice that the best accuracy amongst the different configurations is reached with the

Table 2. Results according to the different datasets with $2, 4, 8$ and 16 labeled samples available. Values correspond to the means and the standard deviations on 20 runs.

Dataset		Simple	Low	Medium	High	C4.5	1-NN	NB
iris	(2)	72.43(±16.70)	71.18(±21.53)	**84.10(±9.95)**	82.92(±12.08)	55.97(±13.10)	79.58(±13.91)	65.00(±15.57)
	(4)	83.62(±10.90)	87.75(±7.96)	84.57(±11.40)	**87.90(±5.84)**	80.94(±11.95)	87.17(±12.09)	84.49(±11.72)
	(8)	89.92(±5.22)	88.81(±5.28)	90.32(±5.52)	89.84(±3.90)	90.16(±4.80)	**94.52(±3.74)**	92.06(±4.32)
	(16)	88.43(±5.22)	89.02(±4.90)	89.51(±5.23)	91.47(±5.76)	93.14(±4.23)	**95.98(±2.10)**	95.29(±2.93)
wine	(2)	79.83(±22.08)	85.29(±14.59)	**93.78(±3.66)**	90.29(±9.62)	50.35(±6.90)	88.14(±6.33)	55.47(±17.63)
	(4)	90.90(±7.81)	92.47(±8.98)	**96.02(±2.26)**	94.52(±3.40)	70.24(±12.48)	90.36(±6.54)	70.78(±11.69)
	(8)	95.19(±2.66)	94.09(±4.05)	**96.36(±2.00)**	96.10(±2.17)	84.61(±5.90)	93.12(±4.31)	92.53(±6.24)
	(16)	94.69(±3.78)	95.46(±3.62)	**96.23(±2.41)**	94.77(±3.59)	86.85(±3.85)	95.77(±2.28)	95.62(±3.20)
breast-w	(2)	84.34(±18.81)	85.73(±18.12)	95.39(±1.19)	**95.73(±2.19)**	63.38(±16.94)	83.52(±17.38)	83.03(±15.28)
	(4)	90.09(±14.32)	92.30(±9.77)	94.51(±1.52)	**95.49(±1.71)**	85.97(±9.15)	94.29(±3.06)	84.21(±16.81)
	(8)	94.65(±1.74)	94.94(±1.75)	94.96(±1.02)	**95.60(±1.61)**	89.34(±3.24)	94.72(±2.09)	94.46(±1.70)
	(16)	94.93(±1.21)	94.70(±1.46)	95.25(±1.48)	**96.29(±1.41)**	90.06(±2.65)	94.45(±2.70)	95.07(±1.79)
diabetes	(2)	52.63(±9.54)	54.96(±9.71)	**55.75(±5.66)**	59.78(±6.31)	52.45(±6.85)	55.56(±7.44)	51.83(±5.00)
	(4)	52.38(±5.75)	53.53(±6.53)	58.05(±10.68)	57.89(±8.28)	**63.03(±5.76)**	58.49(±5.61)	55.05(±6.08)
	(8)	57.89(±8.66)	57.17(±8.00)	56.62(±8.17)	62.97(±6.93)	63.99(±4.98)	63.34(±5.23)	**64.02(±5.86)**
	(16)	58.53(±8.70)	58.89(±8.04)	61.86(±4.66)	62.65(±8.78)	66.20(±5.08)	65.16(±3.72)	**68.82(±3.24)**
ionosphere	(2)	57.53(±14.84)	59.08(±13.65)	**67.87(±13.11)**	62.39(±14.53)	52.56(±10.27)	56.32(±9.65)	58.51(±9.48)
	(4)	59.42(±12.09)	64.83(±12.09)	67.65(±10.65)	**69.30(±10.26)**	63.58(±8.38)	67.18(±9.57)	63.02(±10.42)
	(8)	67.47(±10.83)	68.07(±11.07)	70.36(±11.81)	70.39(±10.74)	69.52(±8.79)	73.10(±9.28)	**82.05(±5.55)**
	(16)	69.69(±7.84)	69.78(±9.57)	72.50(±4.31)	76.03(±7.17)	81.56(±5.58)	78.88(±7.74)	**82.94(±4.25)**
anneal	(2)	61.68(±9.17)	56.48(±14.52)	67.05(±8.44)	68.96(±7.06)	54.55(±10.67)	**72.02(±8.63)**	45.09(±8.98)
	(4)	67.60(±9.02)	62.51(±12.25)	73.57(±5.25)	75.65(±4.44)	72.94(±10.01)	**81.89(±4.61)**	69.51(±7.97)
	(8)	68.94(±7.98)	72.08(±5.61)	76.22(±4.04)	78.14(±4.16)	83.80(±5.60)	**87.18(±3.30)**	85.16(±5.26)
	(16)	71.76(±6.43)	68.78(±7.54)	76.35(±5.20)	79.95(±3.62)	92.27(±3.60)	**90.09(±2.17)**	88.80(±2.81)

medium and *high* ones. This result enforces the intuitive feeling that adding more clusterings improves the result, as the objects are described with more details (i.e. have more attributes). The Fig. 1 (a),(b),(c) and (d) illustrates this result and shows the increase of accuracy according to the number of available samples. This figure also illustrates that when the number of samples increases, the supervised approaches give better results. For example, when 16 are available, the supervised methods outperformed the proposed approach on 5 of the 6 datasets. It confirms that supervised methods need much examples to produce efficient predictive models.

As stated in the introduction, if the data space of the dataset is not correlated with the class information, using clustering is useless. This affirmation can be study on the *anneal* dataset, where using clustering is less interesting than a standard 1-nearest-neighbor, regardless of the number of available samples.

We also evaluate the influence of the size of the dataset available for the unsupervised learning. The Fig. 1 (e) and (f) show the evolution of the accuracy according to the size of the dataset used for the unsupervised learning (10%, 25% and 50% of the datasets). The Fig. 1 (f), corresponding to the evaluation on the *wine* dataset, indicates that the increase of unlabeled samples available helps to produce better results. This is due to the ability of the clustering to better grasp the dataset when the density of objects increases.

5 CONCLUSION

In this paper it has been shown that many clustering results can be combined through a supervised classification, in order to achieve better results than traditional algorithms, when the number of labeled samples is very low, and when unlabeled samples are available.

Nevertheless some questions remain open. How many clustering algorithms must be used ? Is it better to enhance diversity amongst them ? How to detect if a specific dataset can use this method ? These few questions give us some directions to consider in order to improve the presented work.

REFERENCES

[1] Kristin P. Bennett, Ayhan Demiriz, and Richard Maclin, 'Exploiting unlabeled data in ensemble methods.', in *KDD*, pp. 289–296. ACM, (2002).

[2] A. Blum and T. Mitchell, 'Combining labeled and unlabeled data with co-training', in *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, (1998).

[3] Avrim Blum and Tom Mitchell, 'Combining labeled and unlabeled data with co-training', in *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, (1998).

[4] Nitesh V. Chawla and Grigoris J. Karakoulas, 'Learning from labeled and unlabeled data: An empirical study across techniques and domains.', *J. Artif. Intell. Res. (JAIR)*, **23**, 331–366, (2005).

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, 'Maximum likelihood from incomplete data via the EM algorithm', *Journal of the Royal Statistical Society*, **39**(1), 1–38, (1977).

[6] Sally Goldman and Yan Zhou, 'Enhancing supervised learning with unlabeled data', in *Proc. 17th International Conf. on Machine Learning*, pp. 327–334. Morgan Kaufmann, San Francisco, CA, (2000).

[7] Thorsten Joachims, 'Transductive inference for text classification using support vector machines', in *Proceedings of ICML-99, 16th International Conference on Machine Learning*, eds., Ivan Bratko and Saso Dzeroski, pp. 200–209, Bled, SL, (1999). Morgan Kaufmann Publishers, San Francisco, US.

[8] J. MacQueen, 'Some methods for classification and analysis of multivariate observations', in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, eds., L. M. Le Cam and J. Neyman, pp. 281–297, (1967).

[9] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

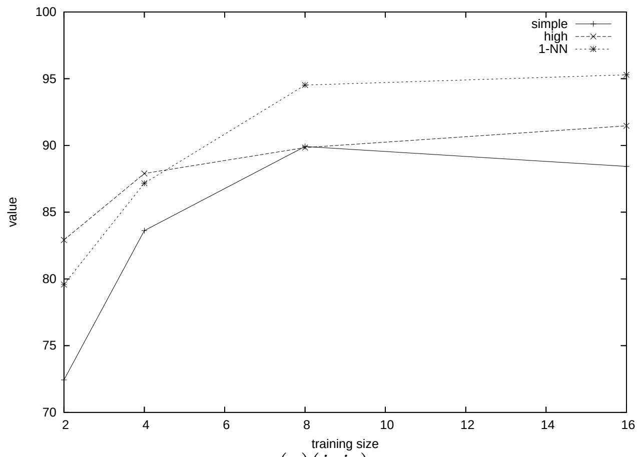
[10] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, 'Text classification from labeled and unlabeled documents using EM', *Machine Learning*, **39**(2/3), 103–134, (2000).

[11] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell, 'Text classification from labeled and unlabeled documents using EM', *Machine Learning*, **39**(2/3), 103–134, (2000).

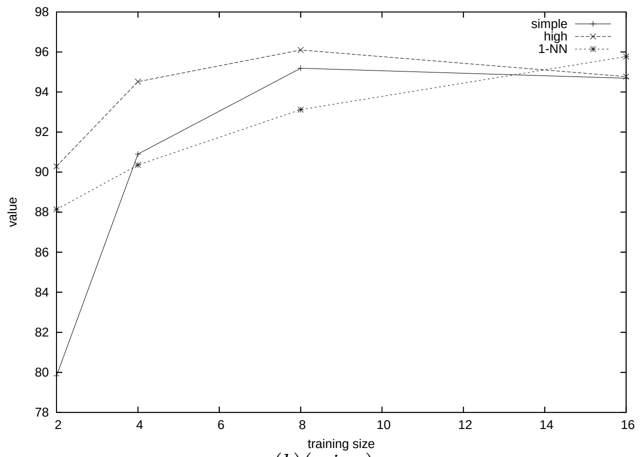
[12] Bhavani Raskutti, Herman L. Ferr, and Adam Kowalczyk, 'Combining clustering and co-training to enhance text classification using unlabelled data.', in *KDD*, pp. 620–625. ACM, (2002).

[13] M. Seeger, 'Learning with labeled and unlabeled data', Technical report, University of Edinburgh, (2002).

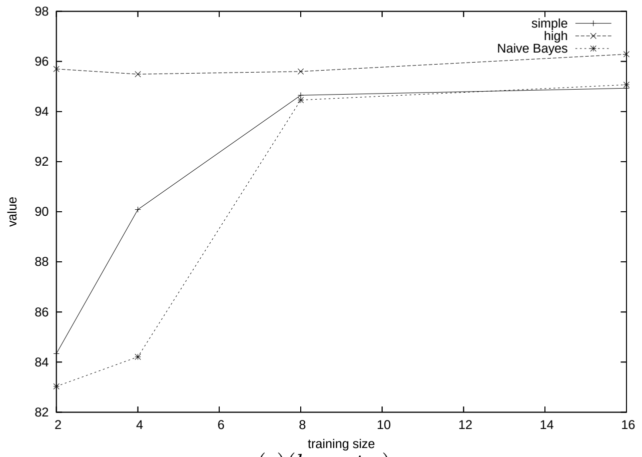
[14] Zhi-Hua Zhou, De-Chuan Zhan, and Qiang Yang, 'Semi-supervised learning with very few labeled training examples.', in *AAAI*, pp. 675–680. AAAI Press, (2007).



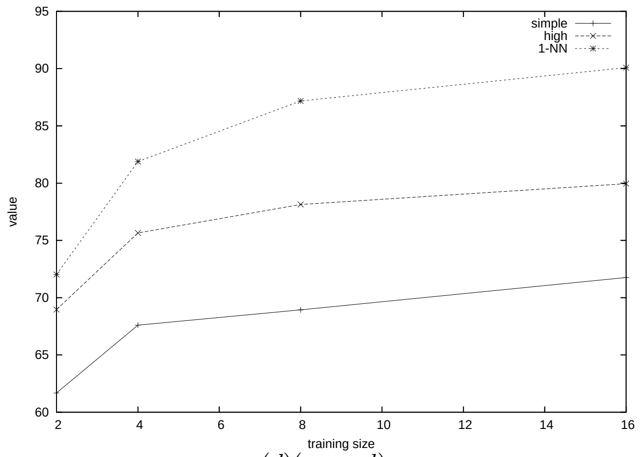
$(a)(iris)$



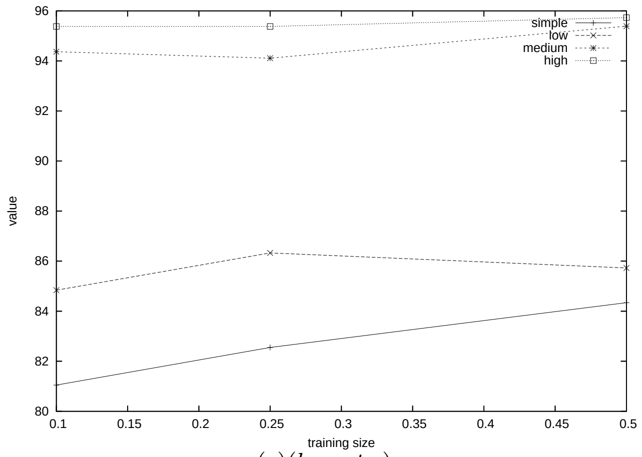
$(b)(wine)$



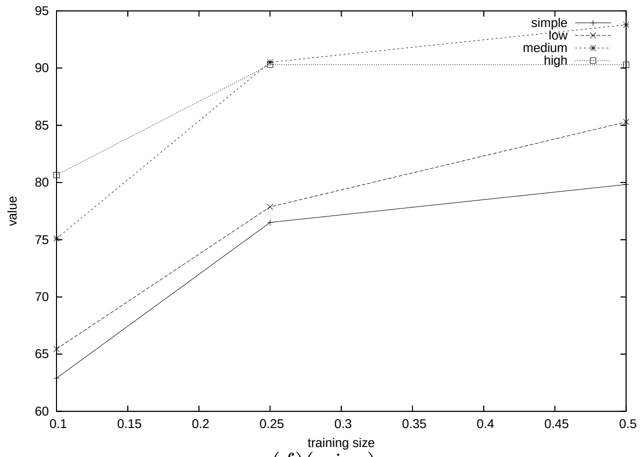
$(c)(breastw)$



$(d)(anneal)$



$(e)(breastw)$



$(f)(wine)$

Figure 1. (a),(b),(c) and (c) show the accuracy according to the number of available samples. (e) and (f) show the accuracy according to the size of the dataset used for the unsupervised learning.

Partitioner trees: combining boosting and arbitrating

Georg Krempl and **Vera Hofer** [1]

Abstract. Many classifier ensembles focus on creating diverse classifiers by varying training data. This can be done randomly like in bagging, or systematically by using information about the accuracy of each classifier within an ensemble. One systematical approach is boosting, where series of classifiers are created with each classifier focusing on the misclassifications of its predecessor. Observed ambiguity can be used to train arbiters or to derive new features used for training combiners or for stacking. Classifiers' self-confidence in predictions can be used to delegate uncertain instances to specialised classifiers. External referee–classifiers can be used to predict classifier accuracy, like in arbitrating or grading. However, this paper presents a new technique called partitioner trees that combines boosting and arbitrating. Initially, a preliminary classifier is trained and applied to the training data. Information on this classifiers' accuracy is then used to train a referee called partitioner. This partitioner then splits the data into two disjoint subsets on which two new local classifiers are trained. This process is iterated until a stop criteria is satisfied. Thus, a binary tree with partitioners on the inner nodes and local classifiers on the leafs is constructed. New instances are assigned to branches in the tree by the partitioners and descending the tree until they reach a leaf with a local classifier. This local classifier is then used for prediction. Furthermore possible extentions are discussed and experimental comparisons of single classifiers with partitioner trees, adaboost and bagging are given.

1 INTRODUCTION

Two key issues in the reseach of classifier ensembles are the creation of error diverse classifiers and their combination. Partridge and Yates [20] provide definitions of error diversity and discuss approaches to create error diverse neural networks [14]. They conclude that the most promising approaches are to vary the training data and the network architecture of a neural network, while varying the initial weights and the number of nodes is least useful.

The approach of varying training data randomly is used by *Bootstrap aggregation*, or *Bagging*, a method proposed by Breiman [4]. Different subsets of the training data are created by randomly selecting instances from the original data. Since this is done with replacement, an instance present in the original training data might be present once or more often in a new training set, or might not be present at all. Then each classifier is trained on a different subset. Finally, new instances are classified by combining the prediction of all classifiers by majority voting.

Instead of leaving the variation of training data to chance, *Boosting* techniques use systematic variation. An example of this technique initially proposed by Schapire [16] is the *AdaBoost*–Algorithm presented by Freund and Schapire [11]. Here each classifier focuses on the instances that his predecessors misclassified. This is done by assigning misclassified instances more weight within the data set. New instances are then classified by combining the predictions of *all* classifiers by majority voting. Due to weighting the data during the training process all but the first classifier are trained on data that systematically differ in their composition from the new data on which the classifiers are later used.

A more complex way of combining classifiers is to use meta-classifiers. Chen and Stolfo [9] proposed such techniques and suggested *arbiters* and *combiners*. Initially, several error-diverse base classifiers are trained. In their first approach instances on which predictions are ambiguous are used to train a so-called *arbiter*. This arbiter then decides which base–classifier to use for the final decision. In their second approach new features are derived from the base–classifiers predictions. On this derived features a *combiner* is trained to finally predict the correct class. Thus, the final decision can differ from all of the base classifiers decisions. Another approach that uses such information about previous misclassification as input for a succeeding classifier is *Stacked Generalisation*, or *Stacking*, proposed by Wolpert [19]. In this approach the training data is first split into two disjoint subsets, then several base learners are trained on one subset and tested on the other. Their predictions are used as inputs for a second level of classifiers which are trained to predict the correct value of the response variable.

Combinations of the idea of *boosting* and using meta–classifiers to assess each classifiers expertise on a new instance are also discussed in literature. On the one hand, uncertain instances can be delegate to other classifiers based on the base-classifiers own confidence in its predictions. Such a method of *delegating* is suggested by Ferri et al. [10], where succeeding classifiers are solely trained on instances on which its predecessors where uncertain. On the other hand an external classifier can be trained to assess the quality of a classifiers' prediction. Such an approach is the method of *arbitrating* between classifiers proposed independently by Ortega and by Koppel and Argamon [13]. They propose to train a *referee* for each base–classifier which estimates its reliability on a new instance. The estimates of all referees are used to arbitrate between the base–classifiers and to select one classifier whose prediction is finally used. Another approach that uses an external classifier is *grading*, proposed by Seewald and Fürnkranz [17]. Instead of selecting only the classifier with the highest reliability they combine the prediction of all trustworth classifiers. This combination of the predictions of a subset of base–classifiers is done either by voting or by weighting these predictions by their confidence estimates.

Such a referee can also be used as a *partitioner* to split the data into two disjoint subsets. Instead of keeping the initial base–classifier for instances where it succeeded, the *preliminary* base–classifier is discarded. Thus, on *both* subsets a new specialised local classifier must be trained. New instance are then assigned by the partitioner to one of

[1] Department of Statistics and Operations Research, University of Graz, Austria, email: georg.krempl@uni-graz.at; vera.hofer@uni-graz.at

the two new *local classifiers*. By iterating this approach a *partitioner tree* with *partitioners* on the inner nodes and *local classifiers* on the leafs is created. This new combination of the approaches described above will be discussed in this paper. This paper and the following section focus on the discussion of this method and its possible extentions, applications and advantages for some classification tasks. Nevertheless, the third section provides results on a dataset from the University of California (UCI) Machine Learning Repository. However, they are solely meant to show on which base–classifier–methods this approach is most promising to pioneer further research and comparisons on other data sets. Thus, an overview and discussion on possible research topics for future research is given in the last section.

2 PARTITIONER TREES

A tree of classifiers is constructed by iterating the process of

1. creating a preliminary classifier,
2. testing and dismissing this preliminary classifier,
3. using information about the accuracies of the preliminary classifier to train a meta-classifier (a so-called partitioner),
4. using this partitioner to split the data into two disjoint subsets,
5. repeating this process by training a preliminary classifier on each subset,
6. deciding for each branch wheather to continue with partitioning and branching or to stop by using the preliminary classifier as local classifier.

New instances are then predicted by

1. using the (first) partitioner to decide which branch to follow,
2. using the next partitioner on this branch to descend further in the tree and iterating this process until
3. a leaf is reached and its local classifier is used to predict the instance.

The approach of dismissing the preliminary classifier and training new local classifiers on both subsets can improve the ensemble performance due to the following reasons: In practice, a partitioner (or referee) will not be able to predict exactly on which instances a classifier might fail. Otherwise constructing an error-free ensemble for binary classification tasks would be possible by simply using the referee to decide whether the predicted class or its opposite are to be used. Thus, it might be hard for a partitioner to exactly determine areas of expertise. This problem is avoided by replacing the need for accurately predicting misclassifications by the easier task to *split* data *systematically* and *consistently* into two subsets. Thus, our approach seeks to guarantee that the structure of new data in a subset will be exactly the same as in the training data. This higher representativeness might yield in more accurate classifiers and thus in a lower *validation* error. For this technique which can be seen as a combination of *AdaBoost* and *arbitrating*, the meta-classifier also needs to be *applied* during the ensemble construction phase which can yield to longer training time. However, its application phase differs only slightly from its paragons. In the following subsections further details on how to construct and apply this partitioner tree are explained.

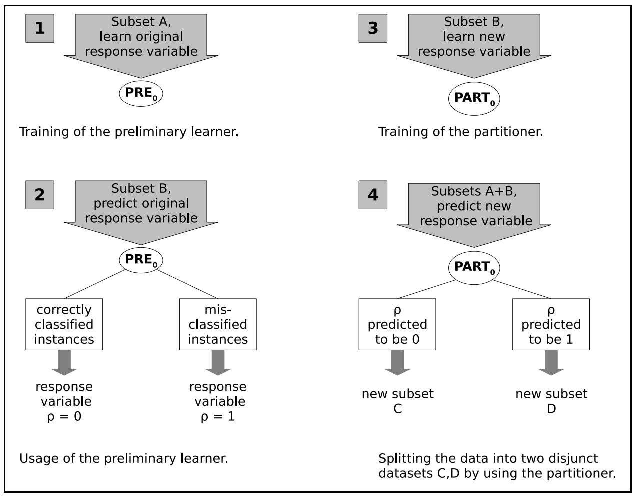
2.1 Construction phase

Initially, the training data is split randomly into two disjoint, equally sized subsets A and B. On the first subset, A, a *preliminary classifier* (PRE) is trained. This preliminary classifier is then used to predict the class membership of the data in the training set B. This yields a new dichotome response variable, ρ. This variable is zero, if the prediction was correct, and otherwise one. This is shown in the left column of figure 1.

Now a new classifier, referred to as *partitioner* (PART), is trained on the dataset B to predict the value of ρ. This partitioner is used on the complete training set to predict ρ. These predicted values of ρ, denoted as ρ_{pred}, are used to split the training data into two disjoint subsets C and D. The first subset contains all instances where $\rho_{\text{pred}} = 0$, the second all instances where $\rho_{\text{pred}} = 1$. This process is shown in the right column of figure 1.

If sufficient instances are in each of the two subsets, the process can be restarted. In this case, each subset (C and D) is divided into two disjoint subsets (C_A, C_B, D_A and D_B). On each subset (C_A and D_A) a new specialised preliminary learner is trained to be applied to C_B (or D_B, resepectively) to obtain new partitioners. However, if one subset becomes too small, there are two possible ways to proceed: First, the partitioner can be modified to create subsets more commensurated in size. Second, the process can be finalised by learning a *local classifier* (L) on the complete training set (thus both subsets C and D are merged into a single training set) [2]. This iteration on the second level of the partitioner tree is depicted in figure 2. In the case of a dichotome response variable, the resulting tree will be a binary tree as shown in figure 3.

Figure 1. Creation of the first partitioner in the first level.


1
Subset A, learn original response variable
PRE$_0$
Training of the preliminary learner.
3
Subset B, learn new response variable
PART$_0$
Training of the partitioner.
2
Subset B, predict original response variable
PRE$_0$
correctly classified instances
mis-classified instances
response variable ρ = 0
response variable ρ = 1
Usage of the preliminary learner.
4
Subsets A+B, predict new response variable
PART$_0$
ρ predicted to be 0
ρ predicted to be 1
new subset C
new subset D
Splitting the data into two disjunct datasets C,D by using the partitioner.

2.2 Application to new data

New instances are first assigned to a local classifier on a leaf of the partitioner tree which is then used to predict their class membership. The assignment starts at the first partitioner node which is connected to two other nodes in the tree. Based on the value of ρ_{pred} given to the new class by the first partitioner, one of the two succeeding nodes is selected. This succeeding node is either an inner node with

[2] The preliminary learner might be used as a local classifier to reduce the training time. Nevertheless, this preliminary classifier was only trained on half of the locally available training data which might result in reduced performance.

a partitioner. In this case the process is repeated by using this local partitioner to estimate a new value of ρ_{pred} and the algorithm descends to the corresponding node. Or the node is a leaf with a local classifier. Then the local classifier is used to predict the final class membership of the new instance.

2.3 Parameters and complexity

The algorithm requires two kinds of parameters to be set: First, the types of classifiers used as partitioners and local classifiers. Second, the stop criteria for branching.

2.3.1 *Choice of classifiers types*

Base classifiers are used for two different tasks in the algorithm, for partitioning (actually predicting misclassifications) on the one hand and for classifying the data on the other hand. For the first task, *consistency* in predictions is crucial, while accuracy is not. For the second task, which concerns preliminary classifiers and final local classifiers, *accuracy* is the primary issue. Although it would be possible to use a different types of classifier for each task, this paper focuses on partitioner trees using the same type of classifier for both tasks.

Figure 2. Creation of the partitioners in the second level

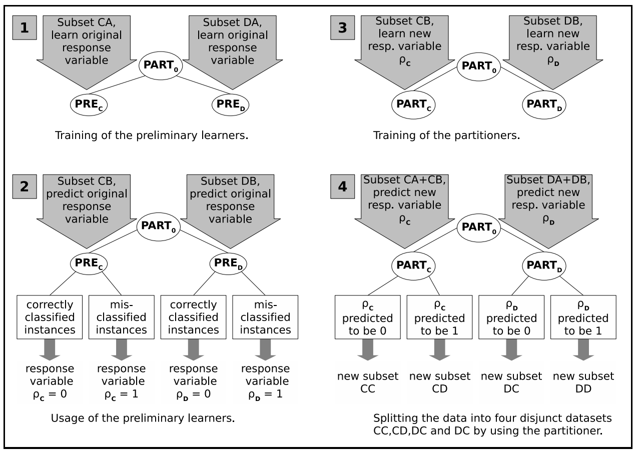
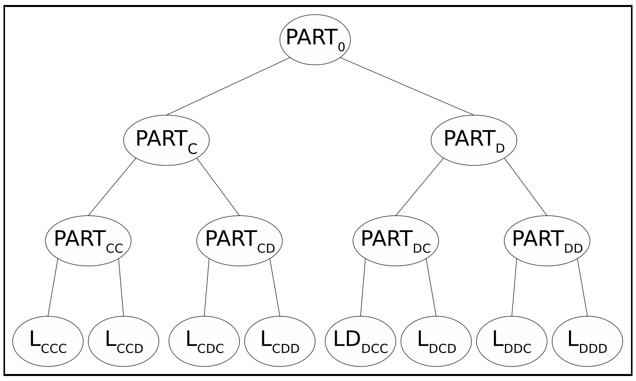

1
Subset CA, learn original response variable
Subset DA, learn original response variable
PART$_0$
PRE$_C$
PRE$_D$
Training of the preliminary learners.
3
Subset CB, learn new resp. variable ρ_C
Subset DB, learn new resp. variable ρ_D
PART$_0$
PART$_C$
PART$_D$
Training of the partitioners.
2
Subset CB, predict original response variable
Subset DB, predict original response variable
PART$_0$
PRE$_C$
PRE$_D$
correctly classified instances
mis-classified instances
correctly classified instances
mis-classified instances
response variable $\rho_C = 0$
response variable $\rho_C = 1$
response variable $\rho_D = 0$
response variable $\rho_D = 1$
Usage of the preliminary learners.
4
Subset CA+CB, predict new resp. variable ρ_C
Subset DA+DB, predict new resp. variable ρ_D
PART$_0$
PART$_C$
PART$_D$
ρ_C predicted to be 0
ρ_C predicted to be 1
ρ_D predicted to be 0
ρ_D predicted to be 1
new subset CC
new subset CD
new subset DC
new subset DD
Splitting the data into four disjunct datasets CC,CD,DC and DC by using the partitioner.

Figure 3. Partitioner Tree of depth three.



While all kinds of classification techniques including ensemble of classifiers themselves can be considered to be used as base classifiers, the choice has to be made with regard to three aspects: First, the time needed to train and to apply the classifier. Second, the capability of a classifier to estimate class probabilities and third the levels of the response variable. Further necessary considerations are the classifiers' complexity and its dependence on the full training data. In section 3.3 some suggestion will be made on the choice of this parameter.

2.3.2 *Tree depth*

The tree depth d, defined as the number of levels of partitioners in the tree, can either be fixed in advance or determined indirectly. In the first case the tree is branched until the defined depth is reached. In the second case definition of additional stop criteria is required. They can be based on a minimal subset size, on a maximum time limit or on the accuracy of the predictors in a branch. While the first and second are obvious, two approaches exist for using the accuracy: Either a maximum accuracy is reached, for instance if only one class is left in a subset. Or the cumulated performance of both branches drops below the performance of its preceding branch. Since disjoint subsets are used for training and testing, this accuracy is equal to the validation error.

These four parameters have to be chosen with regard to the size of the training set, the number of features in the data, the used base classifiers and the available training time. Since the training data is split into smaller subsets, in general 2^d - times more training data is needed than for a single classifier [3].

If not enough training instances are available, the algorithm might be modified to not split the training data into the two subsets A and B. Instead, the complete training data available at each iteration might be used for training and testing the preliminary local classifier. In this case, the training error is calculated and used instead of the validation error for partitioner creation and accuracy calculation. The algorithm used in section 3 of this paper uses only a minimum number of instances in each subset and a maximum tree depth as stop criteria.

2.3.3 *Computational complexity*

Training
For a tree depth d the proposed approach requires the training of 2^d final local classifiers and of $2^d - 1$ preliminary local classifiers and partitioners, respectively. Thus $3 \cdot 2^d - 2$ classifiers need to be trained. Nevertheless, the training and testing of all 2^i (preliminary) local classifiers on each level i can be done parallelly, followed by a parallel training and testing of all partitioners on that level. Thus, the training time complexity of $O(d^2)$ on a single-processor machine can be reduced to $O(d)$ on a multiprocessor grid, unlimited parallel processing capacities presumed. This is comparable to an *AdaBoost* algorithm, where all classifiers need to be trained and tested subsequently, thus also resulting in a time complexity of $O(d)$ (where d denotes the number of different classifiers involved). However, a bootstrap aggregation approach would allow a parallel training of all classifiers at once. Thus, a time complexity of $O(1)$ on a multiprocessor grid results, again unlimited parallel processing capacities presumed.

[3] This factor is a lower bound, provided that the data is splitted always into two equally sized subsets at each node. In practice even more training data might be required.

Prediction
Prediction of new data has to be done by using the classifiers sequentially. Thus, the prediction of a new instance has a time complexity of $O(d)$. Since in *AdaBoost* and bagged aggregation all classifiers can be used parallelly, the prediction of a new instance would result in $O(1)$ on a multiprocessor grid, with the same presumptions as above.

2.4 Possible extentions

2.4.1 Possible extentions and variations

By using all partitioners in the tree for each leaf the probability of a new instance to belong to it can be calculated. This probabilities can be used to weight and combine the predictions of all local classifiers for this new instance. Such an extention could be seen as a combination of *AdaBoost* and *grading* and might be of particular interest for regression problems.

2.4.2 Nominal response variables

The application of partitioner trees for classification tasks depends on the methods used for the base classifiers. Provided the use of a proper base classifier this approach is therefore not limited to the classification of dichotome response variables. In the case of nominal response variables there are three possible approaches: First, it is possible to use several partitioner trees, like other classification techniques for binary response variables, in an "one-against-all" approach and then combining their output.

Second, if a base classifier is chosen that supports classification of a nominal response variable, the partitioner tree can be used as described above. This means that ρ stays a dichotome response variable, where a value of one indicates any error. As partitioner also base classifiers only suitable for dichotome response variables can be used.

Third, as an extention to the second approach, ρ can be changed itself to a nominal variable by distinguishing the different types of errors. In this case, it is possible to create a separate branch for each kind of error. Nevertheless, it has to be taken into account that $(m^2 - m)$ different errors can occur, where m is the number of levels of the response variable. This means that the resulting tree will spread very fast and the training data will be divided into smaller subsets.

3 EXPERIMENTS

In preparation to further research on this algorithm, information is needed which basic–classifier–types are most promising. Thus, different types of base–classifiers and different tree depths will be compared. Due to partitioning, this algorithm requires a large number of training instances. Thus, a dataset with a very large number of instances was chosen and compared using implementations of classifiers for the R environment for statistical computing [15].

3.1 Data

From the University of California Machine Learning Repository [2] the *CoverType* data set was used, which was provided by Jock A. Blackard, Dr. Denis J. Dean and Dr. Charles W. Anderson [3][4]. This data set includes data about the forest cover in the Roosevelt National Forest of northern Colorado, where seven major cover types are distinguished, belonging to two divisions (Pinophyta,Magnoliophyta) of plants. The data set includes 54 quantitative features and one categorical response variable.

To obtain a data set with a dichotome response variable, the new response variable *division* was introduced, which was one if the tree was belonging to the division of Magnoliophyta, and zero otherwise. Due to he fact that fields covered with trees belonging to the division of Magnoliophyta made up only approximately 2.1% of all instances, the data was modified by selecting only the four groups Cottonwood/Willow, Aspen, Douglas-fir and Krummholz out of the seven tree cover groups. Finally, the original response variable was replaced by the new dichotome response variable.

Thus, a data set with 50117 instances, thereof 12240 Magnoliophyta (class 1, 24.4%) and 37877 Pinophyta (class 0, 75.6%), was created. 50 different partitionings of this data into 37587 training instances (75%) and 12530 validation instances (25%) were created by random permutation and used in the experiments.

3.2 Compared methods

We used the following methods as base classifiers:

- Linear Models (LM) [8], provided by the R-functions *lm* and *predict.lm*, see also [18].
- Generalised Linear Models (GLM) with logistic regression as link function, provided by the R-functions *glm* and *predict.glm*, see also [8].
- Single-Hidden-Layer Neural Networks (NNET) with size=2 and randomly created initial weights, provided by the R-functions *nnet* and *predict.nnet* from the package *nnet*.
- Support Vector Machines (SVM) with default settings [5], provided by the R-functions *svm* and *predict.svm* from the package *e1071* by David Meyer.
- Recursive Partitioning Trees (RPART) with default settings [6], provided by the R-functions *rpart* and *predict* from the package *rpart*, see also [7].
- *AdaBoost* (ADA) with classification trees (rpart) as single classifiers and a tree depth of 30, provided by the R-functions *adaboost.M* and *predict.boosting* from the package *adabag* by Esteban Alfaro, Matias Gamez and Noelia Garcia, see [1], [12] and [6].
- Booststrap Aggregation (BAGG), again with classification trees (rpart) as single classifiers and a tree depth of 30, provided by the R-functions *bagging* and *predict.bagging* from the package *adabag* by Esteban Alfaro, Matias Gamez and Noelia Garcia [1], [6] and [5].

Each method was used as a single classifier and as base classifier for a partitioner tree. The maximal tree depths were set to 0 (single classifier), 2 and 4, exept for *AdaBoost*, *Bagging* and *SVM*, where due to the needed long training time only maximum tree depths of 0 and 2 were used.

For each of the 50 partitions, these methods were trained on the training set and finally tested on the different validation set. Finally, the average performance over these 50 validation sets was calculated.

[4] This data is copyrighted by the Remote Sensing and GIS Program, Department of Forest Sciences, College of Natural Resources, Colorado State University, Fort Collins, CO 80523.

[5] The default setting for this data with 54 features was a radial kernel with a gamma value of 54^{-1} and a cost parameter of 1.

[6] The default method on the used data is "class".

3.3 Experimental results

On this particular dataset the partitioner tree could increase the performance of most classification techniques compared to single classifiers. Only support vector machines and *AdaBoost* obtained worse or equal results when used in a partitioner tree framework. However, the best result regarding the arithmetic mean over all 50 partitionings was obtained by a single *Adaboost* algorithm (98.59% accuracy), compared to 97,61% accuracy of a partitioner tree with bagged recursive partitioning trees as base classifiers[7]

On bagged classifier ensembles the performance was increased from 94.53% to 97.61%. The performance of linear models could be increased to 90.84% (compared to 86.68%), those of generalised linear models to 88.95% (compared to 82.70%). On neural networks, the partitioner tree approach increased the performance to 76.79% (compared to 75.93%). Regarding classification trees, a partitioner tree approach increased the performance to 97.25% (compared to 94.16%) on a RPART classifier.

Below the average performances of each method (rows) on different tree depths (columns) are given.

Table 1. Average performances on the validation sets

	Tree depth		
Method	**0**	**2**	**4**
LM	86.68 %	90.84 %	90.88 %
GLM	82.70 %	88.64 %	88.95 %
NNET	75.93 %	76.79 %	76.33 %
SVM	75.60 %	75.60 %	N/A
RPART	94.16 %	97.25 %	97.25 %
ADA	98.59 %	98.47 %	N/A
BAGG	94.53 %	97.61 %	N/A

3.4 Discussion of results

This algorithm aims to reduce complexity within the data. Thus, methods might not profit from this approach, if they depend or make largely use of such structures. This preliminary tests have shown that *Support Vector Machines* did in general not profit from this approach, nor did *AdaBoost*. Since AdaBoost makes use of the same characteristics like this algorithm, this result is not surprising. As far as Support Vector Machines are concerned, the fragmentation of the data might have prevailed to chose the optimal parameters for projection. Since the number of instances in the training data is reduced dramatically methods that increase the instances by *bootstrapping* might increase the performance. The experimental results seem to confirm this hypothesis, however, further research will be necessary on this topic. However, the performance of this algorithm used with bagging was comparable to the performance of a single *AdaBoost*, with 97.25% compared to 98.59%, and better than single *Bagging* (only 94.53)%.

4 CONCLUSION

A new meta-algorithm for creating classifier ensembles was presented that creates a binary tree of so-called partitioners, which are trained to predict misclassifications made by preliminary local classifiers. This binary tree is completed by local classifiers, which are trained on subsets assigned by the partitioners. This algorithm is applied to new data by first assigning the data to a local classifier using the partitioners. Classification is then made by the selected local classifier.

Preliminary experiments show that this method can improve the performance for some classifiers. Nevertheless, further research including comparisons on other data sets is necessary. This is the focus of our current research as well as the implementation of the suggested variations for regression problems. One open topic for future research is the discussed combination of different classifiers for partitioners and local classifiers.

[7] It has to be remarked that for all methods the default settings were used. Thus, by tuning the parameters of each method some parameter dependent methods could certainly be improved compared to other methods. However, a comparison between base–classifier–types is not in the scope of this work.

ACKNOWLEDGEMENTS

We are grateful to the unknown referees for their helpful comments. Furthermore we would also like to thank the maintainers of the University of California Machine Learning Repository [2], and Jock A. Blackard, Dr. Denis J. Dean and Dr. Charles W. Anderson for providing the *CoverType* dataset to the repository.

REFERENCES

[1] Esteban Alfaro Cortes, Matias Gamez Martinez, and Noelia Garcia, ‘Multiclass corporate failure prediction by adaboost.m1’, *International Advances in Economic Research*, **13**(3), 301–312, (2007).

[2] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007.

[3] Jock A. Blackard and Denis J. Dean, ‘Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables’, *Computers and Electronics in Agriculture*, **24**(3), 131–151, (2000).

[4] Leo Breiman, ‘Bagging predictors’, Technical report, Statistics Department, University of California, Berkeley, (1994).

[5] Leo Breiman, ‘Bagging predictors’, *Machine Learning*, **24**(2), 123–140, (1996).

[6] Leo Breiman, ‘Arcing classifiers’, *The Annals of Statistics*, **26**(3), 801–849, (1998).

[7] Leo Breiman, Jerome H. Friedman, Richard Olshen, and Charles J. Stone, *Classification and regression trees*, Wadsworth, 1984.

[8] *Statistical Models in S*, eds., John M. Chambers and Trevor J. Hastie, Chapman and Hall, Ltd. and Wadsworth, 1993.

[9] Philip Chan and Salvatore J. Stolfo, ‘Learning arbiter and combiner trees from partitioned data for scaling machine learning’, in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 39–44, (1995).

[10] César Ferri, Peter A. Flach, and Jose Hernandez-Orallo, ‘Delegating classifiers’, *Proceedings of the 21st International Conference on Machine Learning*, (2004).

[11] Yoav Freund and Robert E. Schapire, ‘A decision-theoretic generalization of on-line learning and an application to boosting’, *Proceeding of the second European Conference on Computational Learning Theory*, 23–37, (1995). Extended version of the original, unpublished manuscript.

[12] Yoav Freund and Robert E. Schapire, ‘Experiments with a new boosting algorithm’, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148–156, (1996).

[13] Julio Ortega, Moshe Koppel, and Shlomo Argamon, ‘Arbitrating among competing classifiers using learned referees’, *Knowledge and Information Systems*, **3**(4), 470–490, (2001).

[14] Derek Partridge and William B. Yates, ‘Engineering multiversion neural-net systems’, *Neural Computation*, **8**(4), 869–93, (1995 (1996)).

[15] R Development Core Team, ‘R: A language and environment for statistical computing’, (2006).

[16] Robert Schapire, ‘The strength of weak learnability’, *Machine Learning*, **5**(2), 197–227, (1990).

[17] Alexander K. Seewald and Johannes Fürnkranz, ‘An evaluation of grading classifiers’, in *Advances in Intelligent Data Analysis: Proceedings of the 4th International Symposium (IDA-01). Lisbon, Portugal*, (2001).

[18] G. N. Wilkinson and C. E. Rogers, ‘Symbolic descriptions of factorial models for analysis of variance’, *Applied Statistics*, **22**, 392–399, (1973).

[19] David H. Wolpert, ‘Stacked generalization’, *Neural Networks*, **5**(2), 241–259, (1992).

[20] William B. Yates and Derek Partridge, ‘Use of methodological diversity to improve neural network generalisation’, *Neural Computing and Applications*, (1995).

Disturbing Neighbors Diversity for Decision Forests

Jesús Maudes and **Juan J. Rodríguez** and **César García-Osorio**[1]

Abstract. Ensemble methods take their output from a set of base predictors. The ensemble accuracy depends on two factors: the base classifiers accuracy and their diversity (how different are these base classifiers outputs from each other). An approach for increasing the diversity of the base classifiers is presented in this paper. The method builds some new features to be added to the base classifier training dataset. Those new features are computed (i) using the nearest neighbor instance from a very small previous randomly selected set and, (ii) the class this k-NN predicts for the instance. We tested this idea using decision trees as base classifiers. An experimental validation on 62 UCI datasets is provided for traditional ensemble methods, showing that ensemble accuracy and base classifiers diversity are usually improved.

1 INTRODUCTION

Ensembles are classifiers that combine predictions from some other classifiers. These combined classifiers in an ensemble are called base classifiers. Some ensemble combination schemas have became popular and they have proved to be successful. Many of them use a set of base classifiers computed each one using the same algorithm.

Ensembles overall accuracy requires base classifiers not to predict wrong the same instances. They need to be diverse in order to complement each other. So, how can a set of base classifiers generated from the same algorithm provide those different outputs from the same inputs? Diversity has been achieved on ensembles using different strategies, most of them are based on modifying the base classifiers training dataset.

In Bagging [4] diversity comes from picking randomly different instances for training each base classifier. The Random Subspaces method [11] chooses different subsets of attributes for training each base classifier. Random Forests [5] are a variant of Bagging, using Random Trees as base classifiers. In this type of random trees, the selection of the attribute for a decision node is done using only a random subset of the attributes.

Boosting [10] trains iteratively the set of base classifiers, modifying the weights of instances to train the current classifier. These new weights are computed from the training error on the previous base classifier, so each new base classifier becomes more specialized in instances that have been misclassified before.

Some of these methods are constrained to use an specific base classifier (e.g. Random Trees), and some others get diversity in a way that cannot be exported to other ensemble methods (e.g., Boosting re-weighting). One important advantage of our approach is that it can be applied to any base method within any ensemble. For example, in this work the experimental validation is focused mainly in ensembles of Decision Trees.

[1] University of Burgos, Spain, email: jmaudes@ubu.es, jjrodriguez@ubu.es, cgosorio@ubu.es

On the other hand, resampling in Bagging, and random feature selection in Random Subspaces, or Random Trees in Random Forests could be considered ways of getting diversity that can be adopted directly in other combination schema, and they can be adopted for different base methods. Diversity in these three last methods is acquired by adding some randomness to base classifiers training process (random resampling, random features selection or random trees). In this work we extend this family of methods with another approach to supply diversity. We think our method belongs to this family because:

1. Our method does not take into account the ensemble method in which it is going to be used (like it happens with Bagging, Random Subspaces and Random Forests). Moreover, we can apply our method to all the ensemble algorithms mentioned before, which have their own way of getting diversity, making their base classifiers become even more diverse, and improving the overall ensemble method performance.
2. It inserts a random element that makes the base classifiers to be built in a different way each time.

In order to insert randomness we use a weak classifier prediction. By weak classifier we mean a not very accurate classifier. This classifier is built each time with a very small subset of instances of the whole training set picked randomly. We have used a *K-Nearest Neighbor* classifier for this purpose. The k-NN output is used to build new features that *disturb* or alter the predictions of the base classifier respect when it is trained using the raw dataset. That is why we call our method *Disturbing Neighbors*.

The paper is organised as follows. Section 2 describes the Disturbing Neighbor method. Section 3 analyses our method applied to state of art representative ensembles of Decision Trees containing the experimental validation. Section 4 concludes.

2 METHOD

The method presented generates different base classifiers by adding new features to training data. This new features are different each time making the base classifiers set to be diverse.

Given a training dataset D, the method takes a number c as parameter. First, c instances from D are selected randomly to build a small 1-NN classifier. Then, for each training instance i in D, we add $c+1$ new features in the following way:

1. The class predicted by the 1-NN classifier is added as a new feature for all the instances in D.
2. Aditional c boolean features are added, one for each of the c selected instances. These features are all zero but the one corresponding with the nearest neihbour to instance i.

So a new dataset D' is obtained combining the original features plus the $c + 1$ features computed by the weak 1-NN classifier. The

D' dataset can now be used for training the next base classifier in any ensemble method, whatever the base classifier method is.

Note that computationally parallelizable ensemble methods (i.e. Bagging, Random Subspaces or Random Forests), keep this algorithmic property when Disturbing Neighbors are used. For our experiments we use $c = 10$, so computational cost does not grow significantly by using our variant in decision tree base classifiers.

3 RESULTS

Validation was made implementing Disturbing Neighbors in Java within WEKA [15]. We tested our method using the WEKA ensemble implementations. Default WEKA parameters are used unless otherwise indicated:

1. Bagging [4].
2. Random Forests [5].
3. Boosting: We use Adaboost [10] and Multiboost [14]. In both boosting versions we consider resampling and reweighting variants, which are respectively denoted as (S) or (W) in tables.
4. Random Subspaces [11]. We have tested two configurations, picking 50% and 75% of the original problem dimensions.

Fifty base classifiers were always used in each ensemble:

1. For Boosting, Bagging and Random Subspaces the base method was J.48 Decision Trees (WEKA implementation of Quinlan C4.5 Decision Tree [13]). We have tested the ensembles with plain J.48, and with J.48 disturbed by our method ($\mathcal{DN}$-Decision Trees). Each 1-NN in the $\mathcal{DN}$-Decision Tree takes always $c = 10$ random instances.
2. For Random Forests obviously Random Trees are used as base classifiers. The ensemble was also tested with plain Random Trees and with the disturbed variant ($\mathcal{DN}$-Random Trees).

We wanted to know if $\mathcal{DN}$-Decision Trees are base classifiers that perform well without any sophisticated combination schema. So we also tested fifty disturbed decision trees getting their final prediction as a straight average of the predictions generated by the individual base classifiers. We denote it by $\mathcal{DN}$-Ensemble from now on.

Finally, we wanted to know if k-NN accuracy was strong enough to be the main reason the disturbed classifiers could improve ensembles accuracy. So, we add IBk (WEKA implementation of k-NN [1]) to the test. We test using fixed $k = 1$ and variable k configurations. Variable k configuration optimizes k value for each data set. NN methods are very robust with respect to variations of data set, so they do not improve very much when combined with standard ensembles [9]. Thus, we have not considered ensembles of k-NN in our test. In particular, Bagging using 1-NN as base classifiers is equivalent to 1-NN [6]. Moreover, Bagging can slightly degrade the performance of stable algorithms (e.g., k-NN), [3].

For our validation we have used the 62 UCI datasets [2] at Table 1. 5x2 stratified cross validation was performed, which provides an acceptable number of repetitions (see [8]).

Results are summarized in tables 2, 3 and 4.

Table 2 shows the methods using the average ranks from [7]. A number is assigned to each method and data set corresponding to its rank position in such a dataset. If there were ties, average ranks are assigned. Then, for each method, the average position is calculated over all datasets (see values at in first column of Table 2). The methods are then ordered using these values.

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
lrd	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

Table 2. Ensemble methods sorted by their average rank.

Average Rank	Method
6.28	$\mathcal{DN}$-MultiBoost (S)
6.73	$\mathcal{DN}$-Random Forest
7.12	$\mathcal{DN}$-MultiBoost (W)
8.01	$\mathcal{DN}$-AdaBoost (S)
8.09	$\mathcal{DN}$-AdaBoost (W)
8.27	MultiBoost (S)
8.65	Random Forest
8.84	$\mathcal{DN}$-Subspaces (50%)
9.23	MultiBoost (W)
9.65	$\mathcal{DN}$-Bagging
10.03	$\mathcal{DN}$-Subspaces (75%)
10.11	AdaBoost (S)
10.23	AdaBoost (W)
11.54	k-Nearest Neighbor
11.90	Bagging
12.31	Subspaces (50%)
14.07	$\mathcal{DN}$-Ensemble
14.31	Subspaces (75%)
14.61	1-Nearest Neighbor

We can see that all ensemble methods were improved by their $\mathcal{DN}$-version. Relative order between such methods using undisturbed decision trees is preserved when $\mathcal{DN}$-version methods are compared on their own. So $\mathcal{DN}$-versions improvements seem to be slightly independent of combination schemas. That is why we think that the ensemble method itself is more important for accuracy than using or not a $\mathcal{DN}$-version of base classifier. This hypothesis is supported by pour ranking of $\mathcal{DN}$-Ensemble. The average ranking of this ensemble is even worse than k-NN, and it shows that simply using $\mathcal{DN}$-base classifiers does not ensure to get the best possible ensemble.

Improvement by using $\mathcal{DN}$-versions is quantified in Table 3 that shows wins, ties and loses of $\mathcal{DN}$-ensemble versions against undisturbed versions. According to the sign test [7], for 62 data sets one method is significantly better than other if the number of wins (plus half the ties) is greater or equal than 39. Hence, for all the methods in Table 3 the $\mathcal{DN}$version is significantly better.

1-NN and k-NN seem to rank poorly in Table 2. Thus we can think that $\mathcal{DN}$-versions are not improved by k-NN algorithm strength, but the diversity induced by the random neighbors selection. Table 4 also shows this issue comparing each $\mathcal{DN}$ensemble against k-NN. We can see that k-NN is significantly worst than all the $\mathcal{DN}$-methods except $\mathcal{DN}$-Ensemble and $\mathcal{DN}$-Subspaces (75%).

Table 3. Comparison of methods with and without $\mathcal{DN}$-based diversity.

Method	Win-Tie-Loss
Bagging	50-1-11
Subspaces (50%)	50-3-9
Subspaces (75%)	54-3-5
AdaBoost (W)	47-0-15
AdaBoost (S)	49-1-12
MultiBoost (W)	47-1-14
MultiBoost (S)	48-0-14
Random Forest	40-3-19

We also tested diversity improvement of $\mathcal{DN}$-Trees using Kappa statistic [12]. Kappa measures how diverse two classifiers are. It can takes values ranged from -1 to 1. Kappa equal to 1 means that both classifiers agree in every example. Kappa equal to 0 means the

Table 4. Comparison of ensemble methods with the k-Nearest Neighbor classifier.

Method	Win-Tie-Loss
$\mathcal{DN}$-Ensemble	26-2-34
$\mathcal{DN}$-Bagging	41-0-21
$\mathcal{DN}$-Subspaces (50%)	40-1-21
$\mathcal{DN}$-Subspaces (75%)	36-2-24
$\mathcal{DN}$-AdaBoost (W)	39-0-23
$\mathcal{DN}$-AdaBoost (S)	39-1-22
$\mathcal{DN}$-MultiBoost (W)	39-0-23
$\mathcal{DN}$-MultiBoost (S)	40-1-21
$\mathcal{DN}$-Random Forest	44-2-16

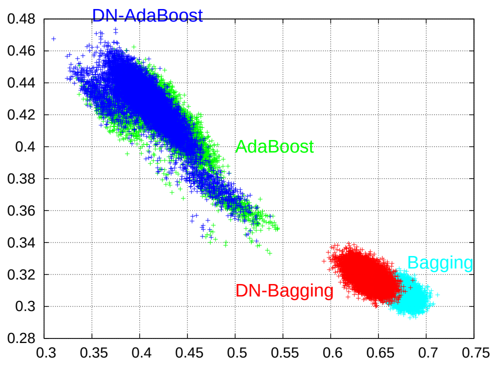


Figure 1. Error vs. Kappa for Boosting and Bagging in *krk* dataset

agreement is the expected if both classifiers output random predictions, and negative Kappa values show they are more diverse than the agreement expected by chance, which rarely happens. Then Kappa values are used to draw Kappa-Error Diagrams [12]. Figure 1 shows an example for *krk* dataset with Bagging and Boosting methods. For each pair of base classifiers we plot a point (x, y), where x is kappa measure for these two classifiers, and y is the average error of them. So we want ideally pairs of base classifiers next to left bottom corner, because it means they are accurate and diverse.

In Figure 1 we see $\mathcal{DN}$-clouds a little bit left than undisturbed ensembles clouds. It means that $\mathcal{DN}$-methods are more diverse. Each diagram in Figure 2 is based on the corresponding Kappa-Error diagram for each dataset. All diagrams are scaled using the maximum and minimum values of Kappa and Error. Each method considered (i.e., Bagging, both tested variants of *sampled* Boosting, Random Forests and Random Subspaces 50% attributes variant) is represented by an arrow pointing from the centre of the not $\mathcal{DN}$-version cloud to the centre of the $\mathcal{DN}$-version cloud of each method. We can see a lot of arrows pointing left, which means a generalized improvement of diversity. The longer the arrow, the bigger the relative difference.

Finally, Figure 3 shows a Kappa-Error diagram for each ensemble method, translating the starting point of every arrow in Figure 2 to coordinates origin. This kind of diagram is a very convenient way of summing up the results shown in Figure 2. The majority of arrows point to left, which is an indicator of diversity. Many arrows also point up showing that generally, increase of diversity is at the expense of individual base classifiers accuracy degradation.

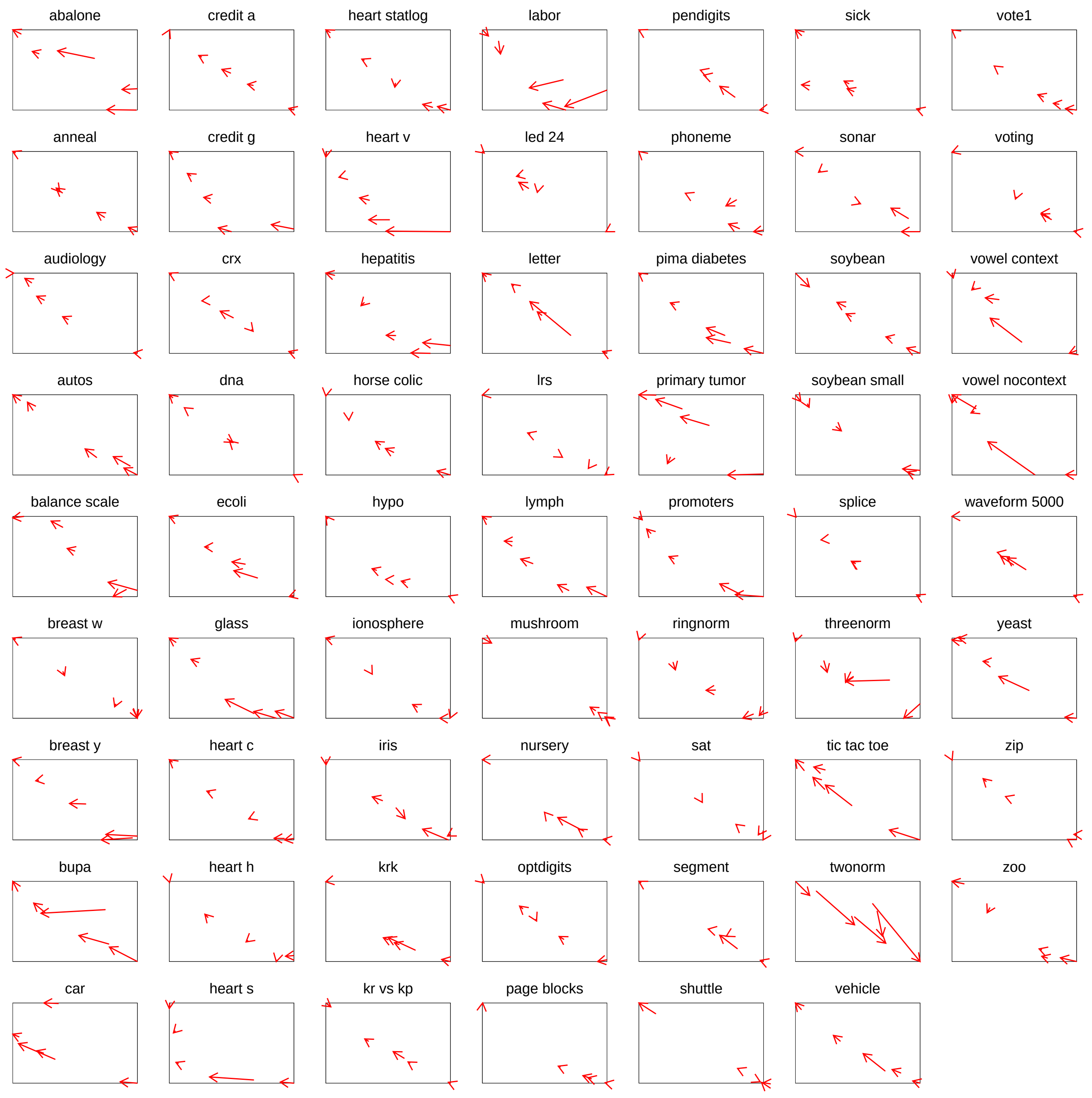


Figure 2. Kappa Error diagram for the 62 datasets. Each arrow points from the centre of the not $\mathcal{DN}$-version cloud of a method to the centre of the $\mathcal{DN}$-version cloud

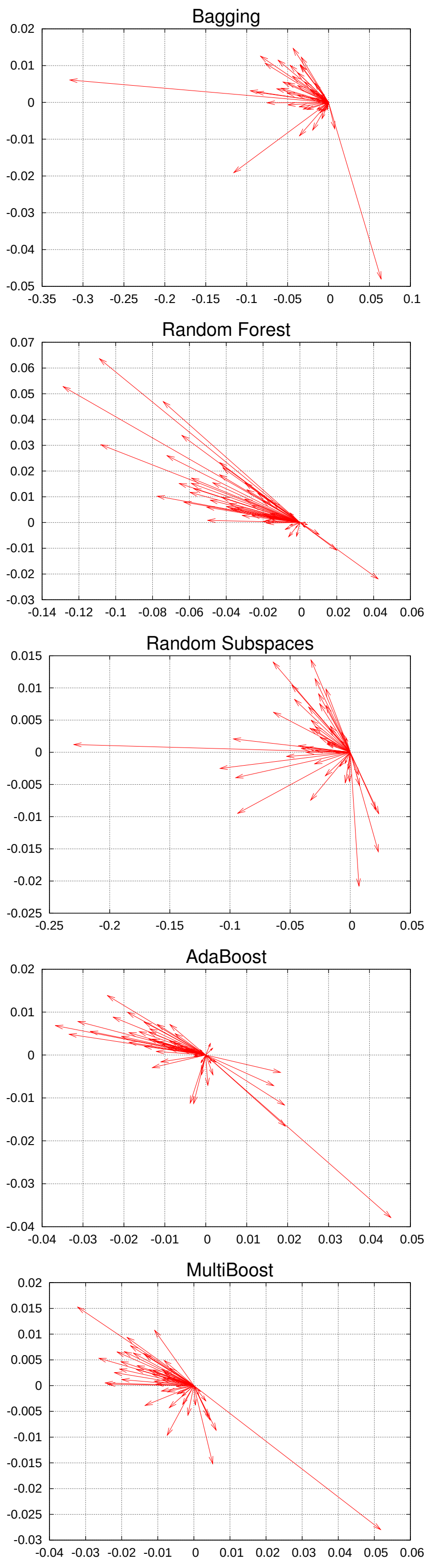


Figure 3. Kappa-Error diagram for each ensemble method. There is an arrow for each dataset pointing the increment of these variables by using $\mathcal{DN}$ versions

4 CONCLUSION

In this paper a method for improving diversity of ensemble base classifiers is presented. Our method builds some new features to be added to the training set. These attributes are different for each base classifier, making them diverse. The new features are computed from a weak 1-NN classifier that disturbs the normal base classifier training (*Disturbing Neighbor*). The weakness of this new base classifier is achieved using a very small subset of the whole training dataset as 1-NN training dataset. That 1-NN classifier training instances are selected randomly, so that randomness is in the end our source of diversity.

An experimental validation has been provided showing that the idea presented in this paper uses to improves accuracy in all considered ensembles of decision tress. Diagrams based on Kappa statistic have shown that diversity is also improved.

Acknowledgements

This work has been partially supported by the Spanish MCyT project DPI2005–08498.

REFERENCES

[1] D. Aha and D. Kibler, 'Instance-based learning algorithms', *Machine Learning*, **6**, 37–66, (1991).
[2] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
[3] Eric Bauer and Ron Kohavi, 'An empirical comparison of voting classification algorithms: Bagging, boosting, and variants', *Machine Learning*, **36**(1-2), 105–139, (1999).
[4] Leo Breiman, 'Bagging predictors', *Machine Learning*, **24**(2), 123–140, (1996).
[5] Leo Breiman, 'Random forests', *Machine Learning*, **45**(1), 5–32, (2001).
[6] Bruno Caprile, Stefano Merler, Cesare Furlanello, and Giuseppe Jurman, 'Exact bagging with k-nearest neighbour classifiers', in *Multiple Classifier Systems*, pp. 72–81, (2004).
[7] J. Demšar, 'Statistical comparisons of classifiers over multiple data sets', *Journal of Machine Learning Research*, **7**, 1–30, (2006).
[8] Thomas G. Dietterich, 'Approximate statistical test for comparing supervised classification learning algorithms', *Neural Computation*, **10**(7), 1895–1923, (1998).
[9] Carlotta Domeniconi and Bojun Yan, 'Nearest neighbor ensemble', in *ICPR (1)*, pp. 228–231, (2004).
[10] Yoav Freund and Robert E. Schapire, 'Experiments with a new boosting algorithm', in *Thirteenth International Conference on Machine Learning*, pp. 148–156, San Francisco, (1996). Morgan Kaufmann.
[11] Tin Kam Ho, 'The random subspace method for constructing decision forests', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(8), 832–844, (1998).
[12] D. D. Margineantu and T. G. Dietterich, 'Pruning adaptive boosting', in *Proc. 14th International Conference on Machine Learning*, pp. 211–218. Morgan Kaufmann, (1997).
[13] J. R. Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann, 1993.
[14] Geoffrey I. Webb, 'Multiboosting: A technique for combining boosting and wagging', *Machine Learning*, **Vol.40**(No.2), (2000).
[15] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2nd edn., 2005. `http://www.cs.waikato.ac.nz/ml/weka`.

Integrating Feature Selection and Committee Training

Erinija Pranckeviciene [1]

Abstract. The purpose of this paper is to discuss a computational paradigm of integration of feature selection into a committee training. The computational paradigm is based on external cross-validation with an inner loop. We show experimentally on the independent data the benefits and cavets of such paradigm. Feature profiles, useful for interpretation, are generated during the training. Our integrated approach addresses the question of how to construct a committee in a high-dimensional classification problem.

1 INTRODUCTION

Ensemble methods are the viable choices in many real life classification problems, for review see [1]. They proved to be successful for feature selection in high-dimensional difficult data [2]. Multiple classifier systems using feature selection algorithms in handwritten character recognition domain were investigated in [3]. Some practical classification problems require the interpretable features more and emphasize the performance of the classifier system less [4]. In the deficit of the data, the profiles of the features accumulated during ensemble learning provide the evidence of the features, important for class separation [5]. Though the committee of classifiers usually improves the performance, in practice, one has to make the informed choices of the system design, such as: what base classifier to use; how many experts; how to combine and validate. The choices are determined either by prior knowledge or customizing the ensemble's configuration rather intuitively by using the available "off the shelf" software.

In this study we analyze the computational paradigm of training the ensemble via feature selection. We experiment with the two class datasets of NIPS2003 Feature Selection (NIPS2003 FS) challenge (see [6], [2] and www.nipsfsc.ecs.soton.ac.uk for more information). The data represents two class real life problems - mass spectra classification (ARCENE dataset) and handwritten digit recognition (GISETTE dataset), and the artificial problem (MADELON dataset). NIPS2003 FS benchmark datasets are augmented with probes- the fake features. Ground truth about the feature identity: useful or probe is available, enabling to test the feature selection methods. All datasets are high-dimensional, their properties are summarized in Table 1. In Agnostic Learning versus Prior Knowledge (ALvsPK)challenge [9], the ensemble of linear discriminants with integrated Liknon feature selection [7] appeared among the top-ranked methods.

The scheme of training the committee by feature selection is described in Section 2. Experimental results on real datasets are discussed in Sections 3 and 4. Section 5 summarizes the study.

[1] Kaunas University of Technology, Lithuania, email: erinija.pranckeviciene@ktu.lt

2 TRAINING THE COMMITTEE: COMPUTATIONAL PARADIGM

The committee training and simultaneous feature selection is performed in external k-fold crossvalidation with the inner loop. The number of folds k, usually 5 or 10, is conditioned by the sample size of the training data. The data subdivision in the outer loop is into **`Training`** set and **`Test`** set. During the development, the **`Training`** set is randomly partitioned several times into two: a balanced $\mathtt{Training}$ set and the remaining $\mathtt{Monitoring}$ set in the inner loop. The classifier is trained on the $\mathtt{Training}$ set. Similar approaches are discussed in [10]. An optimal feature subset is identified in every data partition/split by a wrapper. The optimality criterion is Balanced Error Rate (BER) of the classifier on the $\mathtt{Monitoring}$ set. BER is computed from the confusion matrix $confmat$ of the classifier:

$$confmat = \begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix},$$
$$BER = \frac{1}{2}\left(\frac{FP}{(FP+TP)} + \frac{FN}{(FN+TN)}\right) \ , \qquad (1)$$

where TP is the true positive, FP is false positive, FN is false negative and TN is true negative. The $\mathtt{Monitoring}$ set is used to monitor an over-fitting of the feature selection procedure. Feature subset producing the minimum monitoring BER of the particular data split is selected. Monitoring takes care of feature selection bias [11]. Figure 1 shows the general scheme of the procedure. The union of the optimal

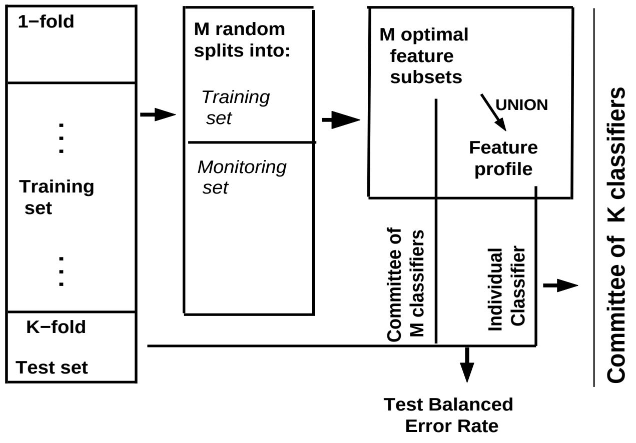


Figure 1. Scheme of training the committee via feature selection.

subsets makes a feature profile, which is input to another classifier (the choice of classifier in based on the knowledge about the problem). The trained classifier is assessed using the fold **`Test`** set. The individual classifiers of the folds are combined into the committee- the final model for classification of the future data. Motivations for the outlined approach are: (i) processing many splits in the inner loop generates different feature subsets and maintains the diversity of the ensemble [12]; (ii) information, pertinent to classification in various data "projections" of feature and sample spaces [13], is aggregated. We determine the base classifier, combination method and wrapper feature selection method by using available knowledge about the problem. If prior knowledge is not available, then selection of a base classifier and combination method from the pool of candidates is guided by the **`Test`** BER. In this study we assess the committee's performance independently on Validation and Test sets of NIPS2003 FS challenge.

3 EXPERIMENTAL SETUP

It is know that ARCENE, GISETTE and MADELON datasets have a nonlinear character. Based on this knowledge, three nearest neighbor classifier (NN3) was chosen as a base, and voting as a combination method. The simple rules prevent from tuning of additional parameters. We used Liknon [14] embedded feature selection method as the wrapper, because of it's success in ALvsPK challenge in the agnostic track as a "black box". Liknon is trained using a balanced training set. The details can be found in [7]. Liknon returns features, optimal for linear separation. In order to reduce the data dimensionality we also perform a univariate feature pre-filtering, using the difference between classes as a criterion:

$$a = (\sum_{i=1}^{N_1} x_i - \sum_{j=1}^{N_2} x_j) \ . \tag{2}$$

The indices i and j denote the samples of the class 1 and class 2 respectively, N_1 and N_2 are sample sizes of the classes. The variables x_i and x_j represent the values of the individual features. Using (2), the features are ranked by decreasing value of a. A percentage of low-a value features is discarded, taking into account the feature to sample-size ratio. It was shown, that simple feature pre-filtering was very effective in building good classifiers on NIPS2003 FS data [6]. The sizes of the data subdivisions in our experiment are summarized in Table 1. The **`Test`** set size of two classes is denoted as **Te1+Te2**, the balanced *`Training`* as *Tr1+Tr2* and remaining *`Monitoring`* as *Mo1+Mo2*. The number of splits M and folds K in the computational experiment equals to 5. The sample sizes of the **i**ndependent Test and Validation data sets of NIPS2003 FS benchmark are denoted as Ti1+Ti2 and Vi1+Vi2 and the dimensionality of the dataset by D.

Table 1. Parameters of the experiment.

Parameters/Dataset	ARCENE	GISETTE	MADELON
D	10000	5000	500
Tr1+Tr2	30+30	150+150	300+300
Mo1+Mo2	5+15	2250+2250	500+500
Te1+Te2	9+11	600+600	200+200
Vi1+Vi2	44+56	500+500	300+300
Ti1+Ti2	310+390	3250+3250	900+900
Discarded %	85	95	98
M #splits	5	5	5
K #experts	5	5	5

In every split we do feature pre-filtering followed by the selection of the optimal feature subset by Liknon. The accumulated feature profile, a union of M feature subsets, is input to NN3 individual classifier. The procedure is repeated in every fold. As a result we obtain K individual classifiers in the committee, operating by voting. We estimate BER's of the committee and individual classifiers on the independent Test and Validation sets of the NIPS2003 FS benchmark. NN3 trained on all training data and all useful features is our baseline - what a single NN3 can achieve with all good features. Normally, in real life problem, such baseline can not be computed, because we do not have information about the feature identity. Note, that different classifiers perform differently on the same data, based on the capability of the classifier to handle the complexity of the data. Large NIPS2003 FS Test sets allow faithful estimate of the classification performance. For computational experiments we used functions of PRTools [8]. We investigate whether such scheme produces a committee, with improved performance as compared to the individual classifiers.

4 RESULTS

The baseline and committee performances on the independent Test and Validation datasets are presented in Table 2. For all datasets the committee on the Validation set works similar as on the Test set. It is close to the NN3 baseline performance. We also present the best NIPS2003 FS benchmark entries [6]. ARCENE is the most challenging dataset, because it's dimensionality exceeds the training sample size by orders of magnitude. New-Bayes-nn-red+v method, submitted by Radford Neal, is the best entry for the ARCENE dataset. The description of their method follows. *Bayesian neural network with two hidden layers (20 and 8 units), applied to a set of features selected by examining the ARD hyperparameters found in New-Bayes-lr-sel and New-Bayes-nn-sel. An ARD prior was used here as well to allow some of the features to have more influence than others. The model was fitted to both the training and validation data.* It is a complex learning strategy using both Training and Validation sets. The success of New-Bayes-nn-red+v indicates the high complexity of ARCENE. NN3 might not be capable of handling such complexity. The number of features in their method is 100, no probes. This number is less than the total number of samples (88+112) in Training and Validation sets together. New-Bayes-nn-red+v method for other datasets achieved the following results: Test BER of GISETTE was 0.0186, (326 features / 1 probe) and Test BER of MADELON was 0.0994, (21 feature / 4 probes). The performance of New-Bayes-nn-red+v on MADELON and GISETTE is still comparable with the results of NN3 committee. NIPS2003 FS winners on MADELON and GISSETTE datasets used a combination of the CLOP models trained on both Training and Validation sets. More information can be found in [6].

Figure 2 shows the Validation and Test BER's distribution for ARCENE and MADELON. BER's of the individual classifiers in folds are indicated by stars. The number next to each star indicates **`Test`** BER of the fold. Diamond represents NN3 baseline. Square shows the committee's performance. There is a big variance of Validation and Test BER estimates of the individual NN3. The variance arises due to the different training datasets in folds and different accumulated feature profiles, indicating the diversity of the individual classifiers.

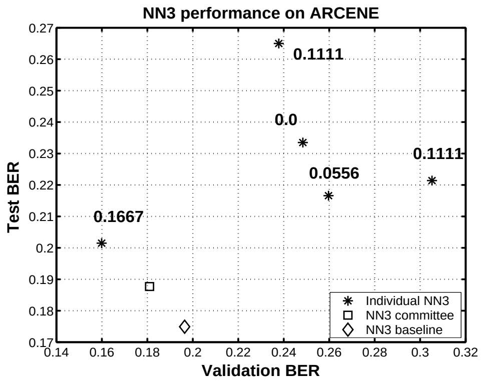
There is a disagreement between the Balanced Error Rates in folds and independent Validation and Test BER's, see Table 3 and Table 4. The disagreement may arise due to sample sizes and inadequate feature to sample size ratio (FSR) in training and validation. In ARCENE, the classifier with the zero BER on the **`Test`** set has

Table 2. Comparison of Balanced Error Rates.

Dataset	Validation BER / Test BER (#Features / #Probes) NNA3 Committee	NN3 Baseline	Best entry
ARCENE	0.1810 / 0.1877 (345 / 63)	0.1964 / 0.1749 (7000 / 0)	0 / 0.0720 (100 / 0) Radford Neal
GISETTE	0.0420 / 0.0292 (357 / 3)	0.0320 / 0.0258 (2500 / 0)	0.0080 / 0.0086 (NA) Theodor Mader
MADELON	0.1183 / 0.1144 (21 / 7)	0.1067 / 0.0994 (20 / 0)	0.0200 / 0.0622 (12 / 0) Shen Kaiquan

independent Validation BER 0.2484 and Test BER 0.2335. Distribution of BERs in MADELON shows a lesser disagreement. FSR in MADELON is much lower than in ARCENE.

For all datasets the feature profile contains a fraction of probes. This might be a reason why the committee performs slightly worse on the Test set than the baseline. Feature selection and classifier learning are intrinsically combined. In wrapper, we select features, optimal for the chosen particular classifier. Advanced general feature selection strategies such as Floating Forward Feture Selection (FFFS) [15] with monitoring option are very time consuming and inefficient in high-dimensional problems, such as ARCENE, GISETTE and MADELON. After pre-filtering, in this study, Liknon was able to identify more useful features than probes. In Liknon training we used small sample size setting, thus larger number of samples appeared in *Monitoring* set for GISETTE and MADELON. **Test** BER of the individual experts may indicate the better feature profiles. Number of features, number of probes and their percentage in the features profiles, the BER of **Test**, Validation and Test sets for individual experts in MADELON and ARCENE, are summarized in Tables 3 and 4.



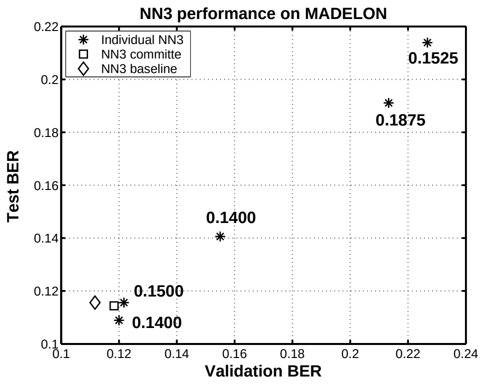


Figure 2. Performance of K individual NN3 classifiers and their committee on the independent Test and Validation sets of NIPS2003 FS benchmark.

Table 3. MADELON. Individual NN3 experts.

Fold	Valid BER	Test BER	**Test** BER	#feature/#probe (%)
1	0.2133	0.1911	0.1875	15/4 (26.67%)
2	0.1550	0.1406	0.1400	10/0 (0%)
3	0.2267	0.2139	0.1525	6/0 (0%)
4	0.1200	0.1089	0.1400	12/1 (8.33%)
5	0.1217	0.1156	0.1500	15/2 (13.33%)

Table 4. ARCENE. Individual NN3 experts.

Fold	Valid BER	Test BER	**Test** BER	#feature/#probe (%)
1	0.2597	0.2166	0.0556	103/20 (19.42%)
2	0.1599	0.2015	0.1667	99/14 (14.14%)
3	0.2484	0.2335	0.0	91/15 (16.48%)
4	0.3052	0.2214	0.1111	103/20 (19.42%)
5	0.2378	0.2650	0.1111	95/36 (37.89%)

In Table 3 the expert of fold 1 is worst. It's feature profile contains 11 useful features and 4 probes. This is the largest number of probes in all folds. The best expert is in fold 4, trained on 11 useful features and 1 probe. In MADELON, the Validation and Test BER estimates are consistent. The small values of either Test or Validation BER reveal better feature profiles. In ARCENE (see Table 4) the Test BER shows that the expert in fold 5 is worst. It is characterized by the largest number of probes. If we look at the Validation BER, the conclusion is different- the expert in fold 5 is the second best. The BER estimate on the **Test** set ranks the expert in fold 5 the same as expert 4, as the third. ARCENE had very small *Training Monitoring* and **Test** set, thus the estimates of BERs are not consistent. The estimates, computed with small sample size, have high variance. The **Test** BER can be used to rank the feature profiles **provided that fold Test set contains enough samples**. This is important for exploratory data analysis and interpretation. There are efforts directed towards determination of "enough samples" for training [17] and testing [16] in specific applications.

5 CONCLUSIONS

We report an early experiment of integrating wrapper feature selection into committee training. We have applied it to high-dimensional datasets of the NIPS2003 FS benchmark. All results reported here are available on the web site www.nipsfsc.ecs.soton.ac.uk. Our investigation was possible, because of the independent platform for testing the feature selection methods. In some real problems, neither information about feature identity, nor large test sets are available. Especially in biomedical domains, the data resembles ARCENE. The following insights were gained:

1. Our result suggests that the committee, trained using the proposed paradigm, integrating a feature selection, will have an improved performance as compared to the individual experts. At least for

the data types similar to the NIPS2003 FS benchmark datasets. Feature profiles, identified simultaneously are important for interpretation and understanding of the expert's performance.

2. The investigated computational procedure generates diverse experts, evidenced by the large variance of BERs estimated on the independent Validation and Test sets. The experts are competent in a particular feature-sample subspaces.

3. The **`Test`** BER in folds can be used for the ranking of the feature profiles, provided there is enough samples for training and testing. The estimates, computed with very small number of samples, are misleading. As a possible solution is a control of Feature to Sample size Ratio (FSR). It can be carried out by the univariate feature filtering, as in this study. The smaller FSR, the more confident we can be about the performance estimates of the experts.

4. The wrapper in the inner loop, the base expert/classifier and a combination method can be selected by using the knowledge about the problem. In the present study we used knowledge that the datasets have a nonlinear character and the fact that Liknon performed well as a "black box" in ALvsPK challenge. Liknon wrapper generates a feature profile, which is optimal for linear separation. We use this feature profile as an input to another expert, realizing a nonlinear rule. We selected NN3 as our base classifier in order to address the nonlinearity of the data, keeping the overall architecture simple. We showed, that the performance of such combination is comparable with the performance of the other methods on the benchmark datasets.

Our study addressed practical issues, arising in integrating feature selection and committee training in high-dimensional classification problems. As a computational paradigm we suggest K-fold external cross-validation with the inner loop. In our study we have chosen simple architecture of the committee. Indeed, the neural net, or SVM with nonlinear kernel function are among the possible choices for the base expert. However, complex classifiers would involve an additional tuning of the parameters, requiring additional validation. In some real life problems we can't afford more validation because of sparse data. Thus we suggest using simple models initially in the first stage of the data analysis. It would be appropriate to estimate the data complexity and find a matching classifier. Such unified methodology doesn't exist yet. There are efforts directed towards development of such methodology [18], also in problems characterized by a deficit of training data [19].

REFERENCES

[1] R. Polikar, Ensemble based systems in decision making, *IEEE Circuits and systems magazine*, **3**, 21–45, (2006).

[2] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature extraction, foundations and applications*, Physica-Verlag, Springer, 2006.

[3] S. Gunter, H. Bunke, Creation of classifier ensembles for handwritten word recognition using feature selection algorithms, 183–188, *Proc. Eight International Workshop: Frontiers in Handwritten Recognition*, IEEE, (2002).

[4] E. Pranckeviciene, R. Baumgartner, R. Somorjai, Using domain knowledge in the random subspace method. Application: Application to the classification of magnetic resonance spectra, 336–345, *Multiple Classifier Systems MCS2005, LNCS*, **3541**, Springer Berlin/Heidelberg, 2005.

[5] A. Bamgbade, R. Somorjai, B. Dolenko and et al., Evidence accumulation to identify discriminatory signatures in biomedical spectra, 463–467, *Artificial Intelligence in Medicine AIM2005, LNCS* **3581**, Springer Berlin/Heidelberg, 2005.

[6] I.Guyon, J.Li, T.Mader, P.A.Pletsher, G.Schneider, and M.Uhr, Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark, *Pattern recognition letters*, **28**, 1438–1444, (2007).

[7] E.Pranckeviciene, R.Somorjai, M.N.Tran, Feature/model selection by the Linear Programming SVM combined with state-of-art classifiers: what can we learn about the data, *Proccedings of International Joint Conference on Neural Networks IJCNN2007*, INNS/IEEE, Orlando, Florida, 1627–1632, (2007).

[8] R.P.W.Duin, P.Juszczak, P.Paclik, E.Pekalska, D.de Ridder, and D.M.J.Tax. *PRTools4 A Matlab toolbox for pattern recognition www.prtools.org*, February, 2004.

[9] I.Guyon, A.Safari, G.Dror, and G.Cawley, Agnostic learning vs. prior knowledge challenge, *Proccedings of International Joint Conference on Neural Networks IJCNN2007*, INNS/IEEE, Orlando Florida, 829–834, (2007).

[10] N.V. Chawla, T.E. Moore, L.O. Hall et al., Distributed learning with bagging-like performance, *Pattern recognition letters*, **24(1-3)** , 455–471, (2003);

[11] C. Ambroise, G.J. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, *PNAS*, **99(10)**, 62–6566, (2002).

[12] P. Cunningham, J. Carney, Diversity versus quality in classification ensembles based on feature selection, 1611-3349, , *Machine Learning: ECML2000, LNCS*, **1810**, Springer Berlin/Heidelberg, 2000.

[13] T.K. Ho, The random subspace method for constructing decision forests, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **20(8)**, 832–844, (1998).

[14] C.Bhattacharyya, L.R.Grate, A.Rizki, D. Radisky, F.J. Molina, M.I. Jordan, M.J. Bissell and I.S. Mian. Simultaneous relevant feature identification and classification in high-dimensional spaces: application to molecular profiling data. *Signal Processing*, **83(4)**, 729–743, (2003).

[15] P. Pudil, J. Novovicova, J. Kittler, Floating search methods in feature selection, *Pattern Recognition Letters*, **12(3)**, 1119–1125, (1994).

[16] I. Guyon, J. Makhoul, R. Schwartz, V. Vapnik, What size test set gives good error rate estimates?. *IEEE Trans. PAMI*, **20(1)**, 52–64, (1998).

[17] S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, C. Campbell, T.R. Golub, J.P. Mesirov, Estimating dataset size requirements for classifying DNA microarray data, *Journal of Computational Biology*, **10(2)**, 119–142, (2003).

[18] E.B. Mansilla, T.K. Ho, On classifier domains of competence, *Proccedings of the 17th International Conference on Pattern Recognition ICPR2004*, Vol. 1, 136–139, (2004).

[19] E. Pranckeviciene, T.K. Ho, R. Somorjai, Class separability in spaces reduced by feature selection, *Proccedings of the 18th International Conference on Pattern Recognition ICPR2006*, Vol. 3, 254–257, (2006).