A DIVIDE-AND-CONQUER METHOD FOR MULTI-NET CLASSIFIERS

D.S. Frosyniotis, A. Stafylopatis, A. Likas

31 – 2001

Preprint, no 31 – 01 / 2001

Department of Computer Science
University of Ioannina
45110 Ioannina, Greece

A Divide-and-Conquer Method for Multi-Net Classifiers

Dimitrios S. Frosyniotis, Andreas Stafylopatis

National Technical University of Athens

Department of Electrical and Computer Engineering

Zographou 157 73, Athens, Greece

e-mail:dfros@cslab.ntua.gr, andreas@cs.ntua.gr

Aristidis Likas

University of Ioannina

Department of Computer Science

451 10 Ioannina, Greece

e-mail:arly@cs.uoi.gr

A Divide-and-Conquer Method for Multi-Net Classifiers

Abstract

Several researchers have shown that substantial improvements can be achieved in difficult pattern recognition problems by combining the outputs of multiple neural networks. In this work, we present and test a pattern classification multi-net system based on both supervised and unsupervised learning. Following the *"divide-and-conquer"* framework, the input space is partitioned into overlapping subspaces and neural networks are subsequently used to solve the respective classification subtasks. Finally, the outputs of individual classifiers are appropriately combined to obtain the final classification decision. Two clustering methods have been applied for input space partitioning and two schemes have been considered for combining the outputs of the multiple classifiers. Experiments on well-known data sets indicate that the multi-net classification system exhibits promising performance compared with the case of single network training, both in terms of error rates and in terms of training speed (especially if the training of the classifiers is done in parallel).

Keywords: Divide-and-Conquer; Multiple Classifier Systems; Classifier Combination; Classifier Fusion; Clustering.

1 Introduction

Several paradigms for multi-classifier systems have been proposed in the literature during the last years. Classifier combination approaches can be divided along several dimensions, such as the representational methodology, the use of learning techniques or the architectural methodology [1, 2]. A major issue in the architectural design of multiple classifier systems concerns whether individual learners are *correlated* or *independent*. The first alternative is usually applied to multistage approaches (such as boosting techniques [3, 4]), whereby specialized classifiers are serially constructed to deal with data points missclassified in previous stages. The second alternative advocates the idea of using a committee of classifiers which are trained independently (in parallel) on the available training patterns, and combining their decisions to produce the final decision of the system. The latter combination can be based on two general strategies, namely *selection* or *fusion*. In the case of selection, one or more classifiers are nominated "local experts" in some region of the feature space (which is appropriately divided into regions), based on their classification "expertise" in that region [5], whereas fusion assumes that all classifiers have equal expertise over the whole feature space. A variety of techniques have been applied to implement classifier fusion by combining the outputs of multiple classifiers [1, 6, 7, 8].

The methods that have been proposed for combining neural network classifiers can provide solutions to tasks which either cannot be solved by a single net, or which can be more effectively solved by a multi-net system. However, the amount of possible improvement through such combination techniques is generally not known. Sharkey [9], and Tumer and Ghosh [10, 11] outline

a mathematical and theoretical framework for the relationship between the correlation among individual classifiers and the reduction in error, when an averaging combiner is used.

When multiple independent classifiers are considered, several strategies can be adopted regarding the generation of appropriate training sets. The whole set can be used by all classifiers [2, 12] or multiple versions can be formed as bootstrap replicates [13]. Another approach is to partition the training set into smaller disjoint subsets but with proportional distribution of examples of each class [12, 14].

The present work introduces a different approach for building a multi-net classifier system. The proposed method is based on partitioning the original data set into subsets using unsupervised learning techniques (clustering) and the subsequent use of individual classifiers for solving the respective learning subtasks. A key feature of the method is that the training subsets represent non-disjoint regions that result from input-space clustering. This partitioning approach produces a set of correlated "specialized" classifiers which attack a complex classification problem by applying the devide-and-conquer philosophy.

Thus, instead of training a single neural network involving a lot of parameters and using the entire training set, neural networks with less parameters are trained on smaller subsets. Through the splitting of the original data, storage and computation requirements are significantly reduced.

The general principle of divide-and-conquer is the basis of several learning approaches, such as the Hierarchical Mixtures of Experts of Jordan and Jacobs [15] and related tree-structured models [16]. The idea of partitioning

the input-space through clustering has been applied to building classifier selection models [5], where classifiers are trained on the same training set and are subsequently assigned to different disjoint regions according to their accuracy. In the approach proposed here, classifiers are assigned to overlapping regions from the beginning and acquire their specialization through training with data sets that are representative of the regions.

In the next section, we address the issue of data partitioning based on unsupervised learning techniques. Sections 3 and 4 describe the training of classifiers and techniques for combining classifier outputs. Experimental results for the evaluation of the proposed method are presented in Section 5. In Section 6 the present work is compared with related work in the literature and, finally, conclusions are presented in Section 7.

2 Partitioning of the Data Set

Consider a classification problem with c classes and a training set D having N supervised pairs $(\vec{x}^i, k^i)$ where $\vec{x}^i \in R^l$ and k^i is an integer indicating the class of the pattern $\vec{x}^i$. The first stage of the proposed classification technique consists of partitioning the original data set $D = \{\vec{x}^1, \ldots, \vec{x}^N\}$ using clustering techniques to identify natural groupings. As a result of clustering, the set D is partitioned into a number M of subsets $D_1, D_2, \ldots, D_M$ as shown in Fig. 1.

We considered two clustering techniques that are based on completely different principles. The first method is fuzzy clustering whereas the second method is based on probability density estimation using Gaussian mixtures. Both techniques allow for the specification of the degree with which a data

point belongs to each cluster, i.e., the data subsets obtained from the clustering stage are not disjoint. This fact provides the flexibility to define a *clustering threshold* q that determines the degree of cluster overlapping. More specifically, a pattern is considered to belong to a given cluster if the membership degree of the pattern to that cluster exceeds the value of the factor q. Experimentally, the best clustering threshold was found to be $q = 1/M$, where M is the number of clusters. An exception is the case $M = 2$ where, experimentally, the best threshold was found to be $q = 0.3$. The two clustering techniques tested in this work are briefly described in the following subsections.

2.1 Fuzzy C-Means Clustering

Fuzzy C-means (FCM) [17] is a data clustering technique in which a data sample belongs to all clusters with a membership degree. FCM partitions the data set into M fuzzy clusters (where M is specified in advance), and provides the center of each cluster. Clustering is usually based on the Euclidean distance:

$$d^2(\vec{x}, \vec{\mu}) = \sum_{j=1}^{l} (x_j - \mu_j)^2 \tag{1}$$

where $\vec{x} \in R^l$ is a training sample and $\vec{\mu} \in R^l$ corresponds to a cluster center. The FCM algorithm provides fuzzy partitioning, so that a given data point $\vec{x}$ belongs to cluster j (with center $\vec{\mu}_j$) with membership degree u_j varying between 0 and 1:

$$u_j = \frac{1}{\sum_{k=1}^{M} \frac{d(\vec{x}, \vec{\mu}_j)}{d(\vec{x}, \vec{\mu}_k)}}, \; j = 1, \ldots, M \tag{2}$$

The membership degrees are normalized in the sense that, for every pattern,

$$\sum_{j=1}^{M} u_j = 1, \tag{3}$$

Starting from arbitrary initial positions for cluster centers, and by iteratively updating cluster centers and membership degrees using Eq. (1) and (2) for each training point $\vec{x}^i, i = 1, \ldots, N$, the algorithm moves the cluster centers to sensible locations within the data set. This iteration is based on minimizing an objective function J that represents the distance from any given data point to a cluster center weighted by the data point's membership degree.

$$J(\vec{\mu}_1, \ldots, \vec{\mu}_M) = \sum_{i=1}^{N} \sum_{j=1}^{M} (u_{ij}^m d^2(\vec{x}^i, \vec{\mu}_j)), \tag{4}$$

where $m \in [1, \infty)$ is a weighting exponent.

The main drawbacks of this algorithm is that its performance depends on the initial cluster centers and that the number of clusters is predefined by the user. Therefore, it is required to run the FCM algorithm several times, each time with a different number of clusters to discover the number of clusters that results in best performance of the classification system. Fig. 2 displays the result of the fuzzy C-means clustering method with the well-known Clouds data set considering three clusters. The clustering threshold q is set 0.333. So, in this example, the data point $\vec{x}$ belongs to the jth cluster (and to the subset D_j), if $u_j \geqslant 0.333$.

In Fig. 2, the three cluster centers are presented with big circles and the patterns of each cluster are presented with crosses, circles and stars respectively. We can also observe a degree of overlapping between clusters, as some patterns belong to two or three clusters simultaneously. Thus, the data sets

created with the fuzzy C-means algorithm are not disjoint. The correlation between the data sets has a beneficial impact increasing the robustness of the multi-net classification system.

2.2 Greedy-EM algorithm for Gaussian mixtures

We have also considered a different technique for partitioning the training set that is based on probability density function (*pdf*) estimation using Gaussian mixtures. According to this approach, the data are assumed to be generated by several parametrized Gaussian distributions, so the data points are assigned to different clusters based on their posterior probabilities of having been generated by a specific Gaussian distribution. A multivariate Gaussian mixture is defined as the weighted sum:

$$p(\vec{x}) = \sum_{j=1}^{M} \pi_j f(\vec{x}; \vec{\phi}_j) \tag{5}$$

where π_j are the mixing weights satisfying $\sum_j \pi_j = 1,\ \pi_j \geqslant 0$, and $f(\vec{x}; \vec{\phi}_j)$ is the l-dimensional Gaussian density

$$f(\vec{x}; \Phi_j) = (2\pi)^{-l/2} \mid S_j \mid^{-1/2} exp[-0.5(\vec{x} - \vec{m}_j)^\top S_j^{-1}(\vec{x} - \vec{m}_j)] \tag{6}$$

parametrized on the mean $\vec{m}_j$ and the covariance matrix S_j, collectively denoted by the parameter vector $\vec{\phi}_j$. Usually, for a given number M of kernels, the specification of the parameters of the mixture is based on the *expectation-minimization* algorithm (EM) [18] for maximization of the data log-likelihood:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \log p(\vec{x}^i) \tag{7}$$

The iterative EM update equations for each kernel j, $j = 1, \ldots, M$, are the following:

$$P(j|\vec{x}^i) = \frac{\pi_j f(\vec{x}^i; \vec{\phi}_j)}{p(\vec{x}^i)} \tag{8}$$

$$\pi_j := \frac{1}{N} \sum_{i=1}^{N} P(j|\vec{x}^i) \tag{9}$$

$$\vec{m}_j := \frac{\sum_{i=1}^{N} P(j|\vec{x}^i)\vec{x}^i}{\sum_{i=1}^{N} P(j|\vec{x}^i)} \tag{10}$$

$$S_j := \frac{\sum_{i=1}^{N} P(j|\vec{x}^i)(\vec{x}^i - \vec{m}_j)(\vec{x}^i - \vec{m}_j)^\top}{\sum_{i=1}^{N} P(j|\vec{x}^i)} \tag{11}$$

In this work we have used the recently proposed *greedy EM* algorithm [19], which is an incremental algorithm that has been found to provide better results than the conventional EM algorithm. This algorithm starts with one kernel and adds kernels dynamically one at a time. Assuming at some point of the algorithm k kernels, regular EM steps are carried out until convergence, and then a new kernel is added to the mixture in a specific way. To locate the optimal position of the new kernel two types of search are employed: (i) efficient global search among all input points, followed by (ii) local search based on partial EM steps for fine tuning of the parameters of the new kernel. Simulation results have shown that the greedy-EM algorithm (running until M kernels have been added) seems to outperform EM (with M kernels). Moreover, it is possible to estimate the true number of components of the mixture as follows: We run the algorithm for a large final value of $M_{\max}$ and for the solution obtained for each intermediate value of M we apply a model selection criterion e.g., cross-validation using a set of test points, a coding scheme based on minimum description length etc. Then we finally select the optimal value of M that corresponds to the optimal value of the

model selection criterion. In this work we have used as a criterion for the specification of M, the log-likelihood value on a validation set of points that have not been used for training.

After the number of kernels and the mixture parameters are specified, we can compute the posterior probability $P(j \mid \vec{x}^i)$ that a pattern $\vec{x}^i$ has been generated from kernel j, according to Eq. (8). Therefore, we can consider that each kernel corresponds to a group (cluster) of patterns and that $P(j \mid \vec{x}^i)$ corresponds to the membership degree u_{ij} of pattern $\vec{x}^i$ to the jth group. In analogy with Eq. (3), it holds that:

$$\sum_{j=1}^{M} P(j \mid \vec{x}^i) = 1, \forall i = 1, \ldots, N \tag{12}$$

Fig. 3 and 4 display the partitioning of the Clouds data set using the greedy-EM algorithm. The optimal number of kernels was found to be $M = 4$. In Fig. 3, we can see the means m and variances s^2 for the kernels, whereas in Fig. 4 we present the patterns corresponding to each kernel (crosses, circles, stars and diamonds) respectively. Since $q = 1/M = 0.25$, the ith data point belongs to the jth kernel if $P(j \mid \vec{x}^i) \geqslant 0.25, \quad \forall\ i = 1, \ldots, N$ and $j = 1, 2, 3, 4$. We can observe some overlapping between clusters, since some patterns belong to two, three or four clusters simultaneously.

3 Training the individual classifiers

In what concerns the classification module, the primary idea is to train a neural classifier, in particular a multi-layered perceptron (MLP), for each group of patterns D_j generated through the partitioning of the original data set D (see Fig.5). In this sense, each classifier learns a subspace of the prob-

lem domain and becomes a "local expert" for the corresponding subdomain. An important advantage of this method is that the training of each subnetwork can be done separately and in parallel. Thus, in the case of parallel implementation, the total training time of the system equals to the worst training time achieved among the neural classifiers. It must be noted that this total training time cannot be greater than the training time of a single neural classifier of the same type dealing with the entire training set. Since such a single network usually requires more parameters, to learn the whole data set (which is much larger), the multi-net approach may lead to reduced execution times even in the case of implementation on a single processor. This case actually occurred in our experimental study, when during data partitioning we came up with clusters of a single class. Thus, there was no need of training a classifier for these clusters, leading to reduced training time compared with the case of single network training.

In this work, each classifier is a fully connected multilayer perceptron (MLP), with one hidden layer of sigmoidal units. We have applied the BFGS quasi-Newton algorithm [20] to train the MLPs using the early stopping technique. Assuming that we have created M groups of patterns after partitioning the original data set, we divide the available training data of each group D_j into two parts, a training set and a validation set for "early stopping". Therefore, we train M MLPs using different (but possibly overlapping) training and validation sets. The classifications produced by the multiple individual MLPs are combined following different formulas as discussed next.

4 Techniques for combining classifications

As described above, the original data set D is partitioned into M subsets and M classifiers are trained, one for each subset. Given a new pattern $\vec{x}$ we obtain the output $\vec{y}_j$, $j = 1, \ldots, M$, produced by each MLP j. We also compute the membership degrees u_j of pattern $\vec{x}$ in group D_j. We considered two different ways to combine the outputs of the MLPs in order to obtain the final classification decision.

4.1 Weighted sum of outputs

Let $\vec{x}$ be an input vector which belongs to one of c classes. Every MLP has c continuous outputs corresponding to each of the c classes. More specifically, for each MLP j, $j = 1, \ldots, M$, the corresponding output vector is

$$\vec{y}_j = ({y_1}^j, \ldots, {y_c}^j) \tag{13}$$

To obtain the combined final output, the weighted sum of the output vectors $\vec{y}_j$ of the MLPs is computed, with the weights corresponding to the membership degree u_j of pattern $\vec{x}$ belonging to group j.

$$y_k = {y_k}^1 \cdot u_1 + \ldots + {y_k}^M \cdot u_M \tag{14}$$

for $k = 1, \ldots, c$. The final decision C of the classification system for input $\vec{x}$ is:

$$C = \arg\max(y_1, \ldots, y_c) \tag{15}$$

Eq. (15) provides a final estimate for the class of an input vector $\vec{x}$, where all classifiers participate in the decision in a fuzzy manner.

4.2 Class probabilities

Consider an input vector $\vec{x}$ which belongs to one of c classes. As with the weighted sum scheme, we compute the output vector $\vec{y}_j$ for every MLP j along with the corresponding membership degree u_j of the input vector $\vec{x}$ for group D_j. A usual approach to obtain the classification of $\vec{x}$ is to compute the probability $P(k \mid \vec{x})$ $(k = 1, \ldots, c)$ that pattern $\vec{x}$ belongs to class k and select the class C with the maximum $P(C \mid \vec{x})$ as the final decision following the Bayes rule.

The probability $P(k \mid \vec{x})$ can be computed as follows: First we select the maximum component C_j of the output vector y_j to obtain the class label suggested by each MLP j. Then, we can use the following formula:

$$P(k \mid \vec{x}) = \sum_{j=1}^{M} u_j I(C_j = k) \tag{16}$$

where $I(z)$ is an indicator function, i.e. $I(z) = 1$ if z =true, otherwise $I(z) = 0$. The above equation states that the class probability $P(k \mid \vec{x})$ results as the sum of the weights u_j of the classifiers that suggest class k. It is easy to check that $\sum_{k=1}^{c} P(k \mid \vec{x}) = 1$. It must be noted that the combination method (16) is more general since it considers the class label suggested by each classifier and not the numerical output vectors as in the weighted sum case. Consequently, the method can also be used with other types of classifiers, eg. decision trees or SVMs.

5 Experimental results

In this section, we present comparative performance results from the use of the proposed classification system using the FCM and greedy-EM algorithms

for data partitioning and the previously described schemes for combining classifier outputs. For comparison, we also present results from individual MLPs with different numbers of hidden units. Five well-known data sets were used in our experiments, as shown in Table 1.

In all the tests presented here, each experiment was run ten times and the *min*, *mean* and *max* errors were calculated from these ten trials. For each experiment, each subnetwork was trained several times with different numbers of sigmoidal hidden units (5, 8, 10, 15, 20) and different initialisations. The best outcome of the trials according to the validation error was used when testing the combination schemes.

The classification performance of the system was evaluated for the following cases of data clustering and classifier combination:

- FCM algorithm and weighted sum (FCM WS),
- FCM algorithm and class probabilities (FCM CP),
- Greedy-EM algorithm and weighted sum (G-EM WS),
- Greedy-EM algorithm and class probabilities (G-EM CP).

5.1 The Clouds data

The Clouds artificial data from the ELENA project [21] are two-dimensional with two a priori equally probable classes. There are 5000 examples in the data set, 2500 in each class (50%). The theoretical error is 9.66%.

In our experiments, we used 2000 patterns for training, 2000 for validation and 1000 patterns for testing the system, respectively. For the FCM algorithm, we obtained the best results by splitting the original data set into

three subsets ($q = 0.333$). The Greedy-EM algorithm divided the Clouds data set into four subsets ($q = 0.25$). The testing results for the multi-net classification system are shown in Table 2.

The classification error obtained with the multi-net system is quite close to the theoretical one; therefore, any further improvement can hardly be achieved.

5.2 The Diabetes data

The Diabetes set from the UCI data set repository [22] contains 8-dimensional data belonging to two classes. It is based on personal data from 768 Pima Indians obtained by the National Institute of Diabetes and Digestive and Kidney Diseases. The diagnostic binary-valued variable that is investigated indicates whether the patient shows signs of diabetes according to World Health Organization criteria.

In our experiments, we used 400 patterns for training, 200 for validation and 168 patterns for testing the system. For the FCM algorithm, we obtained the best results by splitting the original data set into three subsets ($q = 0.333$). The Greedy-EM algorithm divided the Diabetes data set into three subsets ($q = 0.333$). The testing results for the multi-net classification system are shown in Table 3.

It must also be noted that this data set contains some known outliers, that affect the construction of the clusters and eventually the classification performance of the system.

5.3 The Image Segmentation data

The Image Segmentation data set from the UCI data set repository [22] contains 19-dimensional examples belonging to 7 classes. There are 2310 instances drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel. Each patterm corresponds to a 3x3 region.

We used 1000 patterns for training, 500 for validation and 810 patterns for testing the system. For the FCM algorithm, we obtained the best results by splitting the original data set into four subsets ($q = 0.25$). The Greedy-EM algorithm divided the Segmentation data set into five subsets ($q = 0.2$). The testing results for the multi-net classification system are shown in Table 4.

In our experiments, we preprocessed the Image Segmentation data set by applying a principal component analysis (PCA). In addition, the size of the input vectors was reduced to a 7-dimensional space by retaining only those components which contribute more than a specified fraction (defined 0.009) of the total variation in the data set.

5.4 The Phoneme data

The aim of using the Phoneme dataset from the ELENA project [21] is to distinguish between nasal and oral vowels, hence, there are two classes: the *nasals* and the *orals*. The Phoneme dataset contains vowels originating from 1809 isolated syllables. Five features were chosen to characterise each vowel. The features are the amplitudes of the first five harmonics normalised by the total energy integrated over all the frequencies. There are 3818 patterns

from the first class and 1586 patterns from the second class.

In our experiments, we used 2500 patterns for training, 2000 for validation and 904 patterns for testing the system. For the FCM algorithm, we obtained the best results by splitting the original data set into two subsets ($q = 0.3$). The Greedy-EM algorithm divided the Phoneme data set into six subsets ($q = 0.17$). The testing results for the multi-net classification system are shown in Table 5.

5.5 The Breast Cancer data

This is data provided to the UCI repository [22] by Dr. William H. Wolberg [23] from the University of Wisconsin Hospitals, Madison. Data contain 699 patterns belonging to two classes (458 benign and 241 malignant). It involves 9 variables representing cellular characteristics.

In our experiments, we used 350 patterns for training, 150 for validation and 199 for testing the system. For the FCM algorithm, we obtained the best results by splitting the original data set into three subsets ($q = 0.333$). The Greedy-EM algorithm divided the Breast Cancer data set into five subsets ($q = 0.2$). The testing results for the multi-net classification system are shown in Table 6.

5.6 Discussion

An important conclusion that can be drawn from the experimental results is that, as expected, the multi-net system almost always exhibits better performance than a single neural classifier. For the Clouds, Segmentation and Phoneme data sets the gain in performance is quite significant. With FCM

partitioning we achieved better classification error in Diabetes, Segmentation and Breast Cancer data sets, while the partitioning based on the Greedy-EM algorithm led to better performance in Clouds and Phoneme data. However, the Greedy-EM algorithm dynamically determines the number of clusters during data partitioning and this makes it superior to FCM algorithm which assumes a predefined number of clusters. An important result that occasionally came up during data partitioning in our study, was the creation of clusters with examples of a single class. Thus, there was no need of training a classifier for these clusters. Another interesting conclusion is that the weighted-sum combination scheme produces slightly better results than the class probabilities formula, in all cases except for the Clouds and Breast Cancer data sets.

The obtained results show that the proposed multi-net classification system outperforms several methods reported in the literature, like Bagging or Boosting, for the Diabetes and Breast Cancer data sets [3, 4, 13] as well as for the Segmentation data set [4]. Also, for the Phoneme data set, the performance of our approach was better in comparison with the Clustering, Selection and Decision Templates combination scheme [2, 5]. For the Diabetes data set, it yielded superior results in comparison with classifier combining through trimmed means and order statistics [8]. Finally, the proposed multi-net classification system performed better for the Clouds data set in comparison with a variety of soft combinations of multiple classifiers like Majority, Averaging, Weighted averaging, Borda count etc [6]. It should be noted, however, that, since the partitioning of the data may or may not be the same as in our case, this comparison should be considered as rather

indicative.

6 Related work

We have presented a multi-net classification method that implements the "divide-and-conquer" problem solving paradigm through the appropriate combination of unsupervised and supervised learning schemes. First, the input space is partitioned into overlapping subspaces through clustering. Then neural networks are subsequenlty used to solve the respective classification subtasks. Finally, the outputs of the individual classifiers are appropriately combined to obtain the final classification decision. A key feature of the method is that the training subsets correspond to non-disjoint regions of the input space. Thus, the partitioning procedure produces a set of correlated specialized classifiers which cooperate at the decision level in order to tackle a complex classification problem. In this way, smaller networks may be employed that can be trained in parallel on smaller and usually easier to discriminate training sets. Through the splitting of the original data storage and computation requirements are significantly reduced.

The multi-net methodology described here is related to a number of divide-and-conquer approaches in machine learning and neural networks. In general these approaches can be classified into two main categories. The first category contains the methods that follow the "divide-and-conquer" philosophy and can be considered that exhibit analogy with the proposed technique. The second category contains multi-net methods (like bagging, boosting etc) which do not perform input space partitioning. Instead, they consider that each individual classifier is trained using data points from the

whole input space. It is clear, that methods of this category exhibit only marginal relevance with our technique and cannot be considered to follow the same principles.

The main multi-net method that belongs to the first category is the Hierarchical Mixtures of Experts (HME) method of Jordan and Jacobs [15]. It produces a tree-structured model which partitions the data into different regions. Such decomposition ensures that the errors made by the expert nets will not be correlated, as they each deal with different data points. Expert nets learn to specialize in sub-tasks and to cooperate by means of a gating net. Our approach differs from HME and related architectures, in that the mixtures-of-experts model makes the assumption that a single expert is responsible for each example, whereas our combination scheme makes no such mutual exclusivity assumption, and each data point is likely to be dealt with by all component nets in the multi-net system. Since classifiers in the present approach are assigned to overlapping regions and acquire their specialization through training with data sets that are representative of these regions, this approach is appropriate when no model is highly likely to be correct for any point in the input space.

In the second category of multi-net techiques, the most popular methods for creating ensembles are Bagging [13] and Boosting [4]. These methods rely on "resampling" techniques from the *whole input space* to obtain different training sets for each of the classifiers. Thus, it is not possible to use smaller networks and accelerate the training of each classifier. In this group of methods, the individual classifiers and their combination are trained together. On average, Adaboost is better than bagging, but the main problem

with boosting seems to be robustness to noise [3, 4]. In parallel environments, bagging and the method proposed here have a strong advantage because sub-classifiers can be built in parallel. Our approach to designing a classifier combination is to use already trained classifiers (typically a smaller number of them compared to boosting and bagging) and combine their outputs. Simple combination methods, such as weighted sum (WS) and class probabilities (CP), do not even require further training or optimizing beyond training the individual classifiers. Also another multi-net techique of the second category concerns Kuncheva's clustering, selection and decision templates model [2, 5]. In that model, selection is applied in regions of the feature space where one classifier strongly dominates the others from the pool (called clustering-and-selection, CS) and fusion is applied in the remaining regions; decision templates (DT) are adopted for classifier fusion. Two main futures distiguish the proposed method from Kuncheva's work and other related selection-fusion architectures [12, 14]: (i) we apply clustering techniques (unsupervised learning) to partition the input space and (ii) the training subsets resulting from this partition represent non-disjoint regions. We believe that both of these features can play a role in increasing the benefits of combining and improving the robustness of the classification system by producing diverse component networks.

7 Conclusions

In this work we have elaborated on a multi-net classification system that is based both on unsupervised and supervised learning methods. To build the classification system, first the original training set is divided into overlapping

subsets by applying clustering techniques (unsupervised learning). Then, an individual MLP is trained on every defined subset. To obtain the classification of a new pattern, the outputs of the MLPs are appropriately combined using several combination schemes. An important strength of the proposed classification approach is that it does not depend on the type of the neural network, therefore, it is quite general and applicable to a wide class of models including neural networks and any other classification technique. The learning method offers the advantages of the *"divide-and-conquer"* framework, ie., smaller networks may be employed that can be trained in parallel on smaller (and usually easier to discriminate) training sets.

We have considered two algorithms for data clustering. The first is the fuzzy C-means method which creates fuzzy partitioning of the input domain, while the second approach is based on pdf estimation using mixture models, so patterns are assigned to different clusters based on their posterior probabilities. We have considered two schemes for combining the outputs of multiple neural classifiers: the first uses a simple weighted sum, while the second one is based on a probabilistic interpretation. The resulting classification approaches have been tested on different benchmark data sets exhibiting very promising performance.

The multi-net methodology implemented in this work is quite general. There is ample room for the implementation and testing of other techniques both in the clustering and the classification module. It must be noted that we are particularly interested in testing the performance of the classification system when support vector machines (SVM) [24], [25] are employed in the place of MLPs and this constitutes our primary direction for future study.

References

[1] Alpaydin E. Techniques for combining multiple learners. In *Proceedings of Engineering of Intelligent Systems*, volume 2, pages 6–12. ICSC Press, 1998.

[2] Kuncheva L. Combining classifiers by clustering, selection and decision templates. Technical report, University of Wales, UK, 2000.

[3] Maclin R. and Opitz D. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 546–551. AAAI Press/MIT Press, 1997.

[4] Freund Y. and Schapire R.E. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[5] Kuncheva L. Clustering-and-selection model for classifier combination. In *Proceedings of the 4th International Conference on Knowledge-based Intelligent Engineering Systems (KES'2000)*, Brighton, UK, 2000.

[6] Vericas A, Lipnickas A, Malmqvist K, Bacauskiene M, and Gelzinis A. Soft combination of neural classifiers: A comparative study. *Pattern Recognition Letters*, 20:429–444, 1999.

[7] Tumer K. and Ghosh J. Classifier combining through trimmed means and order statistics. In *Proceedings of the International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998.

[8] Tumer K. and Ghosh J. Order statistics combiners for neural classifiers. In *Proceedings of the World Congress on Neural Networks*, pages I:31–34, Washington D.C., 1995. INNS Press.

[9] Sharkey A.J.C. *Combining Artificial Neural Nets : Ensemble and Modular Multi-Net Systems*. Springer-Verlag Press, 1999.

[10] Tumer K. and Ghosh J. Limits to performance gains in combined neural classifiers. In *Proceedings of the Artificial Neural Networks in Engineering '95*, pages 419–424, St. Louis, 1995.

[11] Tumer K. and Ghosh J. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3-4):385–404, 1996.

[12] Alpaydin E. Voting over multiple condensed nearest neighbor subsets. *Artificial Intelligence Review*, 11:115–132, 1997.

[13] Breiman L. Bagging predictors. Technical Report 421, Department of Statistics, University of California, Berkeley, 1994.

[14] Chan P.K. and Stolfo S.J. A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Machine Learning Conference*, pages 90–98, San Mateo,CA, 1995. Morgan Kaufmann.

[15] Jordan M.I. and Jacobs R.A. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, (6):181–214, 1994.

[16] Chen K, Yu X, and Chi H. Combining linear discriminant functions with neural networks for supervised learning. *Neural computing and applications*, 6(1):19–41, 1997.

[17] Bezdek J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[18] Dempster A.P, Laird N.M, and Rubin D.B. Maximum likelihood from incomplete data via the EM algorithm. *Roy. Statist. Soc. B*, 39:1–38, 1977.

[19] Vlassis N. and Likas A. A greedy-EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, to appear.

[20] Dennis J.E. and Schnabel R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[21] ESPRIT Basic Research Project ELENA (no. 6891). [ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases], 1995.

[22] UCI Machine Learning Databases Repository, University of California-Irvine, Department of Information and Computer Science. [ftp://ftp.ics.edu/pub/machine-learning-databases].

[23] Wolberg W.H. and Mangasarian O.L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, pages 9193–9196, U.S.A., December 1990.

[24] Vapnik V. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[25] Burges C.J.C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

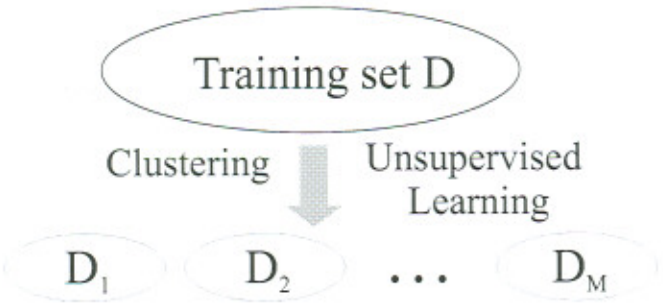

Figure 1: Partitioning of the training set D into M subsets using clustering methods.

Dataset	Cases	Classes	Features	
			Continuous	Discrete
Clouds	5000	2	2	-
Diabetes	768	2	9	-
Segmentation	2310	7	19	-
Phoneme	5404	2	5	-
Breast cancer	699	2	-	9

Table 1: Summary of the data sets.

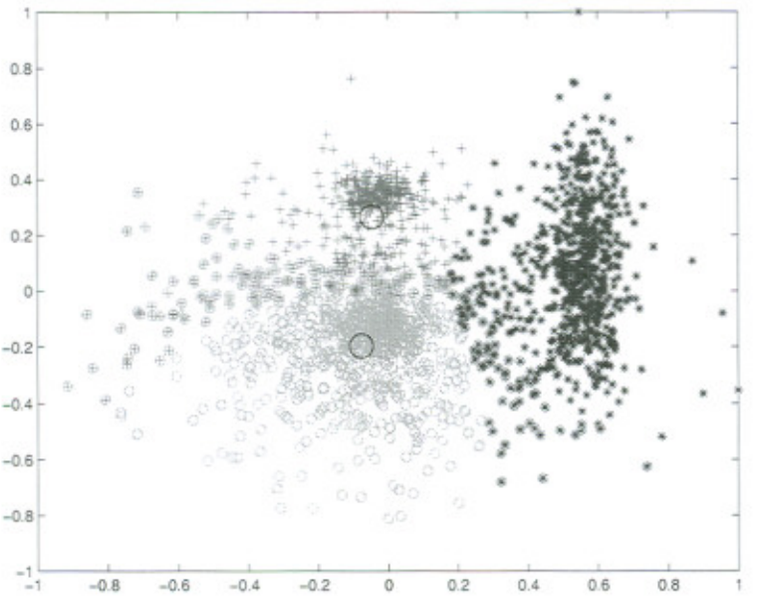

Figure 2: Three-cluster partition of the Clouds data set using the fuzzy C-means algorithm.

Clouds data set			
Classifier	min	mean	max
FCM WS	9.8	**11.26**	12.2
FCM CP	9.7	**11.1**	11.8
G-EM WS	10.1	**11.03**	11.8
G-EM CP	9.8	**10.68**	11.5
MLP(15 hidden units)	10.1	**12.13**	13.7
MLP(20 hidden units)	10.6	**12.48**	16.9
MLP(25 hidden units)	10.8	**11.91**	17.4

Table 2: The Clouds data set: Test set error (%) comparative results.

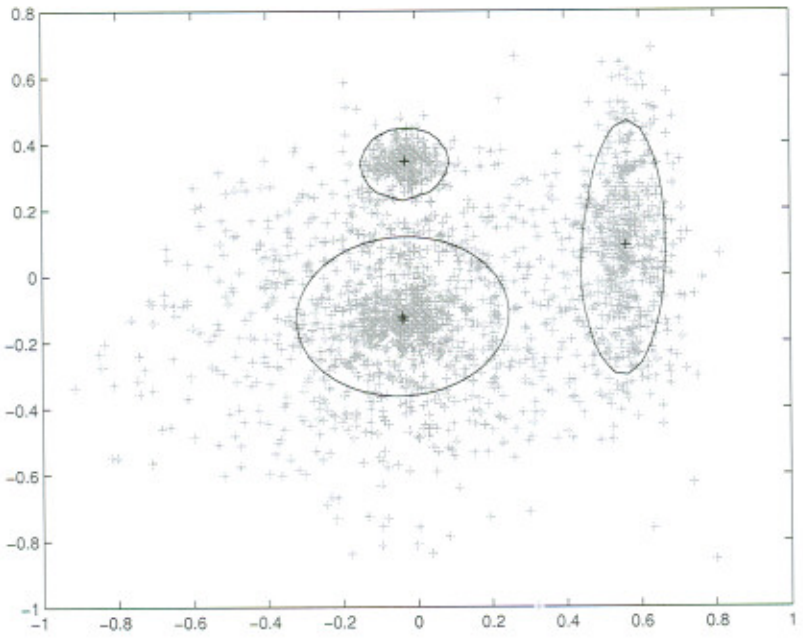

Figure 3: Means and variances of the kernels using the greedy-EM algorithm for the Clouds data set.

Diabetes data set			
Classifier	min	mean	max
FCM WS	19.05	**21.95**	25
FCM CP	20.83	**23.69**	26.79
G-EM WS	18.45	**22.74**	26.19
G-EM CP	19.05	**22.56**	25.6
MLP(15 hidden units)	19.64	**22.97**	25.6
MLP(20 hidden units)	19.05	**22.56**	25
MLP(25 hidden units)	17.86	**23.15**	26.79

Table 3: The Diabetes data set: Test set error (%) comparative results.

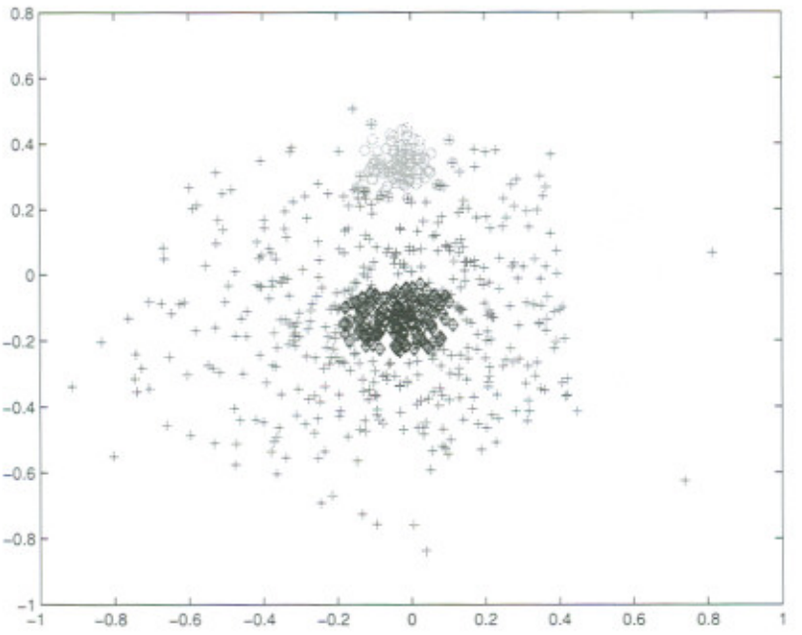

Figure 4: Four-cluster partition of the Clouds data set using the greedy-EM algorithm.

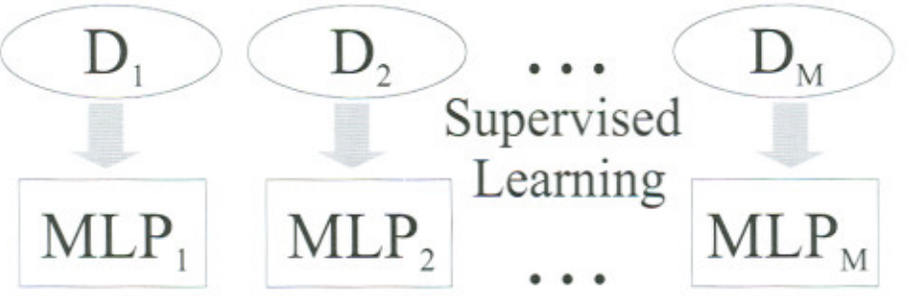

Figure 5: Training a different MLP for each group of patterns.

Segmentation data set			
Classifier	min	mean	max
FCM WS	15.8	**17.79**	19.01
FCM CP	16.54	**18.44**	21.73
G-EM WS	16.3	**21.67**	25.43
G-EM CP	16.17	**21.79**	25.56
MLP(30 hidden units)	33.46	**42.28**	59.51
MLP(35 hidden units)	16.3	**38.25**	55.31
MLP(40 hidden units)	29.51	**54.74**	69.38

Table 4: The Segmentation data set: Test set error (%) comparative results.

Phoneme data set			
Classifier	min	mean	max
FCM WS	15.82	**17.2**	18.69
FCM CP	14.27	**16.91**	18.47
G-EM WS	14.38	**15.47**	16.48
G-EM CP	14.82	**15.85**	16.7
MLP(15 hidden units)	15.49	**19.2**	21.46
MLP(20 hidden units)	15.49	**17.69**	20.13
MLP(25 hidden units)	13.83	**17.24**	19.25

Table 5: The Phoneme data set: Test set error (%) comparative results.

Breast Cancer data set			
Classifier	min	mean	max
FCM WS	2.01	**2.97**	3.52
FCM CP	2.01	**2.8**	3.52
G-EM WS	1.51	**3.67**	5.03
G-EM CP	1.51	**3.62**	5.03
MLP(15 hidden units)	1.01	**3.49**	4.52
MLP(20 hidden units)	1.01	**3.12**	4.02
MLP(25 hidden units)	2.01	**3.92**	5.03

Table 6: The Breast Cancer data set: Test set error (%) comparative results.