Journal of Machine Learning Research 8 (2007) 1-33 Submitted 8/05; Revised 7/06; Published 1/07

Nonlinear Boosting Projections for Ensemble Construction

Nicolás García-Pedrajas NPEDRAJAS@UCO.ES
Department of Computing and Numerical Analysis
University of Córdoba
Córdoba, Spain

César García-Osorio CGOSORIO@UBU.ES
Department of Civil Engineering
University of Burgos
Burgos, Spain

Colin Fyfe COLIN.FYFE@PAISLEY.AC.UK
School of Computing
University of Paisley
Paisley, United Kingdom

Editor: Dale Schuurmans

Abstract

In this paper we propose a novel approach for ensemble construction based on the use of nonlinear projections to achieve both accuracy and diversity of individual classifiers. The proposed approach combines the philosophy of boosting, putting more effort on difficult instances, with the basis of the random subspace method. Our main contribution is that instead of using a random subspace, we construct a projection taking into account the instances which have posed most difficulties to previous classifiers. In this way, consecutive nonlinear projections are created by a neural network trained using only incorrectly classified instances. The feature subspace induced by the hidden layer of this network is used as the input space to a new classifier. The method is compared with bagging and boosting techniques, showing an improved performance on a large set of 44 problems from the UCI Machine Learning Repository. An additional study showed that the proposed approach is less sensitive to noise in the data than boosting methods.

Keywords: classifier ensembles, boosting, neural networks, nonlinear projections

1. Introduction

An ensemble of classifiers consists of a combination of different classifiers, homogeneous or heterogeneous, to jointly perform a classification task. Ensemble construction is one of the fields of Artificial Intelligence that is receiving most research attention, mainly due to the significant performance improvements over single classifiers that have been reported with ensemble methods (Breiman, 1996a; Kohavi and Kunz, 1997; Bauer and Kohavi, 1999; Webb, 2000; García-Pedrajas et al., 2005).

A classification problem of K classes and n training observations consists of a set of instances whose class membership is known. Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)\}$ be a set of n training samples where each instance $\mathbf{x}_i$ belongs to a domain X. Each label is an integer from the set $Y = \{1, \ldots, K\}$. A multiclass classifier is a function $f : X \to Y$ that maps an instance $\mathbf{x} \in X \subset \mathbb{R}^D$ onto an element of Y.

The task is to find a definition for the unknown function, $f(\mathbf{x})$, given the set of training instances. In a classifier ensemble framework we have a set of classifiers $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$, each classifier performing a mapping of an instance vector $\mathbf{x} \in \mathbb{R}^D$ onto the set of labels $Y = \{1, \ldots, K\}$. The design of classifier ensembles must face two main tasks: constructing the individuals classifiers, C_i, and developing a combination rule that finds a class label for $\mathbf{x}$ based on the outputs of the classifiers $\{C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_m(\mathbf{x})\}$. This paper is devoted to the first problem, the combination of the classifiers is being done with a simple majority voting scheme.

For more detailed descriptions of ensembles the reader is referred to other reviews: Dietterich (2000b), Webb (2000), Dzeroski and Zenko (2004), Merz (1999), or Fern and Givan (2003).

Techniques using multiple models usually consist of two independent phases: model generation and model combination (Merz, 1999). Most techniques are focused on obtaining a group of classifiers which are as accurate as possible but which disagree as much as possible. These two objectives are somewhat conflicting, since if the classifiers are more accurate, it is obvious that they must agree more frequently. Many methods have been developed to enforce diversity on the classifiers that form the ensemble (Dietterich, 2000c). Kuncheva (2001) identifies four fundamental approaches: (i) using different combination schemes, (ii) using different classifier models, (iii) using different feature subsets, and (iv) using different training sets. Perhaps the last one is the most commonly used. The algorithms in this last approach can be divided into two groups: algorithms that adaptively change the distribution of the training set based on the performance of the previous classifiers, and algorithms that do not adapt the distribution. Boosting methods are the most representative methods of the first group. The most widely used boosting methods are ADABOOST (Freund and Schapire, 1996) and its numerous variants, and Arc-x4 (Breiman, 1998). They are based on adaptively increasing the probability of sampling the instances that are not classified correctly by the previous classifiers.

Bagging (Breiman, 1996b) is the most representative algorithm of the second group. Bagging (after *B*ootstrap *agg*regat*ing*) just generates different bootstrap samples from the training set. Several empirical studies have shown that ADABOOST is able to reduce both bias and variance components of the error (Breiman, 1996c; Schapire et al., 1998; Bauer and Kohavi, 1999). On the other hand, bagging seems to be more efficient in reducing bias than ADABOOST (Bauer and Kohavi, 1999).

Although these techniques are focused on obtaining as diverse classifiers as possible, without deteriorating the accuracy of each classifier, Kuncheva and Whitaker (2003) failed to establish a clear relationship between diversity and ensemble performance.

Boosting methods are the most popular techniques for constructing ensembles of classifiers. Its popularity is mainly due to the success of ADABOOST. However, ADABOOST tends to perform very well for some problems but can also perform very poorly on other problems. One of the sources of the bad behaviour of ADABOOST is that although it is always able to construct diverse ensembles, in some problems the individual classifiers tend to have large training errors (Dietterich, 2000a). Moreover, ADABOOST usually performs poorly on noisy problems (Bauer and Kohavi, 1999; Dietterich, 2000a).

Schapire and Singer (1999) identified two scenarios where ADABOOST is likely to fail:

1. When there is insufficient training data relative to the "complexity" of the base classifiers.

2. When the training errors of the base classifiers become too large too quickly.

Sebban et al. (2002) proposed a stopping criterion for boosting algorithms, and there are other variants (Schapire and Singer, 1999; Webb, 2000; Bshouty and Gavinsky, 2002; Eibl and Pfeiffer, 2005) that try to overcome the drawbacks of the method. Nevertheless, ADABOOST is still likely to fail on noisy problems or when there is not much available data. Unfortunately, these two scenarios are very common in real-world problems.

The margin (Mason et al., 2000) of a real-valued function $f : X \to \mathbb{R}$ on a training instance $(\mathbf{x}, y) \in X \times \{-1, 1\}$ is defined as $yf(\mathbf{x})$, so that if the sign is correct the margin is positive. The size of the margin can be interpreted as an indication of the confidence of the classification. The success of ADABOOST has been partially explained in terms of the “boosting” of the margins that usually is achieved, however this is not the only important factor in its success. Several experiments have shown that ADABOOST is able to perform better than algorithms that are more efficient than ADABOOST in optimising the margins (Breiman, 1999; Grove and Schuurmans, 1998). There are other attempts to explain the good performance of boosting (Rosset et al., 2004).

Alternatively, some works have focused on using different subsets of the inputs, that is, different subspaces, to train the classifiers. Cherkauer (1996) trained an ensemble of classifiers which consisted of 32 neural networks trained using 8 different subsets of input features together with 4 different network sizes. Chen et al. (1997) studied the use of different features for training multiple classifiers in a framework of text-independent speaker recognition. Tumer and Ghosh (1996) used a similar technique to classify sonar signals, but they found that removing just a few of the input features hurts the performance of the classifiers so much that the resulting ensemble had a poor performance.

As a matter of fact, it has been shown that this technique is capable of obtaining a good performance only when the input features are highly redundant (Dietterich, 2000b). Ho (1998) proposed a method for constructing a decision forest by randomly selecting subspaces from the original data set, and reported very good results.

As an useful alternative, evolutionary computation has also been successfully applied to ensemble construction. Zhou et al. (2002) used a genetic algorithm to obtain a subset of an ensemble of classifiers that was able to outperform the ensemble using all the classifiers. Liu et al. (2000) used the concept of *negative correlation* to improve the diversity of a population of classifiers. García-Pedrajas et al. (2005) developed a cooperative coevolutionary method for ensemble construction with excellent results on classification problems. Ortiz-Boyer et al. (2005) used a real-coded genetic algorithm to optimise the weights of each classifier within an ensemble.

In summary, the construction of ensembles aims to fulfil two different objectives: accuracy of classifiers and diversity among them. These two objectives are partially in conflict, as the more accurate the classifiers are the less diverse they must be. Among the different approaches presented above, we can highlight the following techniques to enforce accuracy and diversity:

- Bagging methods sample the training data with replacement to obtain different sets to train the classifiers.

- Boosting methods enforce accuracy and diversity by putting more emphasis on instances that are misclassified by previous classifiers.

- Subspace methods encourage diversity by training the classifiers using different subsets of input features or different projections of the original data.

In our approach we combine the rationale of these previous approaches. On the one hand, the use of different subspaces, or different projections, is capable of inducing diversity and produces better ensembles. On the other hand, putting more emphasis on difficult instances, as boosting methods do, obtains very good performance and is able to reduce both bias and variance of the classifiers. Nevertheless, boosting is very sensitive to noise and does not usually perform well on small data sets. As the weight of each instance for the training of the classifier depends on whether it is correctly classified too much emphasis may be put on noisy instances or outliers. So, our approach is based on combining the main ideas underlying boosting and random subspace methods, namely:

- All the classifiers should receive all the training instances for learning, and all of them are equally weighted. We are thus neither throwing away any instances which is something which happens in bagging, or putting more emphasis on misclassified instances as in boosting.
- Each classifier uses a different nonlinear projection of the original data onto a space of the same dimension. We are using this to create diversity within the training sets.
- Following the basic principles of boosting, each nonlinear projection is constructed in order to make the classification of difficult instances easier.

This approach is able to incorporate the advantages of boosting without its main drawbacks. The construction of the projection taking into account only instances that have been misclassified by a previous classifier permits the new classifier to focus on difficult instances. Nevertheless, as this classifier receives all the data, the sensitivity to noise and the effect of small data sets is greatly reduced.

In this way, the method presented in this paper is a hybrid of approaches (iii) and (iv) identified by Kuncheva (2001). It uses different feature spaces and only a subset of instances to construct those spaces.

The problem we must solve is how to construct a projection that favours the correct classification of a subset of instances. This problem is solved by means of a neural network as is explained in Section 2.

This paper is organised as follows: Section 2 explains in depth the proposed methodology; Section 3 surveys some related work; Section 4 shows the experimental setup and the results obtained with the proposed algorithm and several standard methods; Section 5 reports some further experiments aimed at understanding the behaviour of the method; and Section 6 states the conclusions of our work.

2. Constructing Nonlinear Projections

The problem of obtaining a useful projection is not trivial. Methods for projecting data are focused on the features of the input space, and do not take into account the labels of the instances. Moreover, most of them are specifically useful for non-labelled data and aimed at data analysis. Among the most widely used of these methods we can cite principal component analysis (PCA) (Jolliffe, 1986), Kohonen's self-organizing maps (Kohonen, 2001), and factor analysis (Gorsuch, 1983). Nevertheless, none of them is appropriate for our problem.

Our approach is based on the use of the projection carried out by the hidden layer of a multilayer perceptron neural network when it is used for classification purposes. In order to get a useful projection, we must take into account the role of the hidden layer in a neural network. As stated in Haykin (1999), p. 199:

> *Hidden neurons* play a critical role in the operation of a multilayer perceptron with back-propagation learning because they act as *feature detectors*. As the learning process progresses, the hidden neurons begin to gradually "discover" the salient features that characterise the training data. They do so by performing a nonlinear transformation on the input data into a new space called the *hidden space*, or *feature space*. In this new space the classes of interest in a pattern-classification task, for example, may be more easily separated from each other than in the original input space.

From this point of view, neural networks can be considered similar to *basis function* models (Denison et al., 2002). These models assume that the function to be implemented, g, is made up of a linear combination of basis functions and corresponding coefficients. Hence g can be written:

$$g(x) = \sum_{i=1}^{H} \beta_i B_i(x), \quad x \in \mathcal{X} \subset \mathbb{R}^D, \tag{1}$$

where $\beta = (\beta_1, \ldots, \beta_k)'$ is the set of coefficients corresponding to basis functions $B = (B_1, \ldots, B_k)$. Typically, the basis functions in (1) are nonlinear transformations. Neural networks can be considered another example of basis function models. Methodologically, there is a major separation in the multilayer perceptron (MLP) approach, as the combination of the basis functions is not always linear, as in (1), and subsequent sets of basis functions, represented by further hidden layers, can be constructed.

Let us consider a feed-forward neural network with D inputs and a hidden layer with H nodes. The hidden layer carries out a non linear projection of input vector x to a vector h where:

$$h_i = f\left(\sum_{j=0}^{D} w_{ij} x_j\right), \qquad x_0 = 1.$$

As we have stated, each node performs a nonlinear projection of the input vector. So, $h = f(x)$, and the output layer obtains its output from vector h. In this context we can consider this projection as a basis function, so $B_i(x) = f\left(\sum_{j=0}^{D} w_{ij} x_j\right)$, and the output of the network is:

$$y(x) = F\left(\sum_{i=1}^{H} \beta_i B_i(x)\right)$$

where F is the transfer function of the output layer, and the β_i represent the weights of the connections from the hidden layer to the output layer. This projection performed by the hidden layer of a multi-layer perceptron distorts the data structure and inter-instance distances (Lerner et al., 1999) in order to achieve a better classification.

So, the projection performed by the hidden layer focuses on making the classification of the instances easier. Our approach to obtain a projection focused on making the classification of difficult instances easier consists of training a neural network with only the instances that have been incorrectly classified by the previous classifier. The number of hidden nodes of the network is the same as the number of input variables, so the hidden layer is performing a nonlinear projection into a space of the same dimension as the input space. As the network has been trained only with a subset of the training instances, the projection performed by the hidden layer focuses on making the classification of only this subset easier.

Once the network is trained, the projection implemented by the hidden layer is used to project all the data set, and these projections are fed to the new classifier to be trained.[1] Then, each classifier performs its task using as input a feature space that is created to make the classification of difficult instances easier.

This nethod can be used with any base classifier as the projection is made before training the classifier and the obtained projected data can be used for any type of classifier. The complexity of the classifiers is not increased, as the feature space into which the data is projected has the same dimension as the original input space.

2.1 Nonlinear Projection Based Algorithm

The initialisation of the network is very important for the overall performance as it has been shown that back-propagation is sensitive to initial conditions (Kolen and Pollack, 1991). For the initialisation of the weights of the networks, we used the method suggested by LeCun et al. (1998) and described in Haykin (1999). The weights are obtained from a uniform distribution within the interval $[-3/\sqrt{D}, 3/\sqrt{D}]$, where D is the number of inputs to the node. In this way, we try to avoid large initial values, that can saturate the transfer function and have a dramatic impact on the performance of the network.

The proposed method for constructing classifier ensembles is shown in Algorithm 1. The proposed algorithm has not incorporated any other ensemble construction method, that may improve its performance, in order not to obscure its behaviour.

Algorithm 1: Nonlinear Boosting Projection algorithm.

```
Data   : A training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, a base learning algorithm, $\mathbb{L}$, and the
         number of iterations $T$.
Result : The final classifier:
         $C^*(\mathbf{x}) = \arg\max_{y \in Y} \sum_{t: C(\mathbf{x}) = y} 1$.

1 $C_0 = \mathbb{L}(S)$
  for $t = 1$ to $T - 1$ do
2     $S' \subset S, S' = \{\mathbf{x}_i \in S : C_{t-1}(\mathbf{x}_i) \neq y_i\}$.
3     Train network $H$ with $S'$ and get projection $\mathbf{P}(\mathbf{x})$ implemented by the hidden layer of $H$.
4     $C_t = \mathbb{L}(\mathbf{P}(S))$
  end
```

We have named the algorithm *Non-Linear "Boosting" Projection* (NLBP) as the projections are constructed using the basic idea of boosting methods. The source code, in C, used for the standard and proposed methods is under GNU License and is freely available upon request to the authors.

Although we need to train an additional neural network for each new classifier of the ensemble, except the first one, the additional complexity of this training process is diminished by the fact that the projection network is trained using only the instances misclassified by the previous classifier.

1. To avoid confusion, we must remark that the network trained using the misclassified instances is not part of the ensemble, and only the projection carried out by its hidden layer is used.

We can illustrate how the method works with a toy example. Consider the simple case of a two-dimensional space and the learning and testing sets shown in Figure 1. The figure shows how there are two points in the training set that are, with a high probability, mislabelled noisy data. Using these two data sets we run our algorithm and ADABOOST with an ensemble of 50 classifiers and a neural network, a C4.5 tree and a support vector machine as base classifiers. The results are shown in Table 1.

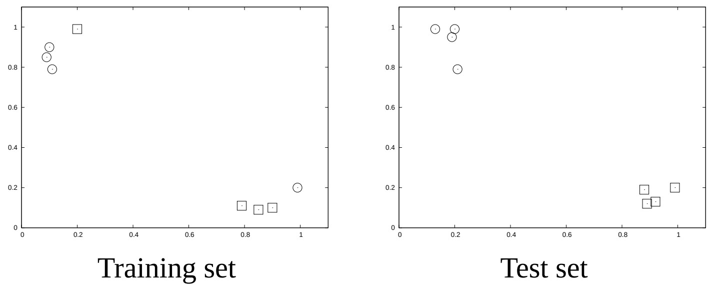

Figure 1: Training and testing data for a two-dimensional problem of two classes with two likely mislabelled noisy instances

The example illustrates the differences of our method with boosting methods and how these differences may constitute an advantage. After the initial classifier is trained, ADABOOST focus its effort on classifying the two misclassified instances correctly. The result is that the learning error drops eventually to 0, but the cost is overlearning the training data and poor generalisation. On the other hand, our method tries to classify the misclassified instances but without allowing the classifier to put too much effort on that task. The result is that the learning error is larger but the generalisation ability is better. This is, of course, a toy problem only intended to illustrate the underlying ideas of our method.

Base classifier	Ensemble method			
	ADABOOST		NLBP	
	Training	Test	Training	Test
Neural network	0.0000	0.5000	0.2500	0.0000
Support vector machine	0.0000	0.2500	0.2500	0.0000
C4.5	0.0000	0.2500	0.1250	0.1250

Table 1: Summary of training and test error results for the toy problem with an ensemble of 50 classifiers.

3. Related Work

The use of different spaces for ensemble construction has been extensive in recent research. Ho (1998) showed that the random subspace method was able to improve the generalisation error. The random subspace method, rooted in the theory of stochastic discrimination (Kleinberg, 2000), has common points with bagging, but instead of sampling instances it samples subspaces (Skurichina and Duin, 2001). The random subspace method has been successfully applied to different problems (Munro et al., 2003; Hall et al., 2003).

Opitz (1999) developed accurate and diverse classifiers for an ensemble using *ensemble feature selection*. In contrast to classical feature selection algorithms, that are focused on finding the best feature subset for a given classifier, ensemble feature selection adds an additional objective of promoting disagreement among the individual classifiers. Different strategies have been proposed to ensemble feature selection, such as hill climbing (Kohavi, 1995; Cunningham and Carney, 2000), a genetic algorithm (Opitz, 1999), and the heuristic method *Ensemble Forward and Backward Sequential Selection* (Aha and Bankert, 1995). A comparative study for medical diagnosis tasks is carried out in Tsymbal et al. (2003).

Utsugi (2001) proposed an ensemble of independent factor analysers (Anderson, 1984). This new statistical model assumes that each data point is generated by the sum of outputs of independently activated factor analysers. A maximum likelihood (ML) estimation algorithm for the parameters is derived using a Monte Carlo EM algorithm with a Gibbs sampler. The independent factor analysers developed into feature detectors that resemble complex cells in mammalian visual systems.

Yand et al. (2000) used mixtures of linear subspaces to create a classifier for face recognition. Rodríguez et al. (2006) proposed a method based on applying PCA to subsets of inputs variables. The method obtained very good results on a large set of real-world problems.

4. Experiments

In order to test the performance of our method on solving classification tasks, we need to compare it with other widely used ensemble methods. In this section we try to make as fair a comparison as possible. So, we have chosen three different base classifiers: a neural network, a C4.5 tree (Quinlan, 1993), and a support vector machine (SVM) (Cristianini and Shawe-Taylor, 2000) . The first two have been widely used for ensemble construction in the literature. The last one has been shown recently to achieve good results as a member of an ensemble (Kim et al., 2002; Diao et al., 2002), although its stability might suggest that it is not a suitable base learner for constructing an ensemble. For the SVM we used a Gaussian kernel and parameters $C = 10.0$ and $\gamma = 0.1$. We used the `LIBSVM` library (Chang and Lin, 2001). For the neural network we used a standard multilayer perceptron trained using a simple back-propagation algorithm (Rumelhart et al., 1986). The network has a hidden layer of 25 nodes and hyperbolic tangent transfer function for the hidden layer and logistic transfer function for the output layer. The network was trained for 100,000 iterations with a learning coefficient $\eta = 0.25$ and a momentum coefficient $\alpha = 0.1$. For C4.5 we used the available code by the author of the algorithm. We must state that these parameters are rather standard and have not been selected in any way to improve the performance of any of the methods.[2]

For all three base classifiers we performed experiments using bagging, Arc-x4, ADABOOST, and MADABOOST (Domingo and Watanabe, 2000).

Before deciding upon bagging we tested its performance against wagging. Wagging is a variant of bagging (Breiman, 1996a) that requires a learning algorithm that can use training instances with different weights. Wagging assigns random weights to the instances of the training set instead of

2. It is obvious that these parameters might not be appropriate for all the data sets. Nevertheless, adjusting the parameters for each data set and each base learner is computationally infeasible. Moreover, as the parameters are common for all the methods, any weakness will be shared for all of them.

resampling the training instances as in bagging. In our implementation the weights assigned to the instances follow a Poisson distribution (Webb, 2000). In a preliminary set of experiments we compared the performance of bagging and wagging, and found that bagging significantly outperformed wagging, so in all the experiments we have used bagging.

We use the Arc-x4 (Breiman, 1998) implementation in Bauer and Kohavi (1999). The factor $(1+e(\mathbf{x})^4)$ that weights each instance is rather heuristic, and the value of 4 for the exponent was experimentally obtained as optimal. These type of methods were named by Breiman (1998) *Arcing* methods after "*A*daptively *r*esampling and *c*ombining".

The ADABOOST algorithm is specifically designed for minimising the exponential loss function:

$$\sum_{i=1}^{m} \exp\left(-y_i f_\lambda(\mathbf{x}_i)\right),$$

where:

$$f_\lambda(\mathbf{x}_i) = \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}_i).$$

One of the most interesting features of ADABOOST is that the test error continues to improve, even when the learning error reaches 0 (Schapire et al., 1998). However, the marginal error reduction produced by each new classifier tends to decrease. On average, each new classifier has less impact on the test error than the previous members of the ensemble.

We use the ADABOOST version in Webb (2000). For ADABOOST it is required that the *weak learning* algorithm, in our case each individual classifier, achieves an error strictly less than 0.5. This cannot be guaranteed; especially when dealing with difficult multiclass problems. In our experiments when this error is not achieved, we generate a bootstrap sample from the original set and continue the algorithm assigning to that classifier a zero weight (Bauer and Kohavi, 1999; Zhou et al., 2002).

For the three base learners we make a preliminary test to compare resampling and reweighting versions of the boosting algorithms. For neural networks we found that reweighting performed better than resampling; for C4.5 and SVM we found that resampling achieved better results. So, in the experiments reported here for the standard methods we use reweighting for neural networks, and resampling for C4.5 and SVMs.

For the case of a neural network as a base classifier we make some additional experiments to improve the quality of the comparison. Firstly, instead of using standard ADABOOST, we used ADABOOST.MH (Schapire and Singer, 1999, 2000). Moreover, we also used the algorithms GENTLEBOOST (Friedman et al., 2000) and LOGITBOOST (Friedman et al., 2000).[3] Secondly, the hidden weights of the non-linear projection are also learned when using standard methods as an additional layer of the network, in order to avoid any spurious advantage for the proposed method derived from the use of the additional hidden layer represented by the non-linear projection.

All the ensembles are made up of 50 classifiers, regardless of the kind of base classifier. This number is fairly common in the literature. Although desirable, fixing the number of classifiers on

3. These three algorithms modify the learning of the base classifier, so we cannot use them with C4.5 and SVMs as we are using standard libraries for these two base classifiers.

each ensemble taking into account either the problem or the type of base learner is computationally unfeasible. Moreover, it is known (Zenobi and Cunningham, 2001) that the diversity and the accuracy of the ensemble usually plateau at some size between 10 and 50 members. The standard ensembles are also made up of 50 networks. Furthermore, for ADABOOST the error bound of the training error is almost 0 after the addition of about 30 classifiers to the ensemble (Kuncheva, 2003), so 50 is a widely used ensemble size.

4.1 Experimental Setup

For testing the validity of the proposed method we have selected 44 data sets from the UCI Machine Learning Repository (Hettich et al., 1998). A summary of these data sets is shown in Table 2.

Three of the data sets were reduced to make them of a manageable size. For isolet and zip we performed a principal component analysis and retained the first 34 and 50 principal components respectively. For the letter data set we use a randomly selected subset of 5000 instances from the whole set of 20000 instances.

In order to avoid confusion we will use the term "learning error" for referring to the error of a classifier on the training set, and the term "test error" for the error of a classifier on the testing set. In the literature, the terms "generalisation error" and "prediction error" are also common for referring to the error on the test set. However, we believe that these two terms are more appropriate for the probability of misclassifying a new sample, thus the generalisation error is the expected test error.

The experiments were conducted following the 5x2 cross-validation set-up (Dietterich, 1998). We perform five replications of a two-fold cross-validation. In each replication the available data is partitioned into two random equal-sized sets. Each learning algorithm is trained on one set at a time and tested on the other set. The original test proposed by Dietterich suffers from low replicability, so to test the differences between two algorithms we use the 5x2cv F test (Alpaydin, 1999).

This test has the following formulation. We have five replications, $i = 1, \dots, 5$, and two-folds, $j = 1, 2$, for each replication. $p_i^{(j)}$ is the difference between the error rates of the two classifiers on fold j of replication i. The average on replication i is $\bar{p} = (p_i^{(1)} + p_i^{(2)})/2$, and the estimated variance is $s_i^2 = ((p_i^{(1)} - \bar{p})^2 + ((p_i^{(2)} - \bar{p})^2$. Alpaydin (1999) proposed the test:

$$f = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} \left(p_i^{(j)} \right)^2}{2 \sum_{i=1}^{5} s_i^2}, \tag{2}$$

that is approximately F distributed with 10 and 5 degrees of freedom. As a general rule we consider a confidence level of 90%. The same 5x2cv partitions were used for all the reported experiments.

In all tables the error measure is $E = \frac{1}{n} \sum_{i=1}^{n} e_i$, where e_i is 1 if instance i is misclassified and 0 otherwise. In the following sections we report all the experiments performed with the proposed method and all the data sets used for testing the method.

The source code, in C, used for the standard and proposed methods as well as the partitions of the data sets, are freely available upon request to the authors.

4.2 Experimental Results

In the first set of experiments we tested our method against the standard ensemble methods. For a neural network as base learner we used bagging, ADABOOST.MH, MADABOOST, LOGITBOOST, GENTLEBOOST, and Arc-x4. Test error results are shown in Table 3. Tables 4 and 5 show the

Data set	Cases	Classes	Features			Inputs
			Continuous	Binary	Nominal	
anneal	898	5	6	14	18	59
audio	226	24	–	61	8	93
autos	205	6	15	4	6	72
balance	625	3	4	–	–	4
breast-cancer	286	2	9	–	–	9
card	690	2	6	4	5	51
dermatology	366	6	1	1	32	34
ecoli	336	8	7	–	–	7
gene	3175	3	–	–	60	120
german	1000	2	6	3	11	61
glass	214	6	9	–	–	9
glass-g2	163	2	9	–	–	9
heart	270	2	6	1	6	13
heart-c	302	2	6	3	4	22
hepatitis	155	2	6	13	–	19
horse	364	3	13	2	5	58
ionosphere	351	2	33	1	–	34
iris	150	3	4	–	–	4
isolet	7797	26	617	–	–	*34*
labor	57	2	8	3	5	29
led24	200	10	–	24	–	24
letter	*5000*	26	16	–	–	16
liver	345	2	6	–	–	6
lrs	531	10	101	–	–	101
lymph	148	4	–	9	6	38
optdigits	5620	10	64	–	–	64
page-blocks	5473	5	10	–	–	10
pendigits	10992	10	16	–	–	16
phoneme	5404	2	5	–	–	5
pima	768	2	8	–	–	8
primary-tumor	339	22	–	14	3	23
promoters	106	2	–	–	57	114
satimage	6435	6	36	–	–	36
segment	2310	7	19	–	–	19
sick	3772	7	2	20	2	33
sonar	208	2	60	–	–	60
soybean	683	19	–	16	19	82
vehicle	846	4	18	–	–	18
vote	435	2	–	16	–	16
vowel	990	11	10	–	–	10
waveform	5000	3	40	–	–	40
yeast	1484	10	8	–	–	8
zip	9298	10	256	–	–	*50*
zoo	101	7	1	15	–	16

Table 2: Summary of data sets. The Inputs column shows the number of inputs to the classifier as it depends not only on the number of input features but also on their type.

results using C4.5 and SVM respectively. As explained, with these two base learners we have used bagging, ADABOOST, MADABOOST, and Arc-x4.

The results in Table 3 show that the proposed method is significantly better than the standard methods for 17 data sets when compared with bagging, 20 with ADABOOST.MH, 19 with GENTLEBOOST, 16 with LOGITBOOST, 17 with MADABOOST and Arc-x4. It is worse than LOGITBOOST in 1 data set and worse than MADABOOST in 2 data sets. The differences are not significant for the other data sets.

Table 4 shows the test error of NLBP and the four standard methods for C4.5 tree as base classifier. As a summary, NLBP is able to significantly improve bagging in 16 data sets, AdaBoost in 13, MadaBoost in 17 and Arc-x4 in 14. It is significantly worse in 3, 5, 3, and 3 data sets respectively. These results show a general advantage of our method. Nevertheless, the experiments using C4.5 as base learner show the worst performance of NLBP. We think the reason is the fact that C4.5 is a tree algorithm that best works with nominal and binary variables; the projection performed by NLBP transform all variables into continuous variables, and that may have a negative effect on C4.5 performance. However, even in this unfavourable scenario NLBP is able to outperform classical methods.

The results in Table 5 show that, for a SVM as base classifier, the proposed method is significantly better than bagging in 19 data sets, than ADABOOST in 23 data sets, than MADABOOST in 16 and than Arc-x4 in 21 data sets. NLBP is significantly worse than the standard methods only once for bagging, MADABOOST, and Arc-x4, and never for ADABOOST. In general terms, the performance of NLBP using a SVM as base classifier, is very good. We believe that the use of a SVM is able to obtain the best of NLBP as the non-linear projections obtained by NLBP are able to make the task of SVM easier.

As a summary, for all the three classifiers NLBP is able to obtain very good results, outperforming both bagging and boosting methods. Of most interest is its performance in some of the most difficult problems, such as breast-cancer, hepatitis, horse, isolet, letter, liver, lymphography, primary-tumor, sonar, and vowel.

5. Analysis of the Proposed Method

In the previous section we have shown that the nonlinear projection approach is very competitive when compared with the standard methods of ensemble construction. In this section we present additional experiments that try to gain some insight into how it works.

In a first set of experiments we investigate if the good behaviour of NLBP is due to any side effect of the proposed algorithm or has its source in the idea of constructing nonlinear projections using only difficult instances. Then, we review the κ-error diagrams and show the κ-error diagrams of the standard and proposed methods. The behaviour of NLBP shown by κ-error diagrams suggests a further study on the sensitivity to noise of the method; that study is performed for all the data sets used in the previous experiments, and the three base classifiers.

5.1 Control Experiments

The first set of experiments of this section is focused on studying whether the basic contribution of our method, that is, using the subset of misclassified instances to obtain a nonlinear projection that makes subsequent classification easier, is responsible for the excellent results reported in the previous section.

Data Set	NLBP	Standard ensemble methods					
		Bagging	AdaB.MH	GentleB	LogitB	MadaB	Arc-x4
anneal	0.0100	0.0114	0.0154	0.0091	0.0512✓	0.0107	0.0103
audiology	0.2557	0.2495	0.2389	0.2531	0.2876	0.2557	0.2584
autos	0.3004	0.3198✓	0.3277✓	0.3257	0.3325	0.3276	0.3296
balance	0.0509	0.0602	0.0522	0.0525	0.0934✓	0.0304✗	0.0330
breast-c	0.3098	0.3168	0.3266	0.3224	0.2951	0.3406✓	0.3056
card	0.1490	0.1498	0.1614	0.1780✓	0.1553	0.1701✓	0.1582
dermatology	0.0246	0.0268	0.0268	0.0295	0.0301	0.0284	0.0268
ecoli	0.1476	0.1494	0.1631✓	0.1923✓	0.1541	0.1583✓	0.1637✓
gene	0.0979	0.1039	0.1026✓	0.1049	0.1017✓	0.1052	0.1015
german	0.2588	0.2620	0.2864✓	0.2802	0.2646	0.2816✓	0.2706✓
glass	0.3206	0.3206	0.3243	0.3327	0.3804✓	0.3271	0.3299
glass-g2	0.2221	0.2257	0.2677✓	0.2478	0.2626✓	0.2551✓	0.2454✓
heart	0.1985	0.2074	0.2096	0.2244✓	0.1941	0.2185✓	0.2215✓
heart-c	0.1762	0.1841	0.1974	0.2027	0.1835	0.2033✓	0.1954
hepatitis	0.1795	0.2094✓	0.2001	0.2064	0.1988	0.2090	0.2051
horse	0.3297	0.3363✓	0.3533	0.3698✓	0.3286	0.3560✓	0.3489✓
ionosphere	0.0980	0.1304✓	0.1418✓	0.1435✓	0.1424✓	0.1418✓	0.1481✓
iris	0.0440	0.0467	0.0493	0.0480	0.0480	0.0493	0.0454
isolet	0.0677	0.0890✓	0.0918✓	0.0806✓	0.1079✓	0.0773✓	0.0886✓
labor	0.1084	0.1964✓	0.1860✓	0.1858✓	0.0981	0.1894✓	0.1930✓
led24	0.3730	0.3780	0.4040✓	0.3910	0.3870	0.3880	0.3900
letter	0.2086	0.2631✓	0.2524✓	0.2194	0.2039✗	0.2075	0.2490✓
liver	0.3084	0.3101	0.3350	0.3523✓	0.3107	0.3246✓	0.3107
lrs	0.1149	0.1085	0.1115	0.1100	0.1247	0.1108	0.1134
lymph	0.1635	0.1878	0.2149✓	0.2135✓	0.1878	0.2189	0.2176✓
optdigits	0.0190	0.0229✓	0.0255✓	0.0197	0.0225✓	0.0218	0.0214✓
page-blocks	0.0328	0.0371✓	0.0379✓	0.0774✓	0.0421	0.0332	0.0351✓
pendigits	0.0104	0.0136✓	0.0142✓	0.0168	0.0381✓	0.0107	0.0131✓
phoneme	0.1936	0.2030	0.1943	0.2133	0.1756	0.1669✗	0.2065
pima	0.2464	0.2500	0.2534	0.2943✓	0.2448	0.2508	0.2451
primary-t	0.5870	0.5940	0.5911✓	0.6253✓	0.5805	0.5975	0.5976
promoters	0.1924	0.2302✓	0.2321✓	0.2302✓	0.1906	0.2302✓	0.2302✓
satimage	0.1061	0.1203✓	0.1216✓	0.1230✓	0.1092	0.1141✓	0.1193✓
segment	0.0332	0.0411✓	0.0398	0.0561✓	0.0594✓	0.0376	0.0454
sick	0.0239	0.0251✓	0.0262✓	0.0557	0.0242✓	0.0241	0.0280
sonar	0.2077	0.2536✓	0.2644✓	0.2683✓	0.2461✓	0.2644✓	0.2673✓
soybean	0.0653	0.0679	0.0682	0.0670	0.1564✓	0.0682	0.0682
vehicle	0.1849	0.1816	0.1983	0.2076	0.1965	0.1790	0.1884
vote	0.0487	0.0552	0.0593	0.0616✓	0.0556	0.0607	0.0616
vowel	0.0731	0.1354✓	0.0820	0.0697	0.2188✓	0.1028✓	0.0731
waveform	0.1373	0.1404	0.1440	0.1580✓	0.1406	0.1434✓	0.1443
yeast	0.4074	0.4123	0.4119	0.4824✓	0.4316	0.4121	0.4189
zip	0.0147	0.0189✓	0.0203✓	0.0169	0.0163✓	0.0175	0.0178✓
zoo	0.0494	0.0631	0.0611	0.0651	0.0812✓	0.0632	0.0651
win/loss		17/0	20/0	19/0	16/1	17/2	17/0

✓/✗ significantly worse/better than NLBP method using F test.

Table 3: Summary of test error results for ensembles using a neural network as base learner and 5x2cv.

Data Set	NLBP	Standard ensemble methods			
		Bagging	AdaBoost	MadaBoost	Arc-x4
anneal	0.0154	0.0165	0.0085	0.0109	0.0109
audiology	0.2672	0.2858✓	0.2434	0.2557	0.2425
autos	0.2907	0.3160	0.2624	0.2750	0.2653
balance	0.0685	0.1527✓	0.2115✓	0.1795✓	0.2125✓
breast-cancer	0.2811	0.2902	0.3245✓	0.3147✓	0.3336✓
card	0.1591	0.1414	0.1449	0.1478	0.1507
dermatology	0.0273	0.0208	0.0323	0.0312	0.0361✓
ecoli	0.1530	0.1750	0.1738✓	0.1756✓	0.1720✓
gene	0.0983	0.0751✗	0.0765✗	0.0718✗	0.0732✗
german	0.2520	0.2534	0.2712✓	0.2626	0.2654✓
glass	0.3150	0.3140	0.3000	0.3028	0.3019
glass-g2	0.2198	0.1756✗	0.1462✗	0.1585	0.1609
heart	0.1859	0.1948	0.2052✓	0.2133	0.2044
heart-c	0.1841	0.1927	0.2046✓	0.2053✓	0.2033
hepatitis	0.1821	0.2066✓	0.2104	0.2118✓	0.2013
horse	0.3258	0.3308	0.3308	0.3258	0.3242
ionosphere	0.0507	0.0815✓	0.0763	0.0798✓	0.0712
iris	0.0467	0.0480	0.0587	0.0573	0.0520
isolet	0.0670	0.2151✓	0.1217✓	0.1545✓	0.1320✓
labor	0.0741	0.1227✓	0.1049	0.1190	0.0978
led24	0.3940	0.3330	0.3680	0.3620	0.3730
letter	0.1128	0.2070✓	0.1196	0.1426✓	0.1243
liver	0.3246	0.3159	0.3252	0.3159	0.3194
lrs	0.1115	0.1454✓	0.1398✓	0.1481✓	0.1428✓
lymphography	0.1838	0.1811	0.1838	0.1973	0.1905
optdigits	0.0193	0.0555✓	0.0206	0.0312✓	0.0242✓
page-blocks	0.0290	0.0285	0.0307	0.0303	0.0298
pendigits	0.0091	0.0271✓	0.0090	0.0186✓	0.0143✓
phoneme	0.1287	0.1378	0.1112✗	0.1137✗	0.1082✗
pima	0.2484	0.2440	0.2612	0.2516	0.2625
primary-tumor	0.5770	0.5705	0.5970	0.5781	0.5858
promoters	0.2189	0.1792	0.2038	0.1793	0.1736
satimage	0.0942	0.1152✓	0.0912	0.0975	0.0931
segment	0.0304	0.0382	0.0233✗	0.0339	0.0280
sick	0.0286	0.0163✗	0.0139✗	0.0156✗	0.0151✗
sonar	0.1836	0.2471✓	0.2462✓	0.2385✓	0.2433✓
soybean	0.0650	0.0764	0.0682	0.0747	0.0679
vehicle	0.2109	0.2650✓	0.2440	0.2610✓	0.2518
vote	0.0506	0.0469	0.0593	0.0620	0.0625
vowel	0.0786	0.1996✓	0.1208✓	0.1556✓	0.1334✓
waveform	0.1387	0.1678✓	0.1625✓	0.1658✓	0.1599✓
yeast	0.4051	0.4063	0.4348✓	0.4116✓	0.4239✓
zip	0.0163	0.0841✓	0.0371✓	0.0523✓	0.0423✓
zoo	0.0848	0.0835	0.0672	0.0653	0.0673
win/loss		16/3	13/5	17/3	14/3

✓/✗ significantly worse/better than NLBP method using F test.

Table 4: Summary of test error results for ensembles using a C4.5 tree as base learner and 5x2cv.

Data Set	NLBP	Standard ensemble methods			
		Bagging	AdaBoost	MadaBoost	Arc-x4
anneal	0.0136	0.0508✓	0.0267	0.0325✓	0.0312✓
audiology	0.2522	0.2893✓	0.2637	0.2540	0.2593
autos	0.2906	0.3179✓	0.3064	0.3063	0.3034
balance	0.0426	0.0691✓	0.0595	0.0624✓	0.0502
breast-cancer	0.2930	0.3091	0.3790✓	0.3448✓	0.3630✓
card	0.1632	0.2162✓	0.2409✓	0.2238✓	0.2377✓
dermatology	0.0246	0.0290	0.0312	0.0301	0.0317
ecoli	0.1512	0.1476	0.1976✓	0.1482	0.1750✓
gene	0.0998	0.1168✓	0.1162✓	0.1207✓	0.1225✓
german	0.2680	0.2992✓	0.2914✓	0.2916✓	0.2946✓
glass	0.3234	0.3383	0.3411	0.3420✓	0.3327
glass-g2	0.2098	0.2248	0.2196	0.2063	0.2210
heart	0.1844	0.1956✓	0.2281✓	0.2119✓	0.2244✓
heart-c	0.1940	0.2126	0.2510✓	0.2331✓	0.2417✓
hepatitis	0.1847	0.1899	0.1948	0.1911	0.1936
horse	0.3440	0.3698✓	0.3692	0.3648✓	0.3654
ionosphere	0.0496	0.0547	0.0644	0.0581	0.0576
iris	0.0467	0.0413	0.0467	0.0413	0.0493
isolet	0.0682	0.1193✓	0.0743	0.0938✓	0.0794✓
labor	0.0875	0.1086	0.1293✓	0.1119	0.1222✓
led24	0.3680	0.3950✓	0.4330✓	0.4090✓	0.4310✓
letter	0.1016	0.1035✓	0.1380✓	0.0940✗	0.1646✓
liver	0.3310	0.3159	0.3623✓	0.3206	0.3333✓
lrs	0.1130	0.1172	0.1221	0.1232	0.1318
lymphography	0.1811	0.2797✓	0.2595✓	0.2568✓	0.2486✓
optdigits	0.0162	0.0180	0.0226✓	0.0200	0.0203✓
page-blocks	0.0347	0.0384✓	0.0401✓	0.0375	0.0382
pendigits	0.0060	0.0057	0.0067	0.0049	0.0053
phoneme	0.1769	0.1565	0.1574	0.1556	0.1585✗
pima	0.2372	0.2526	0.2953✓	0.2505	0.2711✓
primary-tumor	0.5911	0.6365✓	0.6371✓	0.6289✓	0.6318✓
promoters	0.2057	0.1943	0.2094	0.2113	0.2132
satimage	0.0888	0.0904	0.0929	0.0867	0.0933
segment	0.0413	0.0470✓	0.0409	0.0426	0.0408
sick	0.0313	0.0355✓	0.0343✓	0.0339	0.0339✓
sonar	0.1615	0.2414✓	0.2413✓	0.2568✓	0.2615✓
soybean	0.0659	0.0712	0.0732✓	0.0685	0.0752
vehicle	0.1846	0.2213✓	0.2314✓	0.2194✓	0.2234
vote	0.0446	0.0524	0.0570✓	0.0510	0.0538
vowel	0.0467	0.0555	0.0448	0.0418	0.0434
waveform	0.1418	0.1530	0.1697✓	0.1512	0.1624✓
yeast	0.4093	0.4160	0.4291✓	0.4233	0.4443✓
zip	0.0182	0.0158✗	0.0188	0.0170	0.0172
zoo	0.0533	0.0652	0.0455	0.0515	0.0575
win/loss		19/1	23/0	16/1	21/1

✓/✗ significantly worse/better than NLBP method using F test.

Table 5: Summary of test error results ensembles with using a SVM as base learner and 5x2cv.

The first experiment tests if the performance of the method is due to the use of different projections. It is known that the use of random subspaces is able to improve the performance of an ensemble (Ho, 1998), and it is possible that the use of different projections of the original data, no matter how they are obtained, could be the cause of the good performance of NLBP. So we have trained the classifiers that make up the ensemble using a different nonlinear random projection for each classifier. The random nonlinear projection is obtained using the same neural network used in NLBP, but with the weights randomly initialised in the interval $[-1,1]$ and no training.

The second experiment aims to establish if a similar performance can be obtained if we obtain different nonlinear projections using all the instances of the training set instead of only the misclassified instances. Thus, for this simulation, we apply Algorithm 1 but projection $\mathbf{P}(\mathbf{x})$ is constructed using the whole training set, S, instead of the subset of instances incorrectly classified by the previous classifier, S'. The results of these two experiments for the three base classifiers are shown in Table 6.

The table shows how NLBP is performing significantly better than the two control algorithms for many problems. This experiment supports the approach of the proposed algorithm, ruling out as a cause of its good performance the use of an additional layer of learning. For a neural network NLBP is significantly better than a random projection in 16 data sets, and better than the projection using all the instances in 20 data sets. Similar results are obtained for a C4.5 tree and a SVM.

5.2 κ - Error Diagrams

One way of understanding the behaviour of ensemble methods is by means of a κ-error diagram (Margineantu and Dietterich, 1997; Dietterich, 2000a). These diagrams represent a point for each pair of classifiers. The x coordinate is a measure of the diversity of the two classifiers known as κ measure, the y coordinate is the average error of the two classifiers on the test data. The two values are conflicting, as it is obvious that we cannot have both perfect and independent classifiers. The κ-error diagram is the scatter plot of the points corresponding to all pairs of classifiers.

The κ measure is defined as follows: let us consider a problem with K classes, and let $\mathbf{C}$ be a $K \times K$ matrix such that C_{ij} contains the number of instances assigned to class i by the first classifier and to class j by the second classifier. Let us define:

$$\Theta_1 = \frac{\sum_{i=1}^{K} C_{ii}}{n},$$

and

$$\Theta_2 = \sum_{i=1}^{K} \left(\sum_{j=1}^{K} \frac{C_{ij}}{n} \times \sum_{j=1}^{K} \frac{C_{ji}}{n} \right),$$

where n is the number of instances. Then, the κ statistic is defined:

$$\kappa = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}.$$

When the agreement of the two classifiers equals that expected by chance $\kappa = 0$; when they agree on every instance $\kappa = 1$. Negative values mean a systematic disagreement between the two classifier.

Data Set	Neural network			C4.5			SVM		
	NLBP	Random	All	NLBP	Random	All	NLBP	Random	All
anneal	0.0100	0.0118	0.0096	0.0154	0.0294✓	0.0136	0.0136	0.0243✓	0.0089
audiology	0.2557	0.2734	0.2681	0.2672	0.3823✓	0.2779	0.2522	0.4956✓	0.2487
autos	0.3004	0.3111	0.3394✓	0.2907	0.3404✓	0.2994	0.2906	0.3765	0.2867
balance	0.0509	0.0995✓	0.0851✓	0.0685	0.1002✓	0.0822	0.0426	0.1008✓	0.0877✓
breast-c	0.3098	0.3371	0.3504✓	0.2811	0.2727	0.3140✓	0.2930	0.2769	0.3546✓
card	0.1490	0.1710✓	0.1861✓	0.1591	0.1704	0.1646	0.1632	0.1928✓	0.1785✓
dermatol.	0.0246	0.0263	0.0301	0.0273	0.0306	0.0372✓	0.0246	0.0306	0.0284
ecoli	0.1476	0.1363✓	0.1512✓	0.1530	0.1506	0.1655	0.1512	0.1512	0.1554
gene	0.0979	0.0996	0.1009✓	0.0983	0.2363✓	0.1253✓	0.0998	0.3997✓	0.1008
german	0.2588	0.2626✓	0.2840✓	0.2520	0.2592	0.2750✓	0.2680	0.2686	0.2760
glass	0.3206	0.3383✓	0.3393	0.3150	0.3299✓	0.3421✓	0.3234	0.3524✓	0.3355✓
glass-g2	0.2221	0.2111	0.2417	0.2198	0.2444	0.2259	0.2098	0.2579✓	0.2369✓
heart	0.1985	0.2007	0.2267✓	0.1859	0.1748	0.2037	0.1844	0.1756	0.2348✓
heart-c	0.1762	0.2119	0.2245✓	0.1841	0.1888	0.1927	0.1940	0.2033	0.2199✓
hepatitis	0.1795	0.1949	0.1987	0.1821	0.1820	0.2130	0.1847	0.1860	0.2013
horse	0.3297	0.3346	0.3632✓	0.3258	0.3412	0.3302	0.3440	0.3527	0.3539
ionosph.	0.0980	0.1077✓	0.1521✓	0.0507	0.0524	0.1247✓	0.0496	0.0518	0.1225✓
iris	0.0440	0.0427	0.0480	0.0467	0.0613✓	0.0627✓	0.0467	0.0453	0.0467
isolet	0.0677	0.1038✓	0.0715	0.0670	0.2279✓	0.0875✓	0.0682	0.0734	0.0607✓
labor	0.1084	0.1223	0.1823✓	0.0741	0.1127	0.1543	0.0875	0.0915	0.1575✓
led24	0.3730	0.4000	0.3970✓	0.3940	0.4990✓	0.3790	0.3680	0.4280✓	0.3670
letter	0.2086	0.2692✓	0.2328✓	0.1128	0.2172✓	0.1665✓	0.1016	0.1094	0.1136✓
liver	0.3084	0.3066	0.3344	0.3246	0.3617	0.3554	0.3310	0.3438	0.3414
lrs	0.1149	0.1206	0.1206	0.1115	0.1341✓	0.1255✓	0.1130	0.1729✓	0.1160
lymph.	0.1635	0.1730	0.2108✓	0.1838	0.1824	0.2054	0.1811	0.1865	0.2081✓
optdigits	0.0190	0.0230✓	0.0158	0.0193	0.0494✓	0.0267✓	0.0162	0.0179	0.0142
page-b	0.0328	0.0393✓	0.0340	0.0290	0.0327✓	0.0304	0.0347	0.0373	0.0329
pendigits	0.0104	0.0162✓	0.0137✓	0.0091	0.0133✓	0.0132✓	0.0060	0.0062	0.0075
phoneme	0.1936	0.2103	0.1707	0.1287	0.1554✓	0.1552✓	0.1769	0.2122✓	0.1702
pima	0.2464	0.2396	0.2615	0.2484	0.2458	0.2518	0.2372	0.2354	0.2565✓
primary-t	0.5870	0.5846	0.5905	0.5770	0.5917	0.5793	0.5911	0.5823	0.5852
promoters	0.1924	0.2208	0.2283	0.2189	0.2434	0.2698✓	0.2057	0.5283✓	0.2245✓
satimage	0.1061	0.1154✓	0.1142✓	0.0942	0.1028	0.1032✓	0.0888	0.0891	0.0975✓
segment	0.0332	0.0570✓	0.0380✓	0.0304	0.0459✓	0.0311	0.0413	0.0609✓	0.0369
sick	0.0239	0.0290✓	0.0254	0.0286	0.0373✓	0.0244✗	0.0313	0.0320	0.0251✗
sonar	0.2077	0.2154	0.2558✓	0.1836	0.1789	0.2654✓	0.1615	0.1558	0.2577✓
soybean	0.0653	0.0641	0.0694	0.0650	0.0735	0.0662	0.0659	0.0861✓	0.0682
vehicle	0.1849	0.1936	0.1835	0.2109	0.2624✓	0.2059	0.1846	0.2196✓	0.1870
vote	0.0487	0.0446	0.0570	0.0506	0.0510	0.0556	0.0446	0.0478	0.0570
vowel	0.0731	0.1669✓	0.1810✓	0.0786	0.0845	0.1069✓	0.0467	0.1501✓	0.1729✓
waveform	0.1373	0.1395	0.1407	0.1387	0.1604✓	0.1451	0.1418	0.1550	0.1434
yeast	0.4074	0.4159	0.4119	0.4051	0.4074	0.4171✓	0.4093	0.4102	0.4120
zip	0.0147	0.0215✓	0.0148	0.0163	0.0818✓	0.0212✓	0.0182	0.0268✓	0.0128✗
zoo	0.0494	0.0474	0.0611	0.0848	0.1027	0.0789	0.0533	0.0713✓	0.0533
win/loss		16/0	20/0		20/0	19/1		17/0	17/2

✓/✗ significantly worse/better than NLBP method using F test.

Table 6: Control experiment for NLBP. Test error using a random non-linear projection, and a projection trained using all the instances.

Figures 2, 3, and 4 show κ-error diagrams of the first partition for several data sets with four standard methods and NLBP, and neural networks, C4.5, and SVMs as base classifiers, respectively. These diagrams are fairly representative of the diagrams of all the data sets.

For all the three base classifiers we verify the usual behaviour of bagging and boosting methods. Bagging provides diversity, but to a lesser degree than boosting. On the other hand, boosting's improvement of diversity has the side effect of deteriorating accuracy. NLBP behaviour is midway between these two methods. It is able to improve diversity, but to a lesser degree than boosting, without damaging accuracy as much as boosting. This behaviour suggests that the performance of NLBP in noisy problems can be better than the performance of boosting methods. The next section is devoted to studying the sensitivity to noise of NLBP, and tests that hypothesis.

5.3 Effect of Noise

Several researchers have reported that boosting methods, among them ADABOOST, degrade their performance in the presence of noise. Dietterich (2000a) tested this effect introducing artificial noise in the class labels of different data sets and confirmed this behaviour. In this section we study the sensitivity of our method to noise.

To add noise to the class labels we follow the method of Dietterich (2000a). To add classification noise at a rate *r*, we chose a fraction *r* of the instances and changed their class labels to be incorrect choosing uniformly from the set of incorrect labels. We chose all the data sets and three rates of noise, 5%, 10%, and 20%. With this three levels of noise we have performed the experiments using the 5x2cv setup and NLBP, bagging and ADABOOST ensemble methods and compared the results as the level of noise increases.

For the noise study we have used bagging and ADABOOST as the representative of the boosting methods (for neural networks the used method is ADABOOST.MH as in the previous experiments). Tables 7, 8, and 9 show the comparison of the three methods at noise levels 5%, 10%, and 20%, for a neural network, a C4.5 tree and a SVM respectively. First of all, we can corroborate that tables confirm that bagging is less affected by noise than boosting, as has been shown in several papers, for example, (Dietterich, 2000a).

Table 7 shows the results for a neural network. For a noise level of 5% the results are not much different from the results without noise for all the three methods. Bagging performs slightly worse and ADABOOST.MH slightly better. It seems that for this low noise level the algorithms perform as in the case without noise. For 10% of noise the performance of bagging improves, achieving a win/loss record of 13/3 with NLBP, better than the record without noise, 17/0. On the other hand, the performance of ADABOOST.MH drops to a win/loss record of 28/0 with NLBP. Here the sensibility to noise of ADABOOST.MH is clearly evident.

The behaviour with a noise level of 20% is less clear. Bagging's performance drops to a record of 22/1 and ADABOOST.MH improves to 20/2. Two reasons can account for this behaviour: i) all the algorithms are performing badly due to the large amount of noise, so the results have a higher dependency on random factors, ii) the differences in the test error observed on the different partitions can be very high, making the F test, that depends on the variance between the error within the same partition (see Equation 2), less able to conclude that the differences are significant.

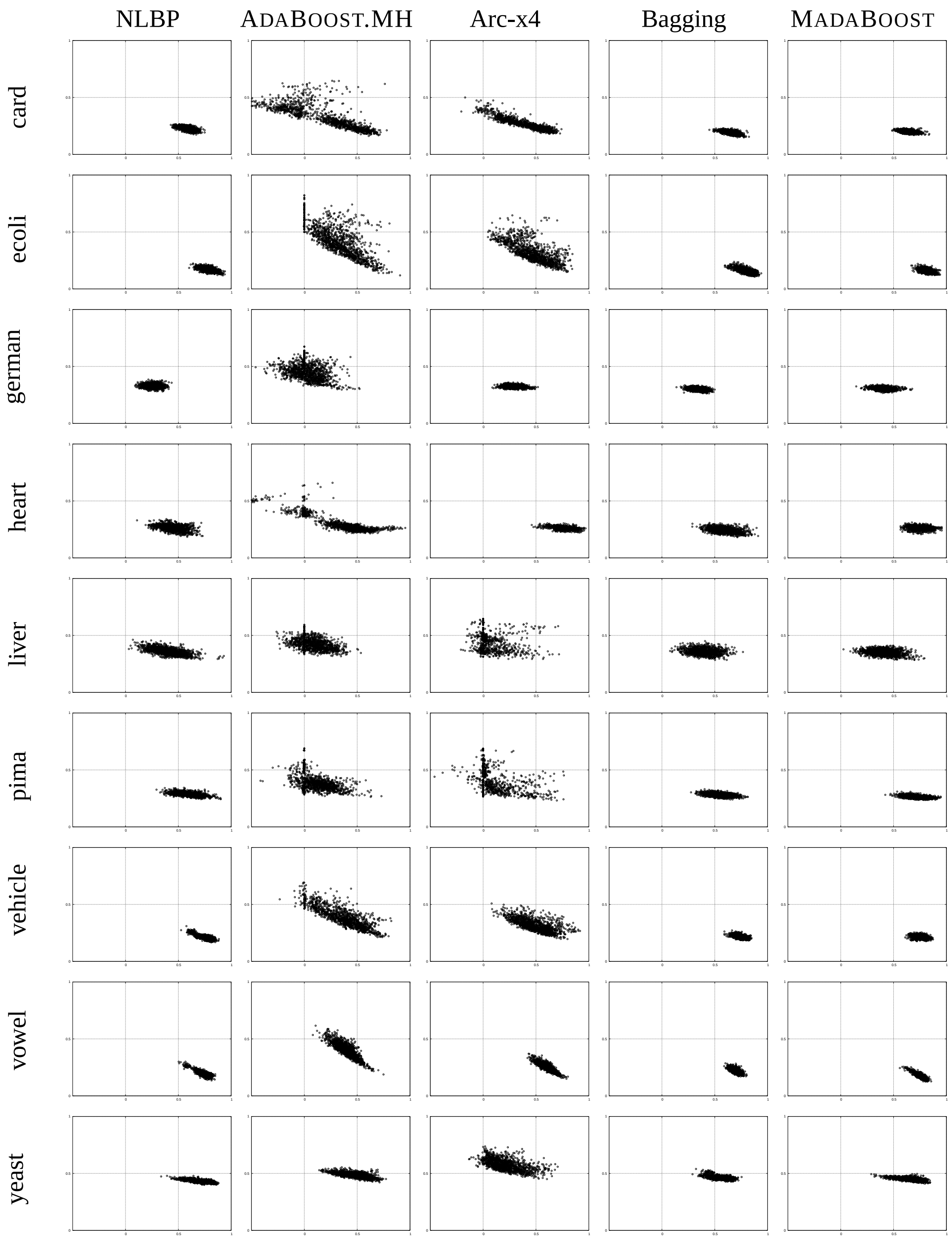

Figure 2: κ-error diagram for the data sets using the five ensemble methods and a neural network as base learner. κ measure is represented on horizontal axis in the interval $[-0.5, 1]$, and error value on vertical axis in the interval $[0, 1]$.

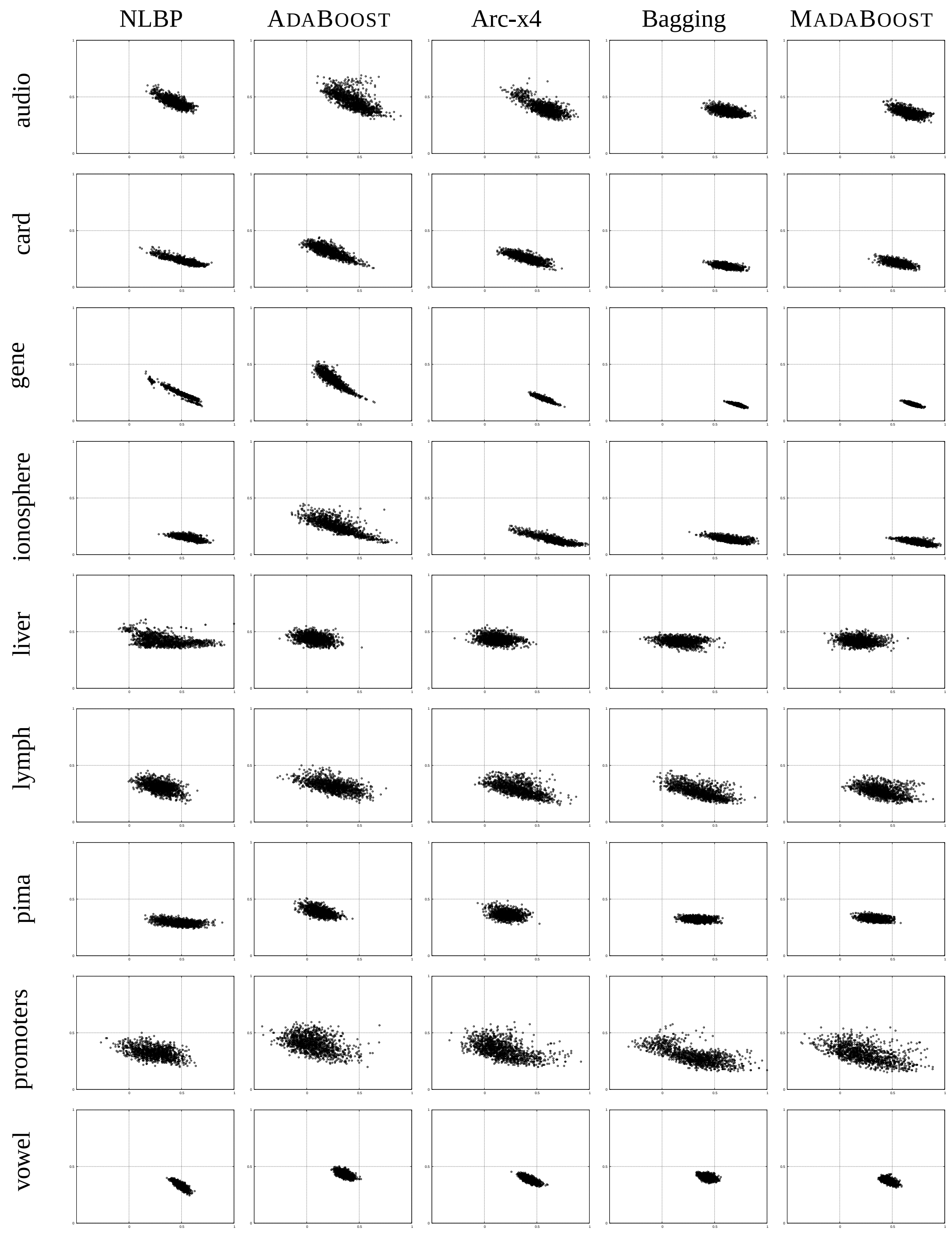

Figure 3: κ-error diagram for the data sets using the five ensemble methods and a C4.5 tree as base learner. κ measure is represented on horizontal axis in the interval $[-0.5, 1]$, and error value on vertical axis in the interval $[0, 1]$.

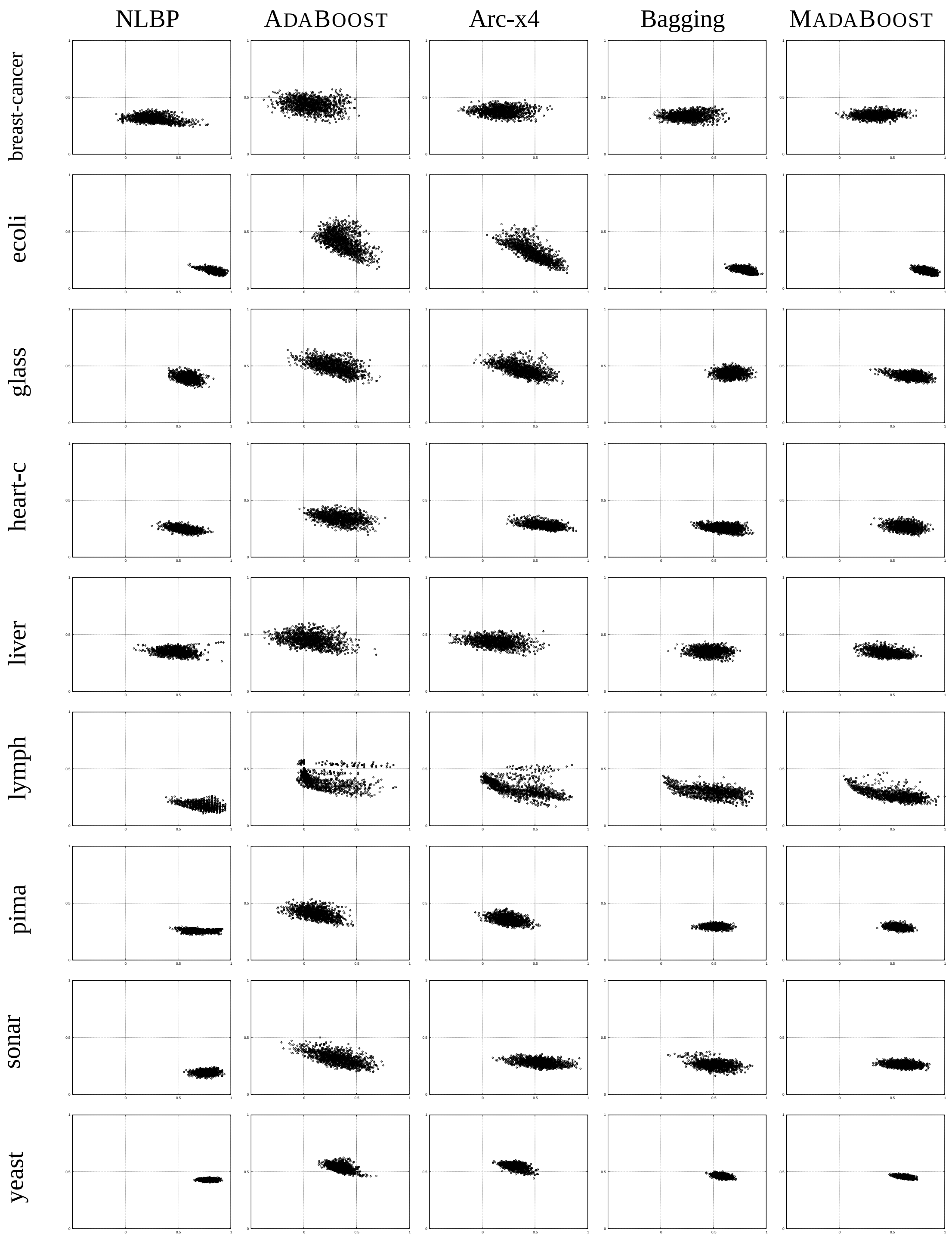

Figure 4: κ-error diagram for the data sets using the five ensemble methods and a SVM as base learner. κ measure is represented on horizontal axis in the interval $[-0.5, 1]$, and error value on vertical axis in the interval $[0, 1]$.

Using C4.5 as base learner, Table 8, the results are much the same. For a noise level of 5% the performance of the three algorithms is similar to the original one. When the noise level increases to 10% the performance of ADABOOST degrades from a win/loss record of 13/5 (with no artificial noise) to a record of 19/2. Bagging slightly worsens compared with NLBP, from a record of 16/3 to 20/6. When the noise level is 20% the effect is more marked for both algorithms, with a record of 18/3 for bagging and 25/1 for ADABOOST.

Table 9 shows a similar behaviour for the three algorithms when using a SVM but with some differences. The performance of ADABOOST dramatically decreases with noise as in the previous cases. For a SVM the negative effect of noise on ADABOOST is even more marked. But the case of bagging is slightly different from the two previous classifiers. With a SVM bagging is less affected by noise, even improving its compared performance with NLBP; for a noise level of 20% bagging is worse than NLBP in only 11 data sets.

In summary, NLBP has the very desirable property of behaving, at least, as robustly as bagging in the presence of noise. This is probably due to the fact that NLBP does not put so much emphasis on incorrectly classified instances as boosting does. However, NLBP does use the incorrectly classified instances in determining the nonlinear representation of the input data which provides it with a performance enhancement over bagging.

5.4 Effect of Ensemble Size

As we have stated, most previous work agrees that boosting methods are fairly resistant to overfitting. Additionally, there is also a general agreement that the most important gain of boosting methods is given by the first few classifiers. These two arguments together support the use of an ensemble of fixed size of 50 base classifiers as reasonable, and that it is not likely that the size of the ensemble might contaminate the experimental results. Nevertheless, it is possible that for some of the problems the ensemble may be either overfitting or underfitting the data. So, we have performed an additional experiment where the size of the ensemble is obtained by a cross-validation strategy. The experiment is carried out using ADABOOST and the proposed NLBP method.

The method for selecting the number of classifiers for each problem is the cross-validation strategy presented in Zhang and Yu (2005). For each partition of each problem we perform a 5-fold cross-validation to obtain the optimal size of the ensemble within the range of $[10,100]$ classifiers. That is, we divide the training set into five partitions and train the ensemble with four partitions and test the error with the remaining one. This is repeated for each one of the five partitions, and the ensemble size is obtained as the average size of the 5 runs. Then, the algorithm is run with this optimal size of the ensemble and using the whole training set. As the size of the ensemble is estimated for each partition, overfitting or underfitting is less likely to happen. Table 10 shows the results using the three base classifiers of the previous experiments. The parameters of the experiments are the same as the previous runs.

The first noticeable result shown in the table is that the errors of the algorithms using this method are similar to the errors for ensembles of 50 classifiers. These results support the general belief that boosting is quite resistant to overfitting. The results also show that NLBP is also hardly affected by overfitting. The differences among NLBP and ADABOOST are similar to the obtained using 50 classifiers. The significant win/loss record, 21/2, 12/4, and 20/1 for neural network, C4.5 and SVM as base classifiers respectively, is similar to the previous results using 50 classifiers: 20/0, 13/5, and

Data Set	5%			10%			20%		
	NLBP	Bagging	Ada.MH	NLBP	Bagging	Ada.MH	NLBP	Bagging	Ada.MH
anneal	0.0764	0.0909	0.0679	0.1499	0.1254✗	0.1664	0.2904	0.2490✗	0.2608✗
audiology	0.3372	0.3186	0.3425	0.4027	0.4053	0.4372✓	0.5566	0.5655	0.5796✓
autos	0.3851	0.4145✓	0.3975✓	0.4388	0.4278	0.4535	0.5639	0.5834✓	0.5785
balance	0.1030	0.1082	0.1018	0.1530	0.1462	0.1923✓	0.2954	0.2874	0.2928
breast-c	0.3301	0.3664✓	0.3699✓	0.3993	0.3993	0.4413✓	0.4329	0.4273	0.4322
card	0.2148	0.2261	0.2128	0.2458	0.2299	0.2661✓	0.3762	0.3588	0.3632✗
dermatol.	0.0989	0.1180✓	0.1191	0.1628	0.1809✓	0.2071✓	0.2787	0.2874	0.3279✓
ecoli	0.1899	0.2554✓	0.2196	0.2554	0.2643	0.3327✓	0.2994	0.3042	0.3351✓
gene	0.1844	0.1631✗	0.1608✗	0.2079	0.2020	0.2328✓	0.3305	0.3273	0.3395
german	0.2836	0.3142✓	0.3170✓	0.3156	0.3144	0.3436	0.3842	0.3876	0.4096✓
glass	0.3804	0.4037✓	0.3823	0.4561	0.4458	0.4925✓	0.4804	0.4953	0.5168✓
glass-g2	0.2564	0.2761	0.2699✓	0.3375	0.3349	0.3769✓	0.3242	0.3450	0.3254
heart	0.2600	0.2748✓	0.2674	0.3407	0.3185	0.3829✓	0.3741	0.3711	0.3689
heart-c	0.2497	0.2742✓	0.2536	0.3199	0.3046	0.3351	0.4040	0.3815	0.3947
hepatitis	0.2232	0.2555✓	0.2542✓	0.2943	0.2957	0.3345✓	0.3200	0.3355	0.3575✓
horse	0.3632	0.3918	0.3873	0.3973	0.4247✓	0.4297	0.4764	0.4918	0.5165✓
ionosph.	0.1630	0.2268✓	0.1857	0.2376	0.2194	0.2877✓	0.3271	0.2986	0.3248
iris	0.0787	0.1040	0.0973	0.1173	0.1173	0.1773✓	0.3080	0.3360✓	0.3587✓
isolet	0.1107	0.1361✓	0.1448✓	0.1665	0.1912✓	0.1893✓	0.2671	0.3250✓	0.3094✓
labor	0.0912	0.1541✓	0.1860✓	0.2355	0.3054✓	0.2984	0.4457	0.4739✓	0.4739✓
led24	0.4010	0.4440✓	0.4390✓	0.5110	0.5060	0.5240	0.5870	0.6020	0.6120✓
letter	0.1839	0.2394✓	0.3168✓	0.3045	0.3801✓	0.3230✓	0.4038	0.4936✓	0.4922✓
liver	0.3501	0.3559	0.3414	0.3565	0.3443	0.3663	0.4249	0.4285	0.4354
lrs	0.1755	0.1869	0.1676	0.2140	0.2027	0.2249	0.3420	0.3529	0.3405
lymph.	0.2933	0.3378✓	0.3473✓	0.2933	0.3400✓	0.3716✓	0.4041	0.4554✓	0.4581✓
optdigits	0.0711	0.0821✓	0.0755✓	0.1212	0.1252	0.1442✓	0.2208	0.2673✓	0.2304✓
page-b	0.0855	0.0842	0.0897✓	0.1306	0.1410✓	0.1334✓	0.2335	0.2462✓	0.2424
pendigits	0.0625	0.0654✓	0.0660✓	0.1135	0.1165✓	0.1151✓	0.2171	0.2488✓	0.2219✓
phoneme	0.2259	0.2440✓	0.2335✓	0.2588	0.2748✓	0.2558	0.3235	0.3561✓	0.3409✓
pima	0.2719	0.2867	0.2899	0.3073	0.3089	0.3412✓	0.3849	0.3849	0.3896
primary-t	0.6011	0.6212	0.6141	0.6513	0.6595	0.6708	0.7096	0.7356✓	0.7203
promoters	0.2340	0.2472	0.2547	0.2547	0.2962	0.2887	0.4245	0.4472	0.4453
satimage	0.1521	0.1674✓	0.1692✓	0.1988	0.2106✓	0.2128✓	0.3009	0.3154✓	0.3058✓
segment	0.0811	0.0905✓	0.0876✓	0.1322	0.1389	0.1431✓	0.2390	0.2704✓	0.2372
sick	0.0761	0.0792✓	0.0745	0.1297	0.1257✗	0.1418✓	0.2313	0.2402✓	0.2294
sonar	0.2433	0.3106	0.3558✓	0.2914	0.3548✓	0.3414✓	0.3606	0.4010✓	0.4125✓
soybean	0.1318	0.1488✓	0.1280	0.1833	0.1719	0.1921	0.3101	0.3330✓	0.2977
vehicle	0.2525	0.2648✓	0.2624	0.2799	0.2730	0.3109✓	0.3801	0.3965✓	0.3868
vote	0.1195	0.1311	0.1140	0.1710	0.1766	0.2014	0.2427	0.2405	0.2519
vowel	0.2039	0.1756✗	0.2146	0.2499	0.2572	0.2731	0.3432	0.4390✓	0.3701✓
waveform	0.1808	0.1867	0.1822	0.2250	0.2233✗	0.2475✓	0.3102	0.3208✓	0.3115
yeast	0.4412	0.4522	0.4473	0.4747	0.4771	0.4699	0.5220	0.5448✓	0.5283
zip	0.0668	0.0769✓	0.0698✓	0.1207	0.1220✓	0.1295✓	0.2142	0.2374✓	0.2199✓
zoo	0.1760	0.2278	0.1900	0.1642	0.1840✓	0.1977✓	0.2535	0.2813✓	0.2773
win/loss		24/2	18/1		13/3	28/0		22/1	20/2

✓/✗ significantly worse/better than NLBP method using F test.

Table 7: Summary of test error results for ensembles with a neural network using 5x2cv with noise levels of 5%, 10%, and 20%.

Data Set	5%			10%			20%		
	NLBP	Bagging	AdaB	NLBP	Bagging	AdaB	NLBP	Bagging	AdaB
anneal	0.0722	0.0973✓	0.0922	0.1399	0.1817✓	0.1757✓	0.2702	0.3156✓	0.3200✓
audiology	0.3460	0.3027	0.3106	0.3717	0.3566✗	0.3593✗	0.5620	0.5195✗	0.5204✗
autos	0.3871	0.3413✗	0.3725	0.4214	0.3853	0.3911	0.5288	0.5307	0.5367
balance	0.1296	0.3567✓	0.2586✓	0.1718	0.2883✓	0.3082✓	0.3139	0.4198✓	0.4292✓
breast-c	0.3014	0.3007	0.3469	0.3455	0.3881✓	0.4028✓	0.3993	0.4182	0.4392✓
card	0.2029	0.1893	0.1942	0.2371	0.2313	0.2386	0.3495	0.3551	0.3649
dermatol.	0.0891	0.1060	0.0951	0.1563	0.1836✓	0.1776✓	0.2311	0.2475✓	0.2503✓
ecoli	0.1976	0.2125✓	0.2167	0.2494	0.2923✓	0.2976✓	0.2881	0.3298✓	0.3500✓
gene	0.1542	0.1312✗	0.1451	0.2023	0.1746✗	0.1996	0.3242	0.2905✗	0.3227
german	0.2916	0.2912	0.2974	0.3126	0.3320	0.3292	0.3796	0.3904	0.4086✓
glass	0.3841	0.3710	0.3729	0.4393	0.4047	0.4112	0.4738	0.4467	0.4645
glass-g2	0.2627	0.2418	0.2455	0.3350	0.2982	0.3153	0.3316	0.3168	0.3608
heart	0.2445	0.2607	0.2652	0.2941	0.3200	0.3437✓	0.3378	0.3667	0.3830
heart-c	0.2311	0.2583✓	0.2583✓	0.2921	0.3066	0.3175✓	0.3536	0.3960✓	0.3987
hepatitis	0.2259	0.2296	0.2308	0.3021	0.2763✗	0.2983✗	0.2956	0.3047	0.3073
horse	0.3637	0.3522	0.3610	0.3923	0.3995	0.4099	0.4698	0.5066	0.5094✓
ionosph.	0.1459	0.1305	0.1464	0.1715	0.1881	0.1898	0.2706	0.2957	0.3031
iris	0.1027	0.1094	0.1027	0.1147	0.1707✓	0.1720✓	0.3173	0.3560✓	0.3667✓
isolet	0.1152	0.2102✓	0.1706✓	0.1646	0.2489✓	0.2189✓	0.2635	0.3215✓	0.3202✓
labor	0.1050	0.1085	0.1117	0.1863	0.1898	0.1792	0.3576	0.3825	0.3785
led24	0.4350	0.4120	0.4150	0.5270	0.4850	0.4970	0.5860	0.5480	0.5610
letter	0.1645	0.2106✓	0.1893✓	0.2137	0.2499✓	0.2502✓	0.3222	0.3709✓	0.3782✓
liver	0.3768	0.3698	0.3565	0.3843	0.3663	0.3722	0.4493	0.4337	0.4482
lrs	0.1737	0.1936	0.1959✓	0.2098	0.2219✓	0.2174	0.3081	0.3326	0.3296
lymph.	0.2838	0.2838	0.2865	0.2906	0.3203	0.3230	0.3460	0.3851✓	0.4081✓
optdigits	0.0700	0.0873✓	0.0716	0.1207	0.1352✓	0.1244	0.2223	0.2310✓	0.2311✓
page-b	0.0794	0.0842✓	0.0849✓	0.1311	0.1400✓	0.1389✓	0.2312	0.2528✓	0.2526✓
pendigits	0.0588	0.0680✓	0.0614✓	0.1080	0.1167✓	0.1154✓	0.2118	0.2247✓	0.2245✓
phoneme	0.2110	0.1814	0.1716✗	0.2563	0.2075✗	0.2245	0.3247	0.3007✗	0.3340
pima	0.2794	0.2857	0.2927	0.3169	0.3372	0.3331	0.3818	0.3974	0.4083✓
primary-t	0.5941	0.6135	0.6064	0.6430	0.6519	0.6814	0.6902	0.7126	0.7338✓
promoters	0.2358	0.2113	0.1887	0.2793	0.2264✗	0.2547	0.4151	0.4170	0.4170
satimage	0.1385	0.1476✓	0.1391	0.1833	0.1903✓	0.1860✓	0.2849	0.2853	0.2864
segment	0.0820	0.0827	0.0887✓	0.1345	0.1448✓	0.1483✓	0.2446	0.2549	0.2613✓
sick	0.0871	0.0674✗	0.0720✗	0.1422	0.1290✗	0.1442	0.2387	0.2521	0.2818✓
sonar	0.3039	0.2692	0.2769	0.3433	0.3308	0.3260	0.3442	0.3577	0.3837✓
soybean	0.1256	0.1616✓	0.1529	0.1754	0.2234✓	0.2100✓	0.3051	0.3640✓	0.3643✓
vehicle	0.2645	0.3024✓	0.3111✓	0.2976	0.3303✓	0.3343✓	0.4047	0.4388✓	0.4310
vote	0.1154	0.1310	0.1297	0.1697	0.1904✓	0.1959	0.2349	0.2630✓	0.2731✓
vowel	0.1758	0.2085✓	0.1957	0.2244	0.2628✓	0.2554✓	0.3321	0.3667✓	0.3675✓
waveform	0.1821	0.2028✓	0.2045✓	0.2260	0.2432✓	0.2487✓	0.3142	0.3292	0.3364✓
yeast	0.4464	0.4580	0.4625✓	0.4791	0.4945	0.4988	0.5367	0.5566✓	0.5616✓
zip	0.0676	0.1107✓	0.0874✓	0.1201	0.1596✓	0.1404✓	0.2171	0.2390✓	0.2383✓
zoo	0.1859	0.1799	0.2016	0.1841	0.1702	0.2038	0.2653	0.2594	0.2713
win/loss		15/3	12/2		20/6	19/2		18/3	25/1

✓/✗ significantly worse/better than NLBP method using F test.

Table 8: Summary of test error results for ensembles with a C4.5 tree using 5x2cv with noise levels of 5%, 10%, and 20%.

Data Set	5%			10%			20%		
	NLBP	Bagging	AdaB	NLBP	Bagging	AdaB	NLBP	Bagging	AdaB
anneal	0.0757	0.1031✓	0.1238✓	0.1635	0.1682	0.2007✓	0.3114	0.3027	0.3575✓
audiology	0.3469	0.3522	0.5611✓	0.5620	0.4265✗	0.4310✗	0.6691	0.5761✗	0.6035
autos	0.3881	0.3803	0.4671✓	0.4643	0.4262	0.4418	0.5630	0.5463	0.5727
balance	0.1088	0.1264✓	0.1760✓	0.1626	0.1680	0.2003✓	0.3094	0.3146	0.3421✓
breast-c	0.3182	0.3602✓	0.4147✓	0.3727	0.3930✓	0.4301✓	0.4154	0.4273	0.4594
card	0.2180	0.2658✓	0.3049✓	0.2826	0.2907	0.3249✓	0.3875	0.3899	0.4160✓
dermatol.	0.1011	0.0896	0.1465✓	0.1667	0.1656	0.2197✓	0.2574	0.2579	0.3344✓
ecoli	0.1887	0.1839	0.2732✓	0.2476	0.2488	0.3226✓	0.2792	0.2798	0.3607✓
gene	0.1857	0.2050✓	0.3099✓	0.3482	0.2584	0.2869	0.3244	0.3707✓	0.4001✓
german	0.3000	0.3150✓	0.3150	0.3242	0.3310✓	0.3394	0.3780	0.3800	0.3872
glass	0.3729	0.3841	0.4028	0.4327	0.4439	0.4823	0.4682	0.4907✓	0.5252✓
glass-g2	0.2651	0.2652	0.2565	0.3276	0.3117✗	0.3351	0.3192	0.3266	0.3425
heart	0.2333	0.2459	0.2852✓	0.2830	0.3104✓	0.3622✓	0.3452	0.3763	0.3978✓
heart-c	0.2537	0.2596✓	0.3192✓	0.2914	0.3033	0.3576✓	0.3715	0.3854	0.4245✓
hepatitis	0.2194	0.2299	0.2284	0.2866	0.2724	0.2879	0.2866	0.3111	0.3278
horse	0.3709	0.3989✓	0.3989	0.4116	0.4220	0.4160	0.4846	0.4808	0.4879
ionosph.	0.1026	0.1066	0.1391✓	0.1738	0.1721	0.2165✓	0.2684	0.2712	0.3253✓
iris	0.0746	0.0760	0.1013	0.1027	0.1040	0.1414	0.3133	0.3013	0.3347
isolet	0.1101	0.1675✓	0.1721✓	0.1839	0.2144✓	0.2286✓	0.2893	0.3163✓	0.3375✓
labor	0.0979	0.0915	0.1224	0.2005	0.2246	0.2491✓	0.3510	0.3791	0.4179✓
led24	0.4040	0.4270	0.4730✓	0.5270	0.5120	0.5360	0.5910	0.5740	0.6010
letter	0.1859	0.1716✗	0.2432	0.2399	0.2286	0.2683✓	0.3507	0.3647✓	0.4168✓
liver	0.3519	0.3478	0.3762	0.3629	0.3588	0.3890	0.4209	0.4238	0.4592✓
lrs	0.1752	0.1789	0.2079✓	0.2554	0.2268	0.2931	0.3627	0.3439	0.4620
lymph.	0.2838	0.3392✓	0.3176	0.2906	0.4054✓	0.3325	0.3730	0.4297✓	0.4216
optdigits	0.0730	0.0975✓	0.1339✓	0.1276	0.1736✓	0.2368✓	0.2453	0.2857✓	0.3411✓
page-b	0.0865	0.0894	0.0919	0.1346	0.1495✓	0.1388	0.2381	0.2598✓	0.2396
pendigits	0.0605	0.0571✗	0.0997✓	0.1113	0.1092	0.1611✓	0.2171	0.2176	0.2375✓
phoneme	0.2155	0.1988✗	0.1990	0.2496	0.2333	0.2362	0.3171	0.3048	0.3086
pima	0.2755	0.2825	0.3253✓	0.3047	0.3250	0.3664✓	0.3750	0.3878	0.4255✓
primary-t	0.6094	0.6542✓	0.6537✓	0.6519	0.6955✓	0.6955✓	0.7209	0.7327	0.7339
promoters	0.2075	0.1981	0.2226	0.2528	0.2604	0.2812	0.4736	0.4208	0.4227
satimage	0.1500	0.1401✗	0.1630✓	0.1969	0.1877✗	0.2244✓	0.2947	0.2930	0.3498✓
segment	0.0915	0.0994✓	0.1251✓	0.1547	0.1545	0.1829✓	0.2464	0.2563✓	0.2856✓
sick	0.1037	0.0863✗	0.1127✓	0.1505	0.1439✗	0.1785✓	0.2424	0.2479	0.2969✓
sonar	0.2250	0.2914✓	0.3240✓	0.2798	0.3442✓	0.3442✓	0.3356	0.3846✓	0.4010✓
soybean	0.1707	0.1423	0.1699	0.2264	0.1994	0.2407	0.3719	0.3327✗	0.3857
vehicle	0.2456	0.2896✓	0.3104✓	0.2757	0.3187✓	0.3612✓	0.3913	0.4348✓	0.4875✓
vote	0.1117	0.1196	0.1449✓	0.1628	0.1825	0.1977	0.2290	0.2524	0.2763✓
vowel	0.1394	0.1366	0.1703✓	0.1994	0.2077	0.2642✓	0.3174	0.3123	0.3950✓
waveform	0.1955	0.1899	0.2118✓	0.2379	0.2340	0.2436✓	0.3178	0.3249	0.3295✓
yeast	0.4445	0.4507	0.4688✓	0.4725	0.4873✓	0.4965✓	0.5217	0.5368	0.5040
zip	0.0752	0.0896✓	0.1296✓	0.1305	0.1445✓	0.1801✓	0.2276	0.2418✓	0.2591✓
zoo	0.1701	0.1701	0.1743	0.1603	0.1603	0.1821	0.2358	0.2317	0.3031✓
win/loss		16/5	29/0		12/4	25/1		11/2	27/0

✓/✗ significantly worse/better than NLBP method using F test.

Table 9: Summary of test error results for ensembles with a SVM using 5x2cv with noise levels of 5%, 10%, and 20%.

Data Set	Neural network		C4.5		SVM	
	NLBP	ADABOOST.MH	NLBP	ADABOOST	NLBP	ADABOOST
anneal	0.0100	0.0100	0.0118	0.0102	0.0102	0.0278✓
audiology	0.2513	0.2611	0.2823	0.2504	0.2938	0.2779
autos	0.3082	0.3247✓	0.2867	0.2633	0.2945	0.3121
balance	0.0544	0.0502	0.0803	0.2038✓	0.0570	0.0614
breast-c	0.3336	0.3769✓	0.2762	0.3189✓	0.2944	0.3748✓
card	0.1487	0.1797✓	0.1548	0.1551	0.1603	0.2392✓
dermatol.	0.0273	0.0268	0.0301	0.0317	0.0323	0.0361
ecoli	0.1607	0.2006✓	0.1506	0.1768✓	0.1542	0.1863✓
gene	0.0977	0.1036✓	0.1295	0.0798✗	0.1158	0.1208
german	0.2724	0.2868	0.2668	0.2740	0.2892	0.2968
glass	0.3327	0.3505	0.3252	0.3019	0.3187	0.3393✓
glass-g2	0.2197	0.2466✓	0.2284	0.1841	0.2136	0.2858✓
heart	0.1963	0.2356✓	0.1852	0.2208✓	0.1845	0.2089✓
heart-c	0.1835	0.2238✓	0.1801	0.2172✓	0.1861	0.2043✓
hepatitis	0.2052	0.2156✓	0.1936	0.2001	0.2001	0.2000
horse	0.3412	0.3725✓	0.3297	0.3423	0.3330	0.3643✓
ionosph.	0.1282	0.1492✓	0.0638	0.0792✓	0.0980	0.0667
iris	0.0507	0.0454	0.0427	0.0534	0.0427	0.0493
isolet	0.0715	0.0673	0.0756	0.1258✓	0.0649	0.1358✓
labor	0.1432	0.1541✓	0.1192	0.1087	0.1468	0.2006✓
led24	0.3750	0.4130✓	0.4300	0.3810	0.3850	0.4290✓
letter	0.1734	0.1893	0.1172	0.1236	0.1022	0.1006
liver	0.3217	0.3414✓	0.3339	0.3327	0.3426	0.3518
lrs	0.1119	0.1266✓	0.1179	0.1364✓	0.1205	0.1251
lymph.	0.1757	0.2095✓	0.1757	0.1878	0.1932	0.2662✓
optdigits	0.0204	0.0187	0.0241	0.0222	0.0222	0.0221
page-b	0.0339	0.0340	0.0317	0.0303	0.0333	0.0411✓
pendigits	0.0107	0.0064✗	0.0090	0.0095	0.0059	0.0066
phoneme	0.1844	0.1614	0.1405	0.1096✗	0.1712	0.1546
pima	0.2471	0.2695✓	0.2482	0.2674	0.2375	0.2914✓
primary-t	0.5698	0.6064✓	0.5822	0.5970	0.6141	0.6366
promoters	0.2094	0.2227	0.2264	0.2245	0.2302	0.2189
satimage	0.1067	0.1097✓	0.0955	0.0938	0.0939	0.0965
segment	0.0433	0.0363	0.0382	0.0225✗	0.0371	0.0410
sick	0.0465	0.0270	0.0399	0.0151✗	0.0566	0.0344✗
sonar	0.2231	0.2519✓	0.2202	0.2375	0.2087	0.2452✓
soybean	0.0656	0.0706	0.0679	0.0647	0.0688	0.0726
vehicle	0.2002	0.2017	0.2445	0.2513	0.2185	0.2307
vote	0.0529	0.0620	0.0556	0.0556	0.0483	0.0579
vowel	0.0972	0.0717✗	0.0937	0.1301✓	0.0576	0.0456
waveform	0.1361	0.1493✓	0.1473	0.1685✓	0.1481	0.1639✓
yeast	0.4186	0.4146	0.4054	0.4408✓	0.4151	0.4320✓
zip	0.0153	0.0139	0.0206	0.0380✓	0.0160	0.0188✓
zoo	0.0592	0.0592	0.0789	0.0713	0.0474	0.0903✓
win/loss		21/2		12/4		20/1

✓/✗ significantly worse/better than NLBP method using F test.

Table 10: Summary of test error results for ensembles with a neural network, a C4.5 tree and a SVM as base classifiers and the size of the ensemble obtained by cross-validation.

23/0. We can say that the conclusions obtained using 50 classifiers also hold for the results obtained estimating the size of the ensemble by cross-validation.

Table 11 shows the average size of the ensembles. The most interesting result is the fact that most of the ensembles are in the interval between 20 and 30 classifiers. As we have stated, this agrees with most previous work that found little gain in terms of error after about 25 classifiers had been added. It is also interesting to note that, typically, both methods, NLBP and ADABOOST find similar sizes of the ensemble.

6. Conclusions and Future Work

In this paper we have presented a new approach for ensemble construction based on nonlinear projections of the original training instances. The projections are obtained by means of a neural network and are aimed to make the classification of difficult instances easier. This idea is taken from boosting methods, but in contrast with these methods our method does neither resample or weight the instances in the training set. This avoids the drawbacks of either resampling/reweighting, such as poor performance on small data sets and sensitivity to noise.

The only drawback of the proposed method is the necessity of training an additional neural network each time a new classifier must be added to the ensemble. Nevertheless, this network is only trained with the instances misclassified by the previous classifier, so the time needed for its training is much reduced.

We have compared the performance of this method against the performance of bagging, Arc-x4 and several boosting methods. The comparison was made using a fairly large set of real-world problems. Our method showed very good results in terms of test error that clearly outperformed bagging, Arc-x4, and different boosting algorithms. κ-error diagrams showed the ability of the proposed algorithm to improve diversity among classifiers without dramatically affecting accuracy. This improvement of performance has been assessed using ensembles of fixed size and ensembles of variable size obtained by cross-validation.

Additional experiments have been made to test the sensitivity to noise of this approach. These experiments have shown that NLBP is less sensitive to noise than boosting methods, and at least as good as bagging.

We believe that this work opens a new and interesting approach to ensemble construction. Based on the main idea of boosting, that is putting more effort into difficult instances, our approach tries to make the classification of these instances easier without disregarding the instances correctly classified so far. The advantages of this approach have been highlighted by the results presented in this paper. Moreover, an interesting research line opens in which we can devise alternative projection methods that take into account the classification of a subset of instances.

Our future work is mainly directed towards two goals. Firstly, we are working on developing a more theoretical view of our method, that may explain the good experimental results reported here. Secondly, we have experienced some problems with the use of a neural network for the non-linear projections, due to the facility with which the outputs of the hidden nodes of the network saturate, making the projection less efficient. We are working on alternative supervised non-linear projection that can improve the performance of the method.

Data Set	Neural network		C4.5		SVM	
	NLBP	ADABOOST.MH	NLBP	ADABOOST	NLBP	ADABOOST
anneal	16.9	16.8	20.0	18.9	17.4	19.3
audiology	17.7	19.4	25.5	20.5	18.2	14.1
autos	16.0	20.0	21.1	20.6	19.0	13.8
balance	21.6	20.7	21.6	23.0	23.0	21.5
breast-c	20.4	20.1	21.1	20.8	18.6	20.1
card	23.2	23.6	22.0	24.1	24.9	21.7
dermatol.	16.9	17.5	16.7	19.6	18.1	12.6
ecoli	20.5	20.8	20.5	21.9	19.7	20.0
gene	28.5	23.2	25.0	28.8	29.9	16.5
german	23.3	21.5	19.5	24.2	18.9	21.9
glass	18.5	20.1	21.0	22.8	18.0	22.3
glass-g2	19.8	21.5	20.8	22.1	19.6	15.1
heart	18.6	20.0	22.0	24.8	19.0	13.7
heart-c	21.4	22.1	20.8	22.5	19.0	14.6
hepatitis	18.7	18.9	19.6	21.2	18.3	20.5
horse	20.6	20.5	22.1	22.8	21.1	18.1
ionosph.	20.3	18.0	18.0	22.9	18.5	19.1
iris	15.5	16.2	16.8	17.1	15.5	16.7
isolet	28.5	27.9	46.0	49.2	15.7	11.3
labor	16.0	16.0	15.3	17.5	16.0	16.3
led24	20.0	16.4	24.0	17.9	17.5	12.8
letter	28.5	38.5	41.2	36.7	17.2	22.5
liver	23.3	21.0	18.5	23.5	16.0	22.0
lrs	19.0	18.9	19.2	23.0	19.5	13.7
lymph.	17.1	17.5	20.1	21.4	19.3	21.7
optdigits	24.4	26.2	30.5	35.4	15.9	14.8
page-b	23.4	21.6	23.0	21.9	21.0	20.4
pendigits	26.2	25.6	28.7	34.4	18.7	21.8
phoneme	20.0	25.2	21.5	31.9	17.0	22.3
pima	25.0	21.4	24.7	22.2	22.4	22.1
primary-t	21.0	10.8	13.9	11.3	11.4	10.4
promoters	16.5	17.0	16.0	22.1	14.0	12.8
satimage	23.3	26.4	26.6	33.4	16.4	20.2
segment	23.3	21.6	25.0	24.2	23.8	23.2
sick	15.5	22.4	19.7	23.8	15.0	20.6
sonar	19.5	19.2	22.0	23.6	22.0	14.3
soybean	20.1	20.6	18.0	22.1	20.7	14.6
vehicle	21.5	22.2	22.9	26.7	23.8	22.1
vote	18.4	18.6	17.9	20.5	19.5	19.6
vowel	24.0	25.0	27.9	30.7	22.8	22.1
waveform	23.0	25.5	28.4	33.9	16.8	17.1
yeast	22.6	24.5	29.0	22.0	22.3	21.9
zip	23.9	25.1	39.4	41.9	12.2	15.2
zoo	14.9	15.5	18.5	16.9	15.1	18.1

Table 11: Summary of sizes for ensembles with a neural network, a C4.5 tree and a SVM as base classifiers and the size of the ensemble obtained by cross-validation.

References

D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H. Lenz, editors, *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 1–7, 1995.

E. Alpaydin. Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, 11:1885–1892, 1999.

T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 2nd edition, 1984.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–142, July/August 1999.

L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996a.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996b.

L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley, CA, 1996c.

L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–824, 1998.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

N. H. Bshouty and D. Gavinsky. On boosting with polynomially bounded distributions. *Journal of Machine Learning Research*, 3:483–506, 2002.

Ch-Ch. Chang and Ch-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

K. Chen, L. Wang, and H. Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *Journal of Pattern Recognition and Artificial Intelligence*, 11(3):417–445, 1997.

K. Cherkauer. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, pages 15–21, 1996.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In R. L. de Mantarás and E. Plaza, editors, *Proceedings of the Eleventh Conference on Machine Learning ECML 2000*, pages 109–116, Barcelona, Spain, 2000. Springer.

D. G. T. Denison, C. C. Holmes, B. K. Mallick, and A. F. M. Smith. *Bayesian Methods for Nonlinear Classification and Regression*. Wiley Series in Probability and Statistics. John Wiley & Sons, West Sussex, England, 2002.

L. Diao, K. Hu, Y. Lu, and Ch. Shi. A method to boost support vector machines. In M-S. Chen, P. S. Yu, and B. Liu, editors, *Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 463–468, Taipei, Taiwan, 2002. Springer-Verlag.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000a.

T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15. Springer-Verlag, 2000b.

T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857: 1–15, 2000c.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 180–189. Morgan Kaufmann, San Francisco, 2000.

S. Dzeroski and B. Zenko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54:255–273, 2004.

G. Eibl and K-P. Pfeiffer. Multiclass boosting for weak classifiers. *Journal of Machine Learning Research*, 6:189–210, 2005.

A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53: 71–109, 2003.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.

J. Friedman, T. Hastie, and R. Tibshirani. Additice logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

N. García-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation*, 9(3):271–302, June 2005.

R. L. Gorsuch. *Factor Analysis*. Erlbaum, Hillsdale, NJ, USA, 1983.

A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

L. Hall, K. Bowyer, R. Banfield, D. Bhadoria, W. Kegelmeyer, and S. Eschrich. Comparing pure parallel ensemble creation techniques against bagging. In *Third IEEE International Conference on Data Mining*, pages 533–536, Melbourne, FL, USA, 2003.

S. Haykin. *Neural Networks – A Comprehensive Foundation*. Prentice – Hall, Upper Saddle River, NJ, 2nd edition, 1999.

S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

I. T. Jolliffe. *Principal Components Analysis*. Springer – Verlag, New York, NY, 1986.

H-Ch. Kim, S. Pang, H-M. Je, D. Kim, and S. Y. Bang. Pattern classification using support vector machine ensembles. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR´02)*, volume 2, pages 160–163, 2002.

E. Kleinberg. On the algorithmic implementation of stochastic discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):473–490, 2000.

R. Kohavi. *Wrappers for Performance Enhancement and Oblivious Decision Graphs*. PhD thesis, Department of Computer Science, Stanford University, Stanford, USA, 1995.

R. Kohavi and C. Kunz. Option decision trees with majority voting. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 161–169, San Francisco, CA, USA, 1997. Morgan Kaufman.

T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, third edition, 2001.

J. F. Kolen and J. B. Pollack. Back propagation is sensitive to initial conditions. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 860–867. Morgan Kaufmann Publishers, Inc., 1991.

L. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003.

L. I. Kuncheva. Combining classifiers: Soft computing solutions. In S. K. Pal and A. Pal, editors, *Pattern Recognition: From Classical to Modern Approaches*, pages 427–451. World Scientific, 2001.

L. I. Kuncheva. Error bounds for aggressive and conservative adaboost. In *Proceedings of MCS*, number 2709 in Lecture Notes in Computer Science, pages 25–34, Guilford, UK, 2003.

Y. LeCun, L. Bottou, G. B. Orr, and K-R. Müller. Efficient backprop. In G. B. Orr and K-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer-Verlag, 1998.

B. Lerner, H. Guterman, M. Aladjem, and I. Dinstein. A comparative study of neural networks based feature extraction paradigms. *Pattern Recognition Letters*, 20(1):7–14, 1999.

Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, November 2000.

D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In Douglas H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 211–218, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

L. Mason, P. L. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38:243–255, 2000.

C. J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1):33–58, July 1999.

R. Munro, D. Ler, and J. Patrick. Meta-learning orthographic and contextual models for language independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning*, pages 192 – 195, 2003.

D. W. Opitz. Feature selection for ensembles. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 379 – 384, Orlando, FL, USA, 1999. American Association for Artificial Intelligence.

D. Ortiz-Boyer, C. Hervás-Martínez, and N. García-Pedrajas. Cixl2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, 24:33–80, July 2005.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, Oct 2006.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–073, 2004.

D. Rumelhart, G. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*, pages 318–362. MIT Press, Cambridge, MA, 1986.

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

M. Sebban, R. Nock, and S. Lallich. Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. *Journal of Machine Learning Research*, 3:863–885, 2002.

M. Skurichina and R. P. W. Duin. Bagging and the random subspace method for redundant feature spaces. In J. Kittler and R. Poli, editors, *Proceedings of the Second International Workshop on Multiple Classifier Systems MCS 2001*, pages 1–10, Cambridge, UK, 2001.

A. Tsymbal, P. Cunningham, M. Pechinizkiy, and P. Puuronen. Search strategies for ensemble feature selection in medical diagnosis. In M. Krol, S. Mitra, and D. J. Lee, editors, *Proceedings of the Sixteenth IEEE Symposium on Computer-Bases Medical Systems CBMS'2003*, pages 124–129, The Mount Sinai School of Medicine, New York, USA, 2003. IEEE CS Press.

K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifier. *Connection Science*, 8(3–4):385–404, 1996.

A. Utsugi. Ensemble of independent factor analyzers with application to natural image analysis. *Neural Processing Letters*, 14(1):49–60, August 2001.

G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, August 2000.

M-H. Yand, N. Ahuja, and D. Kriegman. Face detection using mixtures of linear subspaces. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 70–77. IEEE Computer Society Washington, DC, USA, 2000.

G. Zenobi and P. Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In L. de Raedt and P. Flach, editors, *12th European Conference on Machine Learning (ECML 2001)*, LNAI 2167, pages 576–587. Springer–Verlag, 2001.

T. Zhang and B. Yu. Boosting with early stooping: Convergence and consistency. *The Annals of Statistics*, 33(4):1538–1579, 2005.

Z-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1–2):239–253, May 2002.