



Supervised subspace projections for constructing ensembles of classifiers [☆]

Nicolás García-Pedrajas ^{a,*}, Jesús Maudes-Raedo ^b, César García-Osorio ^b, Juan J. Rodríguez-Díez ^b

^a Department of Computing and Numerical Analysis, University of Córdoba, Spain

^b Department of Civil Engineering, University of Burgos, Spain¹

ARTICLE INFO

Article history:

Received 3 November 2010

Received in revised form 26 May 2011

Accepted 5 June 2011

Available online 8 July 2011

Keywords:

Ensembles of classifiers

Subspace methods

Supervised projections

Classification

ABSTRACT

We present a method for constructing ensembles of classifiers using supervised projections of random subspaces. The method combines the philosophy of boosting, focusing on difficult instances, with the improved accuracy achieved by supervised projection methods to obtain very good results in terms of testing error. To achieve both accuracy and diversity, random subspaces are created at each step, and within each random subspace, a supervised projection is obtained using only the misclassified instances. The next classifier is trained using all available examples, in the space given by the supervised projections.

The method is compared with AdaBoost and other ensemble methods, showing improved performance on a set of 32 problems from the UCI Machine Learning Repository. In terms of testing error, it obtains results that are significantly better than AdaBoost and random subspace method, using a decision tree as base learner. Furthermore, the robustness of the method in the presence of class label noise is above the results obtained with AdaBoost. A study performed using κ -error diagrams shows that the proposed method improves the results of boosting by obtaining diverse and more accurate classifiers. The decomposition of testing error into bias and variance terms shows that our method performs better than Bagging in terms of reducing the bias term of the error, and better than AdaBoost in terms of reducing the variance term of the error.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

An ensemble of classifiers consists of a combination of different classifiers, homogeneous or heterogeneous, to jointly perform a classification task [1,2]. A classification problem of L classes and n training observations consists of a set of instances with known class membership. Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ 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 $Y = \{1, \dots, L\}$. A multiclass classifier is a function $f: X \rightarrow Y$ that maps an instance $\mathbf{x} \in X \subseteq \mathbb{R}^d$ into an element of Y .

The task consists of finding 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 $\mathbb{C} = \{C_1, C_2, \dots, C_m\}$, each classifier performing a mapping of an instance vector $\mathbf{x} \in \mathbb{R}^d$ into the set of labels $Y = \{1, \dots, L\}$.

Techniques that use multiple models usually consist of two independent phases: model generation [3] and model combination [4]. Most techniques focus on obtaining a group of classifiers that are as accurate as possible and that disagree as

[☆] This work was supported in part by the Project TIN2008-03151 of the Spanish Ministry of Science and Innovation.

¹ <http://pisuerga.inf.ubu.es/ADMIRABLE/>.

* Corresponding author.

E-mail addresses: npedrajas@uco.es (N. García-Pedrajas), jmaudes@ubu.es (J. Maudes-Raedo), cgsosorio@ubu.es (C. García-Osorio), jjrodriguez@ubu.es (J.J. Rodríguez-Díez).

URL: <http://cibrg.org> (N. García-Pedrajas).

much as possible. These two objectives are somewhat conflicting, because more accurate classifiers must agree more frequently. Many methods have been developed to enforce diversity on the classifiers that form the ensemble [5]. Kuncheva [6] 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. The latter approach is likely 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 method is ADABOOST [7] and its numerous variants. It is based on adaptively increasing the probability of sampling the instances that are not correctly classified by the previous classifiers.

Bagging [8] is the most representative algorithm of the second group. Bagging (after *Bootstrap aggregating*) generates different bootstrap samples from the training set. Several empirical studies have shown that ADABOOST is able to reduce the bias and variance components of the error [9–11]. Notably, Bagging seems to be more efficient than ADABOOST [11] in terms of reducing variance.

Although most ensemble techniques focus on obtaining classifiers that are as diverse as possible, Kuncheva and Whitaker [12] failed to establish a clear relationship between diversity and ensemble performance. Thus, in this work we first improve the accuracy of classifiers and subsequently aim for diversity promotion. Our experiments show that this approach yields good results.

The popularity of boosting methods is mainly due to the success of ADABOOST [13]. Boosting constructs an ensemble in a stepwise manner. At each step a new classifier is added to the ensemble. The basic idea is that the new classifier is trained on a distribution of the learning instances biased towards the most difficult instances. In this way, each instance has an associated weight that is higher if the instance has been misclassified by several of the previous classifiers. ADABOOST tends to perform very well for some problems, but it can perform very poorly for others. One of the sources of the bad behavior of ADABOOST is that, although it is always able to construct diverse ensembles, the individual classifiers tend to have large training errors in some problems. Moreover, ADABOOST usually performs poorly on noisy problems [11].

A different approach for constructing ensembles is the random subspace method (RSM) [14], which is rooted in the theory of stochastic discrimination. It shares some characteristics with Bagging, but it samples attributes [15] instead of instances. Ho [14] showed that the random subspace method improved the generalization error. The use of different spaces for ensemble construction has been extensive in recent research. In this work, we present a method based on the combination of ideas from boosting and from subspace methods.

The two major aims of constructing an ensemble of classifiers are accuracy and diversity. Although promoting diversity is usually considered a necessity, reported experiments have shown that accuracy may be more relevant than diversity for explaining ensemble performance [16]. Thus, the first concern of our method is to improve classifier accuracy using a supervised projection. This supervised projection is constructed using only the instances that were misclassified in the previous classifier that was added to the ensemble.¹ In this way, although each classifier is constructed using all of the training examples, it focuses on difficult instances. Importantly, the use of this method alone decreases diversity [17] with respect to other ensemble approaches. To avoid this negative effect, the supervised projections are performed using disjoint subspaces of the inputs, which increases diversity while maintaining the high accuracy of the individual classifiers.

Our method uses random subspaces to introduce diversity in the ensemble. This mechanism is combined with the boosting principle of focusing learning on the most difficult instances. Instead of using the standard scheme of boosting, which involves weighting the instances, we construct supervised projections determined using only the misclassified instances. Thus, learning is biased towards difficult instances and putting too much stress on those difficult instances is avoided.

The proposed method combines two different mechanisms to improve the performance of the ensemble:

1. Using a set of disjoint subspaces promotes diversity, as the supervised projection is carried out using different non-overlapping subsets of inputs.
2. The supervised projections determined using only misclassified instances promote accuracy in the most difficult instances. This method is less aggressive than boosting and produces an ensemble that is less sensitive to noise and outliers.

The proposed method could be seen as a kind of supervised random subspace method, as it combines the diversity introduced by choosing random subspaces with the supervised learning of the projections based on misclassified instances. Boosting is achieved by using supervised projections, instead of instance weights, to avoid putting too much stress on noisy or erroneous instances. Diversity is achieved by using random subspaces.

This paper is organized as follows: Section 2 presents the proposed model for ensemble construction; Section 3 reviews some work closely related to our proposal; Section 4 describes the experimental setup; Section 5 describes the results of the experiments; and Section 6 provides the conclusions of our work and suggestions for future lines of research.

¹ To avoid confusion we must emphasize that the axes of the linear projection are constructed using only misclassified instances and that the whole training set is transformed with this projection to obtain the next classifier. In the same way, for a nonlinear projection, the projection is constructed using only misclassified instances, but the whole training set is subsequently projected and used to train the next classifier.

2. Supervised subspace projection method

One of the sources of failure of boosting is putting too much stress on correctly classifying all of the instances. Outliers or noisy instances become too relevant in the training set and undermine the performance of the ensemble. In a previous work [1] we constructed ensembles projecting the input variables in a way that simplified the classification of misclassified instances. This projection was performed using the hidden layer of a multilayer perceptron.

Our new approach incorporates the advantages of boosting without its main drawbacks. The construction of the projection only accounts for instances that have been misclassified by a previous classifier, which permits the new classifier to focus on difficult instances. Nevertheless, as this classifier receives a uniform distribution of training instances, the sensitivity to noise and the effect of small datasets are greatly reduced. This method using the whole input space [17] reduces diversity, and makes the projection difficult to obtain, especially in the case of many inputs and few misclassified instances. To avoid this effect we perform the supervised projection using random disjoint subspaces instead of using all of the inputs.

The proposed method at each step t considers only the subset of instances, $S' \subset S$, misclassified by the classifier added in step $t - 1$. To train the classifier at step t , the input space is divided into several disjoint subspaces. We use the instances in S' to obtain either a linear or a non-linear supervised projection that focuses on misclassified instances into a subspace of the same size. The original training set is subsequently projected using these transformations, and the next classifier is trained on these projections using a uniform distribution of the instances. The proposed method is shown in Algorithm 1. In Algorithm 1, the projection is always performed using as inputs the original variables. Although the projection obtained in the previous step might be used as well, our experiments reported worse results when that method was used.

Algorithm 1: Algorithm for constructing the ensemble of classifiers using subspace linear/non-linear supervised projections.

Data: A training set $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, a base learning algorithm, $\mathbb{L}$, and the number of iterations T .

Result: The final classifier: $C^*(\mathbf{x}) = \operatorname{argmax}_{y \in Y} \sum_{t: C_t(\mathbf{x}) = y} 1$.

```

1   $C_1 = \mathbb{L}(S)$ 
  for  $t = 2$  to  $T$  do
2     $S' \subset S, S' = \{\mathbf{x}_i \in S: C_{t-1}(\mathbf{x}_i) \neq y_i\}$ 
3    Divide input space,  $\mathbb{F}$ , into  $s$  disjoint random subspaces,  $f_1, f_2, \dots, f_s$ ,
      such as  $\mathbb{F} = \{f_1 \cup f_2 \cup \dots \cup f_k\}$  and  $f_i \cap f_j = \emptyset, i \neq j$ 
      foreach  $f_i$  do
4        Obtain supervised linear/non-linear projection  $\mathbf{P}_i(\mathbf{x}^i)$  using  $S'$ ,
          where  $\mathbf{x}^i$  is instance  $\mathbf{x}$  in subspace  $f_i$ .
        end
5     $C_t = \mathbb{L}(\mathbf{P}(S))$ , where  $\mathbf{P}$  is the union of all  $\mathbf{P}_i$ 
  end

```

From the point of view of supervised projection algorithms, subspace projection has several advantages over projecting the whole input space. The algorithms are faster and more stable, as many inputs with few instances usually yield to ill-posed problems. For non-linear projections there is an additional advantage. The non-linear projection algorithms used in this paper obtain the projection of the training set and use a general regression network [18] to learn the projection when it must be used with unseen instances. In the original input space, which usually has many inputs, the network is hardly able to learn the projection and the unseen instances are projected using a gross approximation of the original projection performed by the algorithm. For the case of subspaces, the smaller number of inputs allows the regression network to approximate the projection more accurately.

For some iterations of the method, we may find that the previous classifier is highly accurate, making S' set too small to obtain meaningful projections. To avoid this negative effect, we set an arbitrary minimum for S' of 10% of the instances in the training set. If the number of misclassified instances is below this limit we add instances until this minimum is reached. The instances to be added are chosen based on their margin of classification, adding the instances with the smallest margin first. Thus, we add instances that, although correctly classified, have the highest risk of being misclassified.

2.1. Supervised projections

Quite often the intrinsic dimensionality of a dataset is much lower than the dimensionality of the dataset. In classification tasks the important characteristics of the data are those that define its intrinsic dimensionality. Therefore, the dimensionality of the dataset can be reduced so long as the intrinsic characteristics are preserved. In addition, the projection may reduce the effects of noise.

One of the most commonly used methods for linear projection is principal component analysis (PCA). PCA projects a dataset onto the directions that account for most of the variance in the dataset. Assuming a Gaussian distribution, these are the

directions with more information. The main drawbacks of PCA are that it is an unsupervised technique and does not take into account the class labels of the dataset. Alternatively, supervised projection methods take into account the class labels of the instances. They are designed to improve the separability of the different classes, and thus they should make the classification task easier. In the following we describe the linear and non-linear supervised projection methods relevant to our work.

2.1.1. Linear supervised projections

In this section, we provide a short review of the linear supervised projection methods used in our experiments.

Nonparametric Discriminant Analysis. Linear discriminant analysis (LDA) was first used for two classes. LDA has two serious disadvantages: (i) the Gaussian assumption over the class distribution of the data samples; and (ii) the dimensionality of the subspaces obtained is limited by the number of classes; at most $L - 1$ dimensions for a dataset with L classes. Nonparametric Discriminant Analysis (NDA) is an alternative to LDA that avoids these limitations. Fukunaga and Mantock [19] presented this nonparametric technique and illustrated it for the two class case. The formulation for the multiclass case is as follows [20]:

$$S_b^{\text{NDA}} = \sum_{i=1}^L P_i \sum_{\substack{j=1 \\ j \neq i}}^L \sum_{l=1}^{n_i} \frac{w_l^{(ij)}}{n_i} D_j(x_l^{(i)}) \cdot D_j(x_l^{(i)})^T, \quad (1)$$

$$S_w^{\text{NDA}} = \sum_{i=1}^L P_i \sum_{l=1}^{n_i} \frac{w_l^{(ii)}}{n_i} D_i(x_l^{(i)}) \cdot D_i(x_l^{(i)})^T, \quad (2)$$

$$w_l^{(ij)} = \frac{\min \left\{ d(x_l^{(i)}, x_{\text{kNN}}^{(i)})^\rho, d(x_l^{(i)}, x_{\text{kNN}}^{(j)})^\rho \right\}}{d(x_l^{(i)}, x_{\text{kNN}}^{(i)})^\rho + d(x_l^{(i)}, x_{\text{kNN}}^{(j)})^\rho}, \quad (3)$$

where ρ is a control parameter between zero and infinity, n_i is the number of instances in class i , P_i is the *a priori* probability of class i , $d(x_l^{(i)}, x_{\text{kNN}}^{(j)})$ is the distance from $x_l^{(i)}$ in class i to its k -th nearest neighbor (NN) in class j , $D_j(x_l^{(i)}) = x_l^{(i)} - M_j^k(x_l^{(i)})$ is the difference between the point $x_l^{(i)}$ and $M_j^k(x_l^{(i)}) = (1/k) \sum_{t=1}^k x_{\text{tNN}}^{(j)}$, the mean vector of the k nearest neighbors of $x_l^{(i)}$ in class j , its “local k -NN mean”.

NDA has two disadvantages: (i) parameters k and ρ are usually decided by rules of thumb, in the experiments we have used $k = 1$ and $\rho = 1.0$, and (ii) when the within-class scatter matrix in NDA is still with parametric form and the training set size is small, NDA will have a singularity problem.

Nonparametric weighted feature extraction. Nonparametric weighted feature extraction (NWFE) [20] was proposed to address the singularity problem encountered in NDA. The main ideas of NWFE include putting different weights on every sample to compute the “weighted means” and defining new nonparametric between-class and within-class scatter matrices as follows:

$$S_b^{\text{NWFE}} = \sum_{i=1}^L P_i \sum_{\substack{j=1 \\ j \neq i}}^L \sum_{l=1}^{n_i} \frac{\lambda_l^{(ij)}}{n_i} D_j(x_l^{(i)}) \cdot D_j(x_l^{(i)})^T, \quad (4)$$

$$S_w^{\text{NWFE}} = \sum_{i=1}^L P_i \sum_{l=1}^{n_i} \frac{\lambda_l^{(ii)}}{n_i} D_i(x_l^{(i)}) \cdot D_i(x_l^{(i)})^T, \quad (5)$$

$$w_{lk}^{(ij)} = \frac{d(x_l^{(i)}, x_k^{(j)})^{-1}}{\sum_{t=1}^{n_j} d(x_l^{(i)}, x_t^{(j)})^{-1}}, \quad (6)$$

$$\lambda_l^{(ij)} = \frac{d(x_l^{(i)}, M_j(x_l^{(i)}))^{-1}}{\sum_{t=1}^{n_i} d(x_t^{(i)}, M_j(x_t^{(i)}))^{-1}}, \quad (7)$$

where $D_j(x_l^{(i)}) = x_l^{(i)} - M_j(x_l^{(i)})$ is the difference between the point $x_l^{(i)}$ and $M_j(x_l^{(i)}) = \sum_{k=1}^{n_j} w_{lk}^{(ij)} x_k^{(j)}$, the “weighted mean” of $x_k^{(j)}$ in class j , where all the points are taken in consideration and not only the neighborhood of points as for NDA. Using the first k

nearest neighbors to estimate the local mean may result in loss of some information. NWFE proposes the weighted mean and uses weighted between-class and weighted within-class vector to improve NDA.

Regularization can be used to overcome the singular (problem *ill-posed*) or close to singular (problem *poorly posed*) matrix problems. In the experiments presented herein, the regularization process [21] was applied whenever the size of the dataset was smaller than four times the dimension. The new regularized version of the within-class scatter matrix is:

$$S_w^R = \alpha \text{diag}(S_w) + \beta \frac{\text{trace}(S_w)}{p} I + \gamma S_w, \quad (8)$$

where p is the dimensionality of the data and α , β and γ are mixing parameters with $0 \leq \alpha, \beta, \gamma \leq 1$ and $\alpha + \beta + \gamma = 1$. In the experiments $\alpha = \beta = \gamma = 0.33$, and this regularization was used both for NDA and NWFE.

Evolutionary discriminant analysis. Evolutionary discriminant analysis (EDA) [22] was developed to overcome one of the problems that appears when using supervised projection methods for classification. Although our objective is minimizing the classification error, the previous methods do so indirectly, minimizing certain criteria that should produce a better error. EDA is aimed at optimizing the training error in a more straightforward manner. EDA searches for linear projections of an original D dimensional space into a p dimensional space, $p < D$, of the form:

$$\bar{x}_j = \sum_{i=1}^D x_i w_{ij}, \quad (9)$$

for optimal classification purposes. A projection is evaluated using the number of misclassified instances as fitness value to be optimized. Once a projection is performed, the center of each class c , $\bar{\mathbf{m}}^c$, that is, the centroid of the instances belonging to class c , is obtained in the p -dimensional projected space, and the class of an instance is obtained from the closest projected mean:

$$C^*(\mathbf{x}) = \arg \min_c \left(\sum_{j=1}^p (\bar{x}_j - \bar{m}_j^c)^2 \right). \quad (10)$$

Instances are thus projected to place them closer to their own class mean than to others. The function to optimize is as follows:

$$J(\mathbf{w}) = \frac{1}{n} \sum_{\mathbf{x}} \mathfrak{D}(C(\mathbf{x}), C^*(\mathbf{x})), \quad (11)$$

where n is the number of instances, $C(\mathbf{x})$ is the class label of $\mathbf{x}$, and the value returned by $\mathfrak{D}(a, b)$ is 1 when a and b are different and 0 otherwise. Because the number of errors is a discrete non-differentiable quantity, the authors resort to an evolutionary strategy to minimize it. The complete algorithm is as follows:

- i) The dimension (p) of the projected space is chosen and fixed during the evolution. In our experiments the dimension of the projected space is the same as the dimension of the original space.
- ii) The size of the parent population (μ), the size of the offspring population (λ) and the size of the family (ρ) are chosen. An offspring's family is the subset of parents, which, after recombination, gives rise to the offspring. The notation $(\mu, \lambda | \rho)$ specifies the size of the populations, including the family, and the replacement scheme (comma replacement takes the best μ projections among the λ offspring as the new parent population).
- iii) The mutation step σ is chosen and fixed during the evolutionary process.
- iv) Initial weights are chosen at random in $[-0.5, 0.5]$ and the following steps are repeated until a prescribed number of generations without improvements are performed.
 - A group of λ weights, $w_{ij}^l, l = 1, \dots, \lambda$, is generated by the discrete recombination of ρ parent weights. Discrete recombination works as follows. For each offspring, ρ individuals are drawn at random from the parent population. Next, each allele of the offspring is equaled to the corresponding allele of one of the ρ parents, which are drawn at random from the family subset.
 - Each recombined weight is mutated by adding independent random deviations σ_{ij} to each weight component. These deviations are drawn from normal distributions $N(0, \sigma)$.
 - The comma strategy is used in the evolutionary strategy as generally recommended for continuous objectives. Therefore, a new parent population is created out of the μ best among the λ offspring weights.

2.1.2. Non-linear supervised projections

Although there is a broad range of unsupervised non-linear projection techniques, the number of supervised techniques is much lower. We have considered two methods: S-Isomap [23] and S-LLE [24].

Local linear embedding (LLE) [25] aims to preserve the distances of each point to the points in its neighborhood, thereby maintaining the local structure of the data. It assumes that the data manifold is locally linear and, hence, that each data point can be obtained as a linear combination of its nearest neighbors. First, it calculates the weights, W_{ij} , that best describe each point, $\mathbf{x}_i$ as a function of its neighborhood. In other words, it calculates the weights that minimize the error function:

$$\varepsilon(\mathbf{W}) = \sum_i \left\| \mathbf{x}_i - \sum_j W_{ij} \mathbf{x}_j \right\|^2, \quad (12)$$

subject to $\sum_j W_{ij} = 1$ and $W_{ij} = 0$ if $\mathbf{x}_j$ does not belong to the set of neighbors of $\mathbf{x}_i$. Second, it finds the projections, $\mathbf{y}_i$, associated with each $\mathbf{x}_i$ that best reproduce the same reconstruction weights, and, therefore, minimize the following error function:

$$\phi(\mathbf{Y}) = \sum_i \left\| \mathbf{y}_i - \sum_j W_{ij} \mathbf{y}_j \right\|^2. \quad (13)$$

Both problems can be solved using algebraic techniques.

In the supervised version of LLE [24] before determining the neighborhood of the points, the distances Δ are modified according to the following formula:

$$\Delta' = \Delta + \alpha \max(\Delta) A, \quad (14)$$

where A_{ij} is 0 when $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same class and 1 otherwise. Therefore, the distance between points in different classes is increased. Parameter α controls the amount to which class information should be incorporated. A value of $\alpha = 0$ reduces the method to the unsupervised version.

The algorithm produces the projection of the training set instances, so to obtain the projection of unseen instances, in our experiments we have used a general regression neural network (GRNN) [18], as suggested by some authors [23,26]. This is not the only solution, and other methods for mapping new instances could have been used [27–32].

3. Related work

The number of methods developed for constructing ensembles of classifiers using boosting-based approaches is huge. A survey of all these methods is outside the purpose of this paper. The interested reader can refer to the many reviews available in the literature [13,12]. In this section, we will only cite those methods based on projections of the original variables.

As stated in the previous sections, one of the first methods that used projections is the random subspace method (RSM) [14]. This method samples attributes [15] instead of instances, which is the approach used by Bagging. Ho [14] showed that the random subspace method improved the generalization error. Our method shares the idea of RSM, but instead of using a subset of instances in the design of each classifier, it uses all the features, grouped and projected in small subsets.

García-Pedrajas et al. [17] developed a method using supervised projections. However, that method did not use different subspaces, with the negative outcome of decreasing diversity with respect to other ensemble approaches. To avoid this negative effect, in this new present approach, the supervised projections are performed using disjoint subspaces of the inputs, which increases diversity while maintaining the high accuracy of the individual classifiers. Linear projections, both unsupervised and supervised, have also been used for streaming data [33], although with a different purpose.

Rodríguez et al. [16] developed an ensemble based on unsupervised projections of disjoint subspaces, called *Rotation Forest*. This method achieved good results in terms of testing error and diversity of the obtained classifiers. Rotation Forest enforces diversity by applying principal components analysis to different subspaces of the input space. Partially sharing its basis, our method aims to improve diversity and accuracy. Diversity is promoted by introducing different subspaces, and accuracy is promoted by using a supervised projection constructed from the misclassified instances.

Schlar and Rokach [34] used random projections on a lower dimensional space to construct an ensemble with the aim of promoting diversity. Maudes et al. [35] used the same approach for constructing ensembles of linear support vector machines. Wang and Dong [36] proposed a new rule-refinement scheme for fuzzy rules that is based on the maximization of fuzzy entropy on the training set.

Kuncheva [37] proposed a Bayes-optimal solution to the full-class set classification problem using a single classifier and the Hungarian assignment algorithm. Biggio et al. [38] focused on a new strategy to improve the robustness of linear classifiers to adversarial data manipulation, and experimentally investigated whether it can be implemented using two well known techniques for the construction of multiple classifier systems, namely, Bagging and the random subspace method.

4. Experimental setup

For assessment of the validity of our method we used 32 datasets from the UCI Machine Learning Repository [39]. The experiments were conducted using the 5×2 cross-validation set-up [40]. We performed five replications of a twofold cross-validation. In each replication the available data is partitioned into two random equal-sized sets, keeping the class distribution in both sets as much as possible. Each learning algorithm is trained on one set at a time and tested on the other set. The reported testing error is the average testing error of these 10 experiments.

Following [41] we carry out, in a first step, an Iman–Davenport test, to ascertain whether there are significant differences among all the methods. Then, pairwise differences are measured using a Wilcoxon test. The Wilcoxon test assumes limited commensurability. It is safer than parametric tests because it does not assume normal distributions or homogeneity of

Table 1
Summary of datasets.

Data set	Cases	Features			Classes	Inputs
		Continuous	Binary	Nominal		
anneal	898	6	14	18	5	59
arrhythmia	452	279	–	–	13	279
audiology	226	–	61	8	24	93
autos	205	15	4	6	6	72
card	690	6	4	5	2	51
dermatology	366	1	1	32	6	34
euthyroid	3163	7	18	–	2	44
gene	3175	–	–	60	3	120
german	1000	6	3	11	2	61
horse	364	7	2	13	3	58
hypo	3772	7	20	2	4	29
ionosphere	351	33	1	–	2	34
kr vs. kp	3196	–	34	2	2	38
labor	57	8	3	5	2	29
lrs	531	101	–	–	10	101
lymphography	148	3	9	6	4	38
mfeat-fac	2000	216	–	–	10	216
mfeat-fou	2000	76	–	–	10	76
mfeat-kar	2000	64	–	–	10	64
mfeat-pix	2000	240	–	–	10	240
mfeat-zer	2000	47	–	–	10	47
mushroom	8124	–	6	16	2	117
optdigits	5620	64	–	–	10	64
ozone1hr	2536	72	–	–	2	72
ozone8hr	2534	72	–	–	2	72
promoters	106	–	–	57	2	114
satimage	6435	36	–	–	6	36
sick	3772	7	20	2	2	33
sonar	208	60	–	–	2	60
soybean	683	–	16	19	19	82
texture	5500	40	–	–	11	40
waveform	5000	40	–	–	3	40

variance. Furthermore, empirical results [41] show that it is stronger than other tests. All comparison tables show the p -value of the Wilcoxon test, labeled p_w , the win/draw/loss record of every pair of algorithms, labeled s , and the p -value of a sign test over the win/loss record, labeled p_s . Table 1 shows a summary of the datasets. We have chosen subsets with at least 29 inputs to allow meaningful subspaces with all the parameter sets. We have studied the proposed model using as base learner a decision tree using the C4.5 algorithm.

Although a bad choice of parameters is likely to affect all of the methods in a similar manner, it may be argued that our method may be less sensitive to the parameter setting and thus, it benefits from the fact that we have not performed a selection of parameters. To avoid this issue, we employed 10-fold cross-validation when setting the values of the parameters. For each application of the decision tree, we obtained the best parameters from a set of different values. The source code of C4.5 used has two relevant parameters: number of trials and whether to use softening of thresholds. We tested 1 and 10 trials and the option of softening of thresholds trying all four possible combinations. The parameters chosen by cross-validation are used to learn the classifier using all the available training data. This process is repeated each time that a classifier is trained. Although this method does not assure an optimum set of parameters, it guarantees that a good set of parameters will be obtained in a reasonable amount of time.

The source code, in C and licensed under the GNU General Public License, used for all methods as well as the partitions of the datasets are freely available upon request from the authors.

5. Experimental results

This section reports the results of our method compared with other known algorithms, as well as a study of its behavior. We have performed experiments with ADABOOST [42] and our method using the supervised projections EDA, NDA, NWFE and LLE. In the following, we will refer to our method using a certain projection with only the projection name.

The most important parameter of the algorithm is the size of the subspaces to be constructed. This parameter can be chosen in two ways. We can set the number of subspaces and adjust the subspace size that suits that number depending on the number of inputs of the dataset, or we can set subspaces of fixed size and adjust the number of subspaces. For example, if we have a dataset with 15 features, we can set the number of features of each subspace to three, and then adjust the number of subspaces to five, accordingly. In the same way, we can set the number of subspaces to five, and

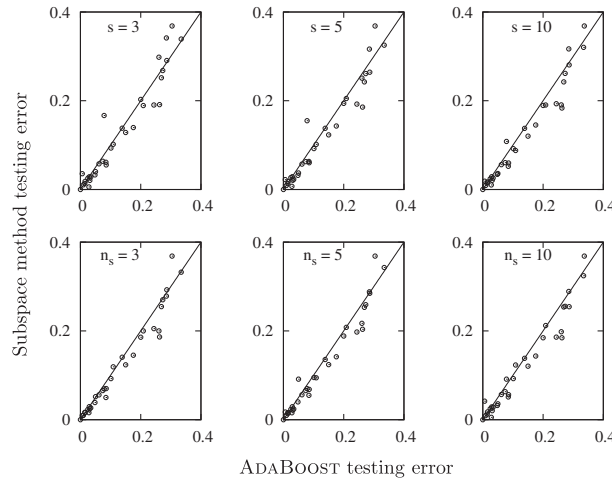


Fig. 1. Comparison of testing error for AdaBoost (x-axis) and the proposed method (y-axis) using NWFE.

then adjust the number of features in each subspace to three. The experiments shown test both alternatives. We have chosen a number of subspaces $n_s = \{3, 5, 10\}$, and a subspace size $s = \{3, 5, 10\}$. Each of the ensembles is composed of 30 classifiers.

In this section, the four projections, NDA, NWFE, EDA and LLE, are compared to AdaBoost. The results for NWFE are plotted in Fig. 1.² The figure represents the testing error of AdaBoost and our method using the six different subspace sizes. Points below the diagonal line show better performance of our method, and points above the diagonal line show better performance of AdaBoost. We can see that there are more points below the diagonal, and that the separation of these points from the diagonal is greater than that of the points above the diagonal. The overall performance of our proposal is better than AdaBoost. Furthermore, for several problems, namely audio, dermatology, ionosphere, lrs, mfeat-fac, mfeat-kar, mfeat-pix, mfeat-zer, promoters, sonar, soybean, texture and waveform, the differences are clear. On the other hand, for the few datasets where NWFE is performing worse than AdaBoost the differences are small, with the exception of gene dataset.

Fig. 2 presents the results of NDA projection method; Fig. 3 presents the results of EDA, and Fig. 4 presents the results for LLE. The behavior of these three projections is similar to that described for NWFE, with a significant advantage of the proposed methodology. For NDA projection, our method, with $n_s = 10$, clearly performs better than AdaBoost for several datasets, such as dermatology, german, ionosphere, lrs, lymph, mfeat-fac, mfeat-kar, mfeat-zer, promoters, sonar, texture and waveform. Similar behavior is observed for EDA projection for dermatology, ionosphere, labor, lrs, lymph, mfeat-fac, mfeat-kar, mfeat-pix, mfeat-zer, optdigits, promoters, sonar, texture and waveform datasets. For LLE we found a similar pattern with some differences. Although LLE performs significantly better than AdaBoost overall, it clearly performs worse for several datasets. One of the reasons of this worse performance of LLE, with respect to the other three projections, might be in the performance of LLE without subspaces, which is very poor. Although the use of subspaces is able to significantly improve the performance of LLE without subspaces, see Section 5.1, the improvement is not enough to beat AdaBoost in all cases.

Comparisons for all of the methods are shown in Table 2. The table shows that our methodology is able to achieve significantly better results than AdaBoost using the four different projections. In fact, the relative performance is very similar with NWFE, NDA, EDA and LLE, with slightly best results for NDA overall, and worst for LLE. The behavior of the number of subspaces, or subspace size, is similar, which supports the efficiency of our method and the robustness with respect to this parameter. The differences are significant at a 99% confidence level for many cases. Thus, as a summary we can state that our proposal is able to consistently beat AdaBoost. For the remaining experiments we will use as best representative of our method NDA projection with 10 subspaces ($n_s = 10$).

Speed considerations are difficult to measure, because we are evaluating not only an algorithm but also a certain implementation. However, because execution time is a relevant issue in any machine learning task, we have included a comparison of the different methods in terms of training time. To allow a fair comparison, we performed all the experiments in the same machine and with the same parameters. The differences in training time are shown in Fig. 5. The figure shows the comparison of running times of our method, with the four different methods of projection, and AdaBoost. For most problems, our method is slower, as it should be expected. However, for the case of linear projections, the differences are small, and our method is under reasonable time bounds. The non-linear projections have a worse behavior, but even in the worse case the time bounds are not dramatically high. Furthermore, this is not a very serious drawback of our method, as training is usually performed off-line. We must also take into account that, although the proposed approach is slower than AdaBoost

² The tables with the complete results for all the experiments shown in this paper are available. They are not included in the paper to avoid making it too long.

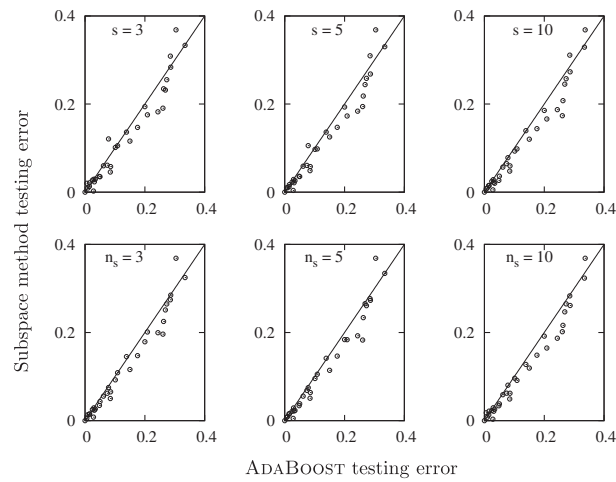


Fig. 2. Comparison of testing error for AdaBoost (x-axis) and the proposed method (y-axis) using NDA.

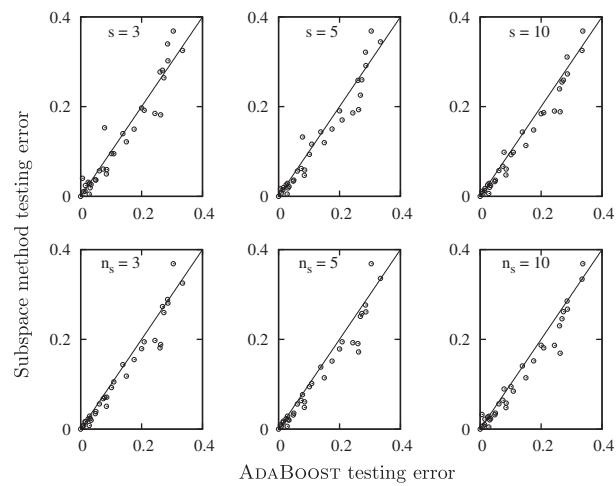


Fig. 3. Comparison of testing error for AdaBoost (x-axis) and the proposed method (y-axis) using EDA.

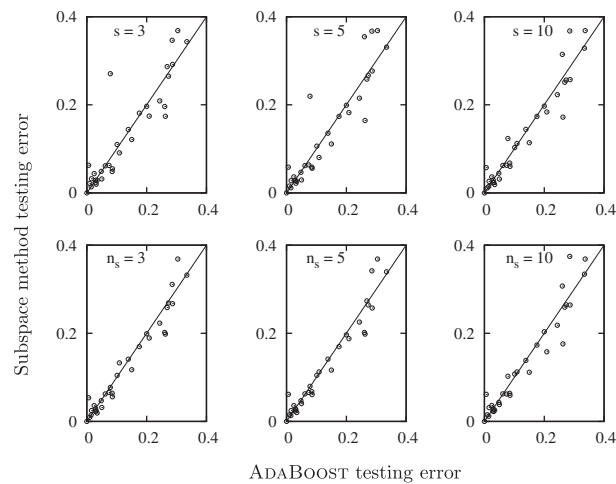
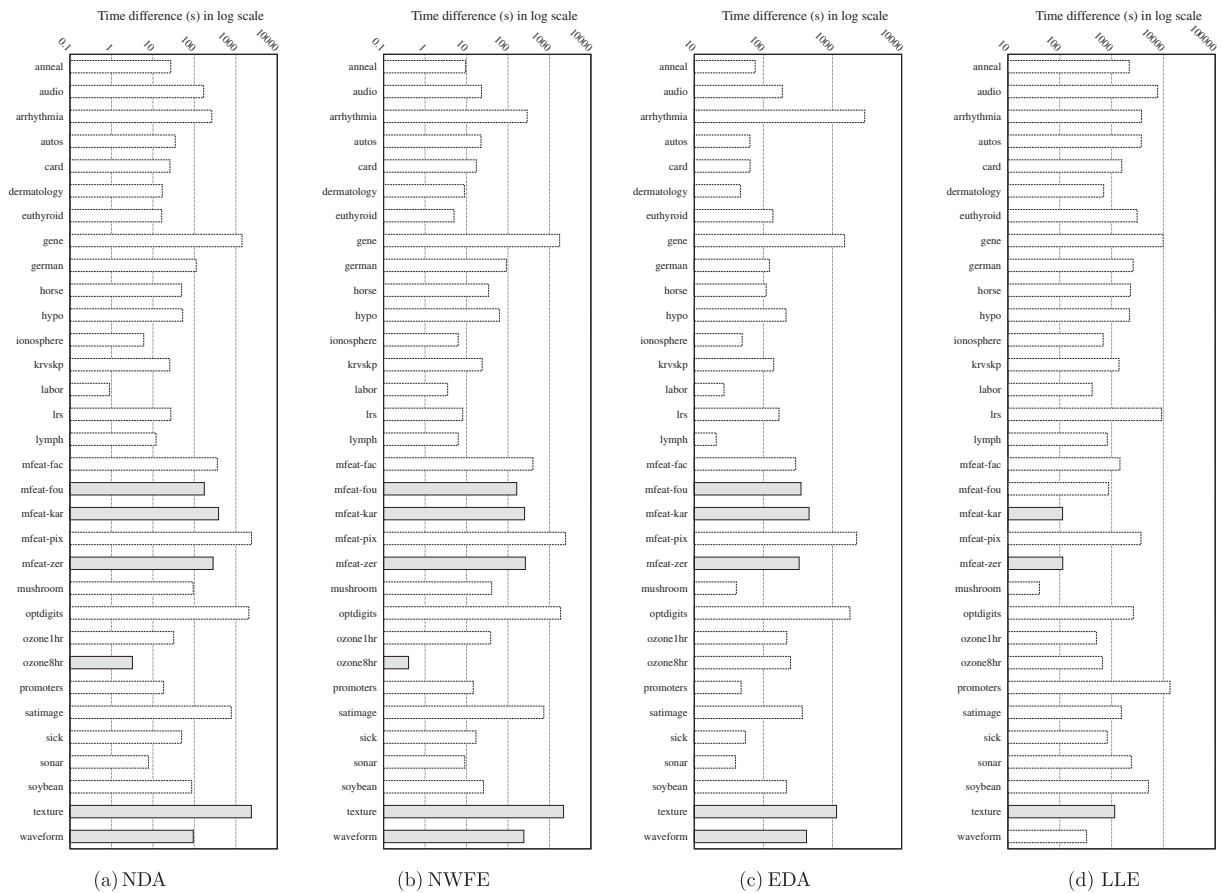


Fig. 4. Comparison of testing error for AdaBoost (x-axis) and the proposed method (y-axis) using LLE.

Table 2

Comparison of testing error results for AdaBOOST and the proposed method for the different projection methods.

		Number of subspaces			Subspace size		
		3	5	10	3	5	10
NWFE							
AdaBOOST	s	21/1/10	26/1/5	26/1/5	23/1/8	26/1/5	24/1/7
	p_s	0.0708	0.0002	0.0002	0.0107	0.0002	0.0033
	p_w	0.0286	0.0008	0.0003	0.0002	0.0002	0.0007
NDA							
AdaBOOST	s	25/1/6	26/1/5	26/1/5	26/2/4	28/1/3	28/1/3
	p_s	0.0009	0.0002	0.0002	0.0001	0.0000	0.0000
	p_w	0.0010	0.0001	0.0000	0.0000	0.0000	0.0000
EDA							
AdaBOOST	s	20/2/10	21/1/10	25/1/6	25/1/6	27/1/4	25/2/5
	p_s	0.0987	0.0708	0.0009	0.0009	0.0000	0.0002
	p_w	0.0624	0.0045	0.0002	0.0000	0.0000	0.0001
LLE							
AdaBOOST	s	17/1/14	22/1/9	20/1/11	22/1/9	19/1/12	19/1/12
	p_s	0.7201	0.0294	0.1496	0.0294	0.2810	0.2810
	p_w	0.1768	0.0700	0.0700	0.0159	0.0386	0.1118

Iman–Davenport test p -value: 0.000.**Fig. 5.** Difference between the execution time, on a logarithmic scale, of our method using NDA, NWFE, EDA and LLE projections and $n_s = 10$ against AdaBOOST. A filled bar means that our algorithm is faster, and an empty bar means that our algorithm is slower.

in training time, the differences in testing time are almost negligible, as the only difference is the projection of a single instance into different subspaces.

Table 3

Comparison of testing error results for AdaBoost and the proposed method for the different projection methods and a support vector machine as base learner.

		Number of subspaces			Subspace size		
		3	5	10	3	5	10
NWFE	<i>s</i>	26/0/6	27/0/5	27/0/5	25/0/7	24/1/7	26/0/6
	<i>p_s</i>	0.0005	0.0001	0.0001	0.0021	0.0033	0.0005
	<i>p_w</i>	0.0000	0.0000	0.0000	0.0001	0.0001	0.0000
NDA	<i>s</i>	3	5	10	3	5	10
	<i>p_s</i>	27/0/5	28/0/4	28/0/4	28/0/4	29/0/3	29/0/3
	<i>p_w</i>	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
EDA	<i>s</i>	19/0/13	22/0/10	24/0/8	23/0/9	26/0/6	23/0/9
	<i>p_s</i>	0.3771	0.0501	0.0070	0.0201	0.0005	0.0201
	<i>p_w</i>	0.0020	0.0003	0.0002	0.0004	0.0001	0.0004
LLE	<i>s</i>	18/0/14	19/0/13	19/0/13	20/0/12	19/0/13	17/1/14
	<i>p_s</i>	0.5966	0.3771	0.3771	0.2153	0.3771	0.7201
	<i>p_w</i>	0.0347	0.0378	0.0378	0.0123	0.0156	0.0240

Iman–Davenport test *p*-value: 0.000.

In the previous results, we have shown that when using as base learner a decision tree, our method improves the results of AdaBoost. As AdaBoost is very successful in boosting the performance of decision trees, this is a powerful argument in favor of our method. However, to test whether our method is still useful using other base learners, we have performed another set of experiments using as base learner a support vector machine (SVM). As SVMs are very sensitive to the learning parameters, we have performed a 10-fold cross-validation for obtaining their values. Each time an SVM was applied, we tried a linear kernel with $C \in \{0.1, 1, 10\}$, and a Gaussian kernel with $C \in \{0.1, 1, 10\}$ and $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, testing all the 21 possible combinations. The best set of parameters obtained by the cross-validation process was then applied to learn the classifier. Table 3 shows the comparison using SVM as base learner. The table shows that the results with SVMs are even more favorable to our method than the results using a decision tree.

5.1. Comparison with other methods

Our method shares common features with other ensemble methods. To present a fair evaluation of its efficacy, we will compare our method with them: (RSM) [14], Rotation Forest [16] and supervised projections [17]. RSM uses subspaces, as each member of the ensemble is trained with a different random subspace of the original input space. Rotation Forest shares our philosophy of using random subspaces to obtain projections, although it constructs unsupervised projections using principal components analysis. Finally, we compare our method with a previous work [17] that used supervised projections without subspaces to construct the classifiers. The comparison with our proposed method, using the partition of in-

Table 4Comparison of testing error results for our proposed method with 10 subspaces, $n_s = 10$, and RSM, Rotation Forest, the same approach without subspaces and Bagging.

		NWFE	NDA	EDA	LLE
RSM	<i>s</i>	29/1/2	28/1/3	29/1/2	21/1/10
	<i>p_s</i>	0.0000	0.0000	0.0000	0.0708
	<i>p_w</i>	0.0000	0.0000	0.0000	0.0534
Rotation Forest	<i>s</i>	20/1/11	22/1/9	22/1/9	13/1/18
	<i>p_s</i>	0.1496	0.0294	0.0294	0.4731
	<i>p_w</i>	0.1118	0.0056	0.0026	0.3833
Supervised projs.	<i>s</i>	25/1/6	25/0/7	26/0/6	25/0/7
	<i>p_s</i>	0.0009	0.0021	0.0005	0.0021
	<i>p_w</i>	0.0000	0.0004	0.0000	0.0001
Bagging	<i>s</i>	22/1/9	27/1/4	25/1/6	19/1/12
	<i>p_s</i>	0.0294	0.0000	0.0009	0.2810
	<i>p_w</i>	0.0041	0.0000	0.0001	0.0782

Iman–Davenport test *p*-value: 0.000.

put space in 10 disjoint subspaces as the representative of our method, $n_s = 10$, is shown in Table 4. For completeness, we have included Bagging method, as it is also widely used.

The comparison with RSM is similar to the comparison with AdaBoost. Our approach outperforms RSM for all cases at a confidence level of 95%. The comparison with Bagging also shows the same behavior, with the exception of LLE, which is not significantly better than Bagging. These two results, together with the previous comparison with AdaBoost, are a powerful argument in favor of our method. Regarding the comparison of the model with and without subspaces, our proposed method consistently and significantly outperformed the method without subspaces, supporting the hypothesis that the introduction of subspaces improves the performance of the ensemble.

We have also performed a comparison of our method with Rotation Forest. For Rotation Forest we chose the parameters given in [16]. The comparison shows that our method is able to improve the results of Rotation Forest using NDA as supervised projection, with a win/loss record of 22/9 and a p -value for the Wilcoxon test of 0.0056, and using EDA, with a win/loss record of 22/9 and a p -value of 0.0026. Using NWFE, our method outperformed Rotation Forest in terms of the win/loss record 20/11, although the differences were not statistically significant. For the case of LLE, the methods performed similarly, with a slightly worse performance of LLE, with a win/loss record of 13/18. The improvement of Rotation Forest performance is an argument in favor of using a supervised projection instead of the unsupervised projection introduced by Rotation Forest.

5.2. Noise tolerance

Several researchers have reported that boosting methods, such as AdaBoost, degrade their performance in the presence of noise [43,11]. Dietterich [13] tested this effect by introducing artificial noise in the class labels of different datasets and confirmed this behavior. In this section we study the sensitivity of our method to noise and compare it to the remaining ensemble algorithms: AdaBoost, Bagging, Rotation Forest and NDA without subspaces.

To add noise to the class labels, we follow the method presented in [13]. 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 from the set of incorrect labels uniformly. We use all of the datasets and three rates of noise, 5%, 10%, and 20%. With these three levels of noise we performed the experiments using the $5 \times 2cv$ setup and AdaBoost, Bagging, Rotation Forest, NDA without subspaces and NDA with 10 subspaces, $n_s = 10$, as the representative of our method.

Fig. 6 shows a summary of the increment in the testing error at the three different levels of noise. The results show that the two methods that do not consider miss-classification of instances, Bagging and Rotation Forest, are robust to noise, as it should be expected. On the other hand, the robustness of our method in the presence of noise is strong, and its behavior is superior to the performance of AdaBoost. For both classifiers, on average, our method increments its error about as much as

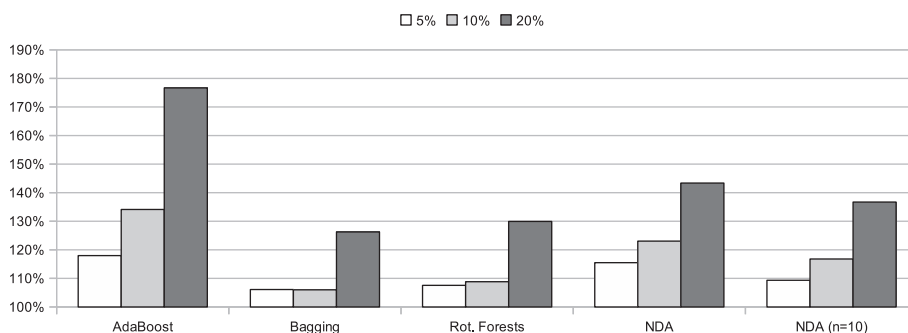


Fig. 6. Comparison of the increment in testing error in the presence of noise for AdaBoost, Rotation Forest, Bagging, NDA without using subspaces and NDA with 10 subspaces, ($n_s = 10$).

Table 5

Comparison of AdaBoost and NDA with 10 subspaces, ($n_s = 10$), for the three levels of noise.

		NDA ($n_s = 10$)			
		No noise	5%	10%	20%
AdaBoost	s	26/1/5	28/0/4	29/0/3	30/0/2
	p_s	0.0002	0.0000	0.0000	0.0000
	p_w	0.0000	0.0000	0.0000	0.0000

Iman–Davenport test p -value: 0.000.

the noise level, while AdaBoost ratio is more than 2:1 on average. Table 5 shows the comparison of our method and AdaBoost for the three levels of noise. In the presence of noise, the advantage of our method is even larger, outperforming AdaBoost for almost all datasets. This is an interesting result, as our proposal is almost as robust in presence of noise as other methods that do not use miss-classified instances.

5.3. κ -error diagrams

One way of understanding the behavior of ensemble methods is by means of κ -error diagrams [44,13]. These diagrams represent a point for each pair of classifiers. The x coordinate is a measure of the diversity of the two classifiers which is known as the κ measure. The y coordinate is the average error of the two classifiers on the test data. These two values are conflicting, due to the fact 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 diagram is useful for identifying the behavior of a method and whether it is more centered on either accuracy or diversity.

The κ measure is defined as follows: let us consider a problem with L classes, and let $\mathbf{C}$ be a $L \times L$ 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 κ statistic as follows:

$$\kappa = \frac{\theta_1 - \theta_2}{1 - \theta_2}, \quad (15)$$

where

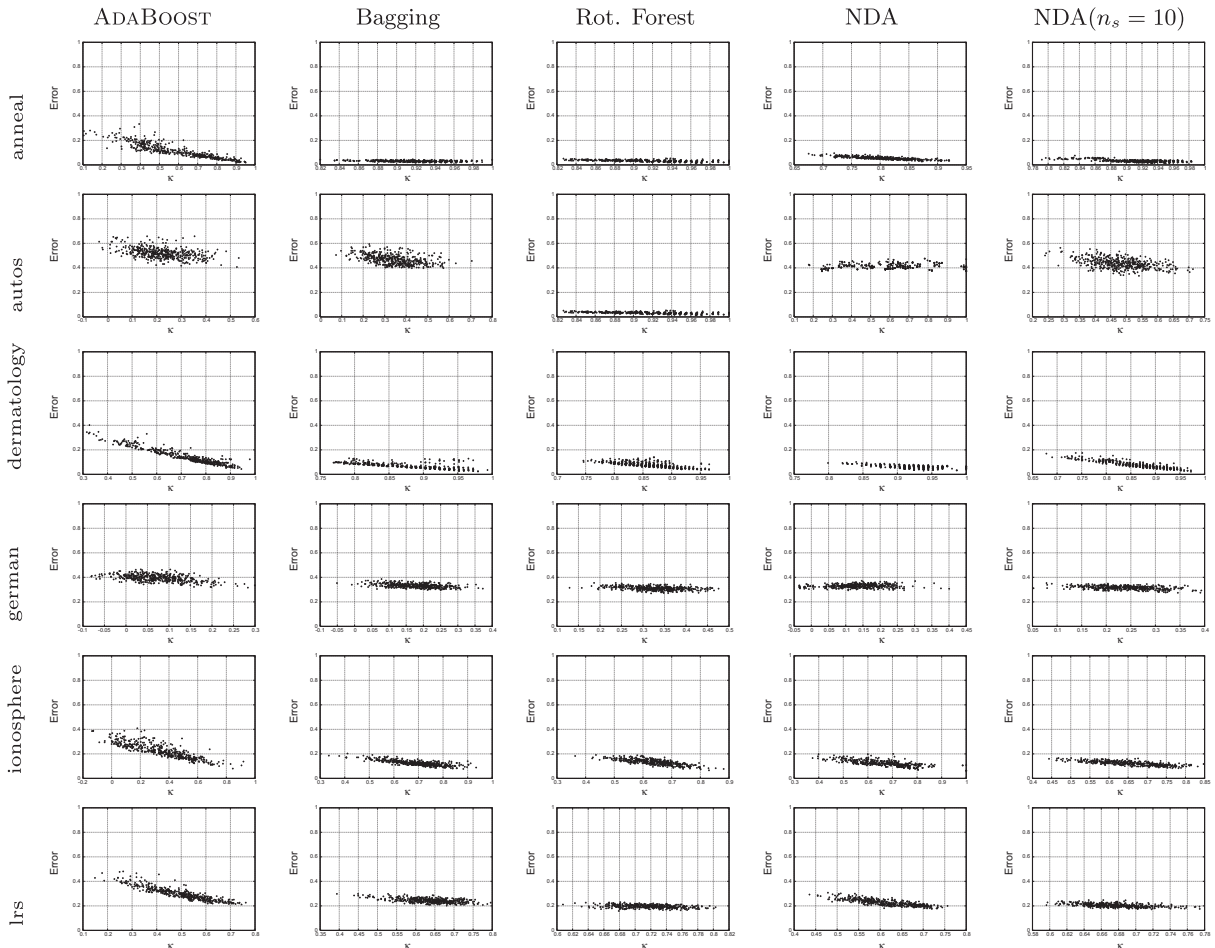


Fig. 7. κ -error diagram for the datasets using five different ensemble methods. A box marks the best method for each dataset.

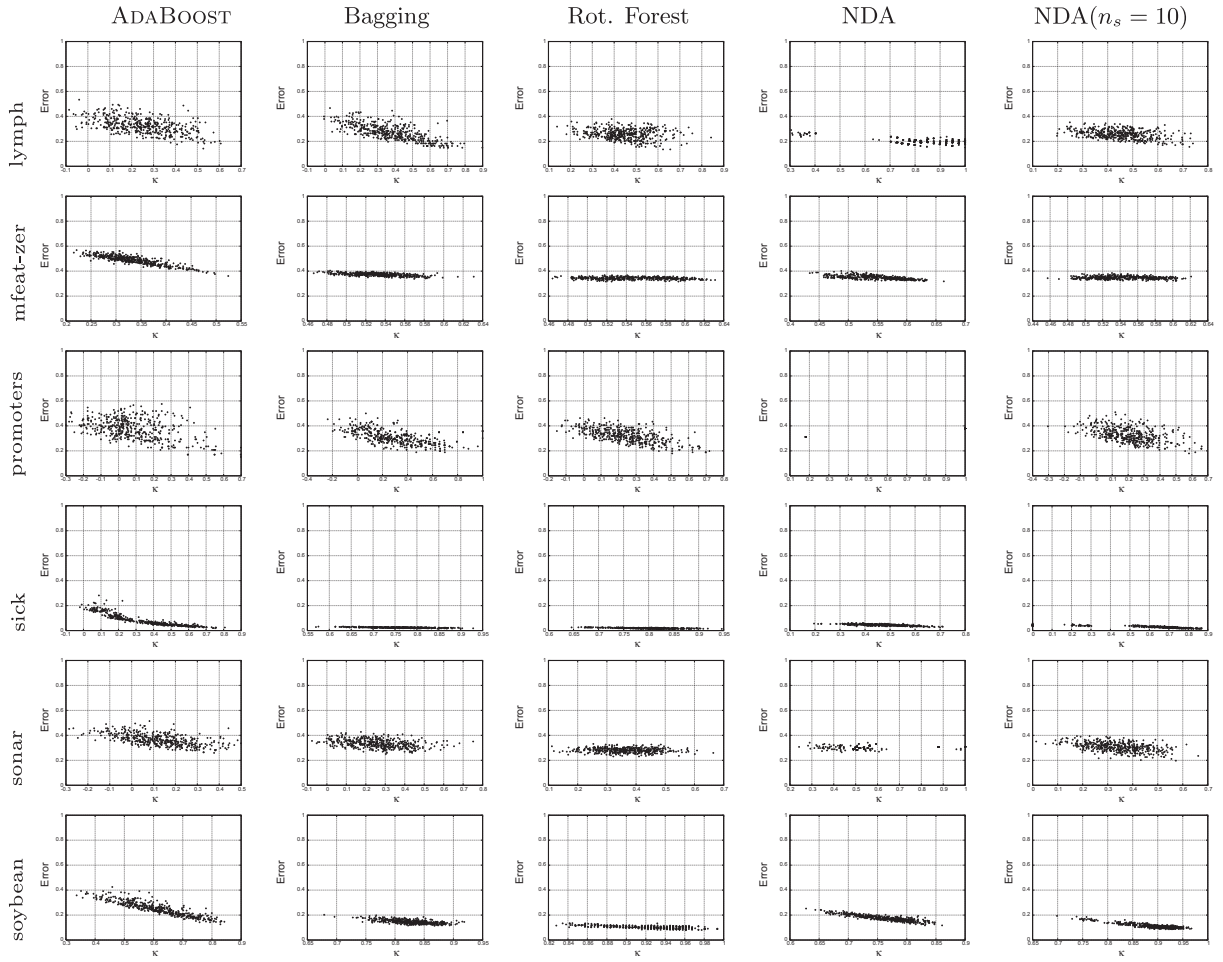


Fig. 8. κ -error diagram for the datasets using five different ensemble methods. A box marks the best method for each dataset.

$$\theta_1 = \frac{\sum_{i=1}^L C_{ii}}{n}, \quad \theta_2 = \sum_{i=1}^L \left(\sum_{j=1}^L \frac{C_{ij}}{n} \times \sum_{j=1}^L \frac{C_{ji}}{n} \right), \quad (16)$$

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

κ -error diagrams are constructed using NDA as representative of our approach. Figs. 7 and 8 shows κ -error diagrams for several datasets and ADABOOST, Bagging, Rotation Forest, NDA without using subspaces and NDA method with $n_s = 10$.

These results corroborate findings in previous reports [12,17] where no clear connection between diversity and ensemble performance was established. Furthermore, in [16], Rotation Forest was shown to outperform ADABOOST, but κ -error diagrams showed that Rotation Forest achieved excellent results by means of reducing individual classifier errors at the cost of less diverse ensembles. Our method follows a similar pattern of behavior. It is able to improve individual classifier accuracy at the cost of less diversity, and the overall performance of the ensemble benefits from that. However, the relevance of diversity is not an established fact as other authors [45] claim that diversity is the key of their proposals.

Furthermore, our results indicate that diversity might be more useful when base learners are mostly accurate. For instance, for the anneal dataset (see Fig. 7) Rotation Forest achieves the best results by improving the diversity of the classifier, but with the particularity that most of the individual classifiers are still highly accurate. This finding suggests that promoting diversity might be advisable provided that we have accurate classifiers.

This behavior is further asserted by the finding presented in Figs. 9–11. The figures show, for the same datasets, the accuracy of the classifiers as they are added to the ensemble and, for ADABOOST and NDA, the weighted accuracy using current instance weights. The supervised projections clearly allow our method to achieve more accurate classifiers than Bagging and, even more notably, than ADABOOST. For weighted accuracy, the behavior is more similar, with better accuracy observed for either our method or ADABOOST depending on the dataset. This result supports the finding that our method is able to achieve good testing error by means of improving individual classifier accuracy through the supervised projection approach.

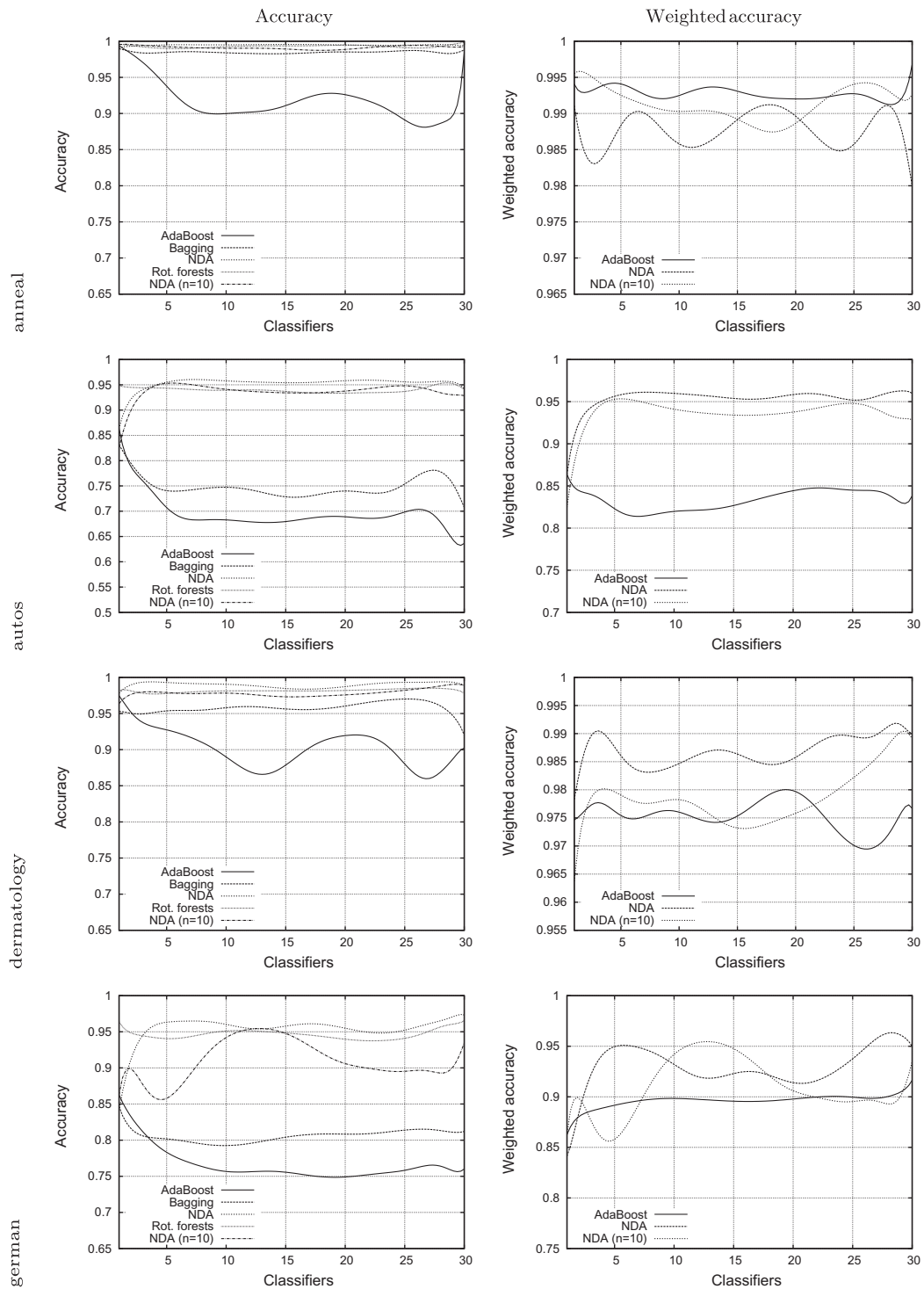


Fig. 9. Individual accuracy of the classifiers as the ensemble is constructed. Weighted accuracy is also shown for AdaBoost, NDA without subspaces and NDA with $n_s = 10$.

Again, the experiments show that putting too much effort on misclassified instances may harm the overall behavior of the ensemble.

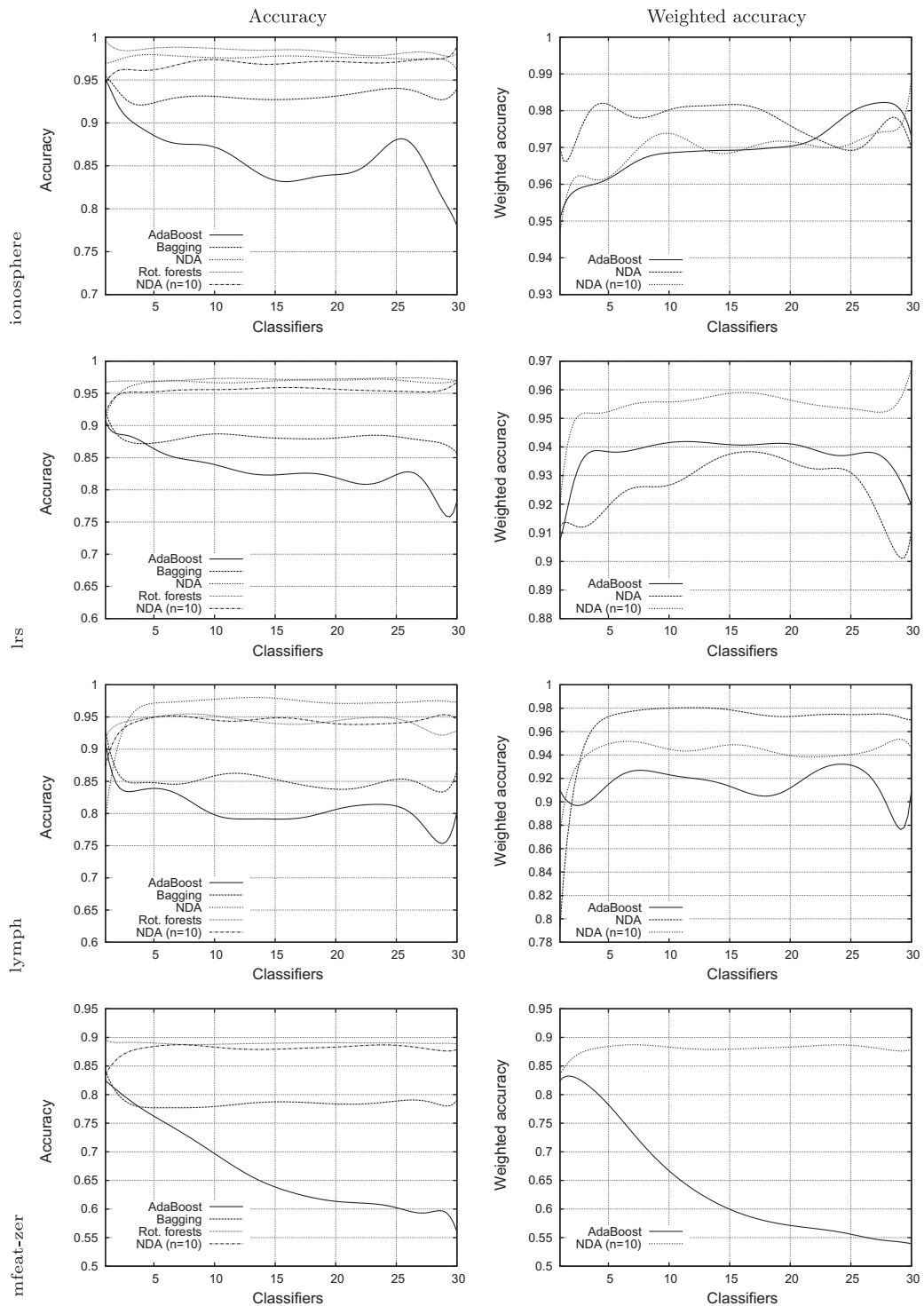


Fig. 10. Individual accuracy of the classifiers as the ensemble is constructed. Weighted accuracy is also shown for AdaBoost, NDA without subspaces and NDA with $n_s = 10$.

5.4. Bias and variance decomposition

Many papers studying ensembles analyze error performance in terms of two factors: *bias* and *variance*. The bias of a learning algorithm is the contribution to the error of the central tendency when it is trained using different data, whilst the

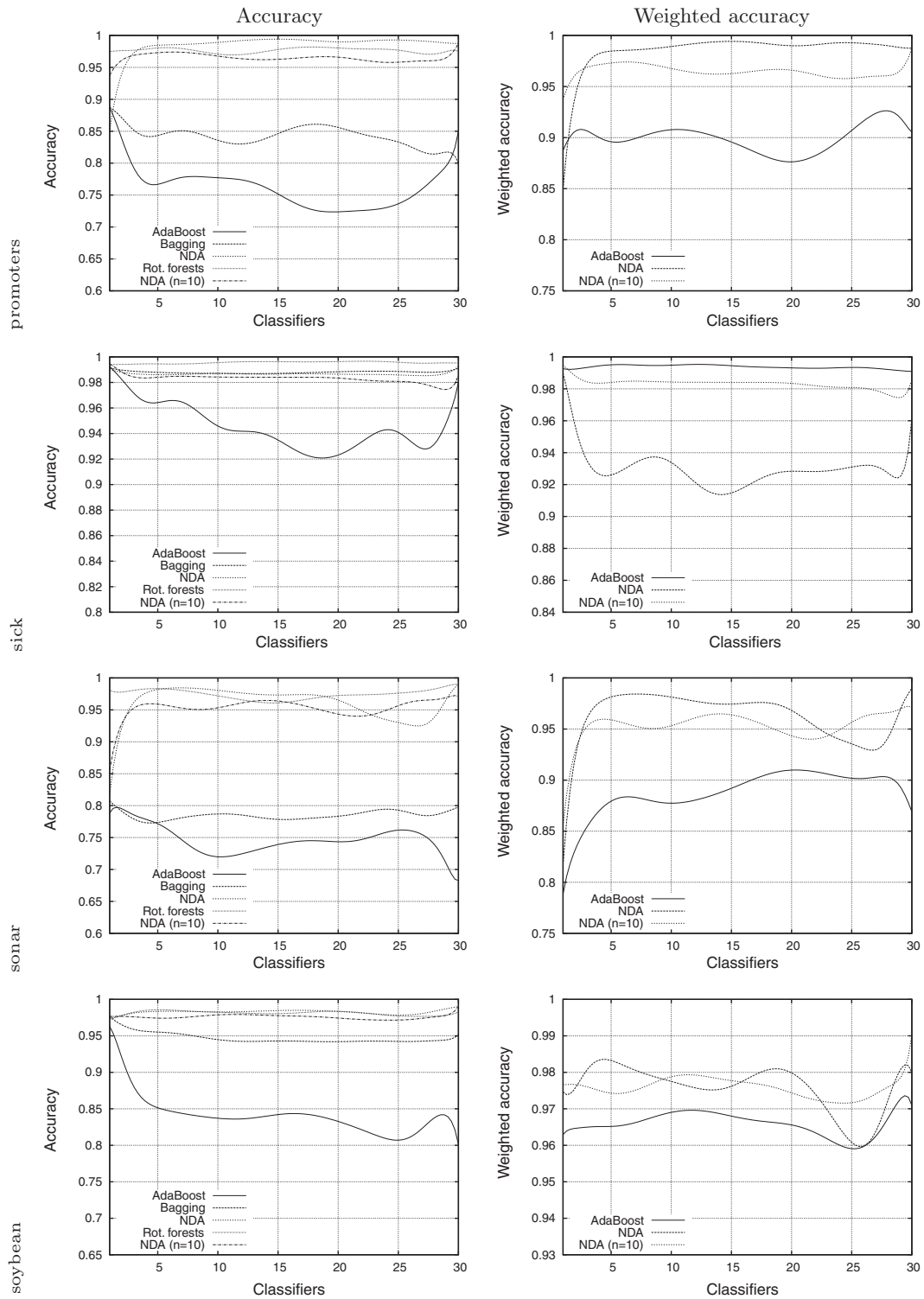


Fig. 11. Individual accuracy of the classifiers as the ensemble is constructed. Weighted accuracy is also shown for AdaBoost, NDA without subspaces and NDA with $n_s = 10$.

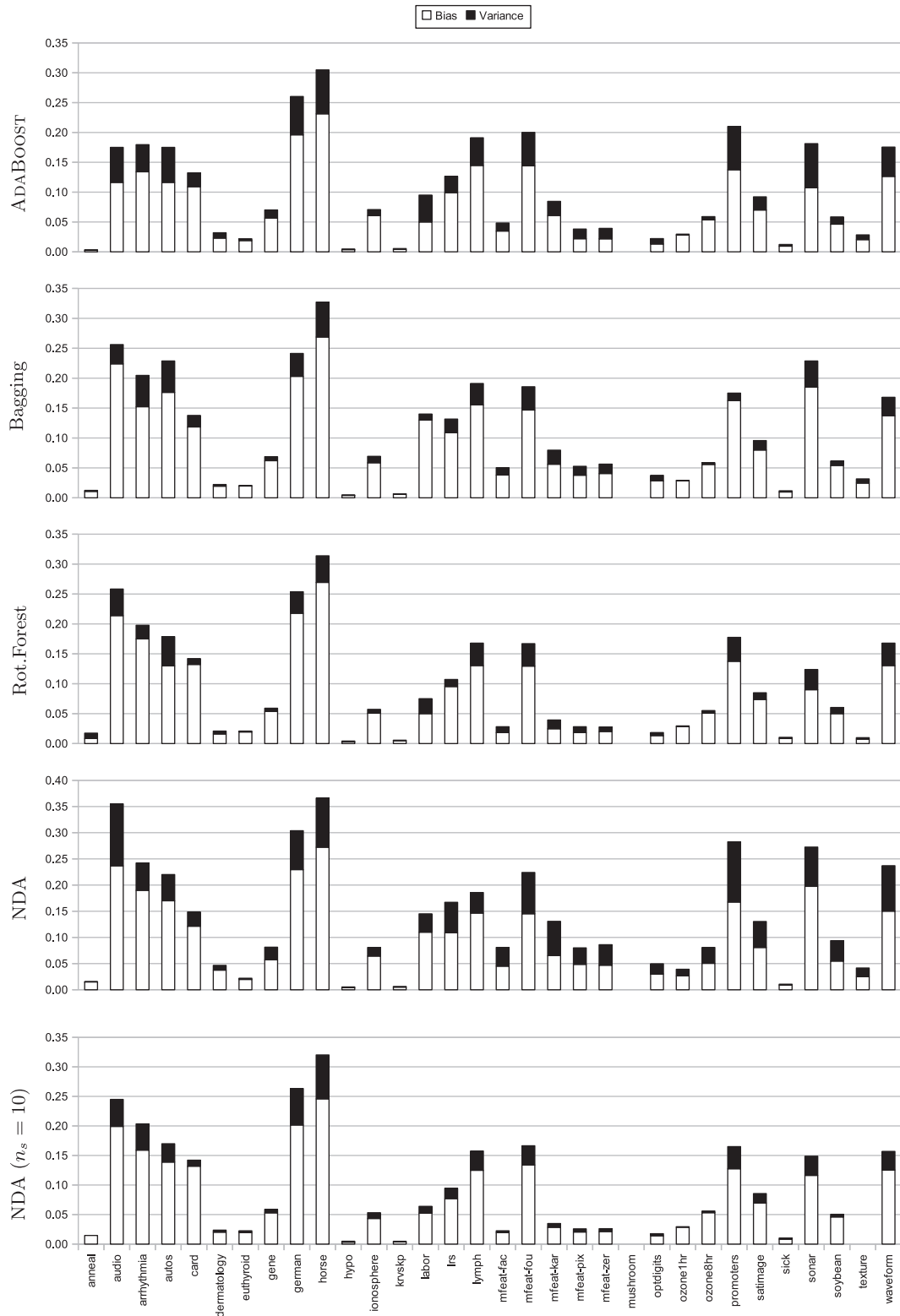


Fig. 12. Bias/variance decomposition for AdaBoost, Bagging, Rotation Forest, NDA without using subspaces, and NDA ($n_s = 10$) using Breiman's definition.

variance is the contribution of the error of the deviations from the central tendency. These two terms are evaluated with respect to a distribution of training sets $\mathcal{T}$ that are usually obtained by different permutations of the available data.

In addition, there is an *irreducible error* that is given by the degree to which the correct answer for an instance can differ from that for other instances with the same description. As this error cannot be estimated in most real world problems, the measures of bias and variance usually include this error.

Several authors have suggested different methods for decomposing the classification error in terms of the bias and the variance terms [46–48]. The proposals have different advantages and drawbacks [42].

Let us assume that the training pairs (x, y) are drawn from a test instance distribution X, Y , and that the classification of instance x by means of classifier $\mathcal{L}$ for a distribution of training datasets $\mathcal{T}$ is $\mathcal{L}(\mathcal{T})(x)$. Kohavi and Wolpert [47] proposed a method for estimating bias and variance terms of the error. However, their measures of bias and variance do not measure the extent to which each of these underlying quantities contribute to the error [42]. The irreducible error is included in the bias term. This estimation has the valuable property that the sum of bias and variance terms equals the total error.

The bias estimation of Kong and Dietterich [46] measures the probability of error of the central tendency of the learning algorithm. This measure is useful when comparing the quality of the central tendency of different classifiers, regarding other considerations such as the frequency or strength of such central tendency. Nevertheless, the estimation of the variance of Kong and Dietterich does not adequately measure the error due to deviations from the central tendency. Therefore, we have used the decomposition defined by Breiman [9]. Let us assume that the training pairs, (x, y) are drawn from a test instance distribution X, Y , and that the classification of instance x by means of classifier $\mathcal{L}$ for a distribution of training datasets $\mathcal{T}$ is $\mathcal{L}(\mathcal{T})(x)$.

$$\text{bias}_B = P_{(Y, X), \mathcal{T}}(\mathcal{L}(\mathcal{T})(X) \neq Y \wedge \mathcal{L}(\mathcal{T})(X) = C_{\mathcal{L}, \mathcal{T}}^0(X)),$$

$$\text{variance}_B = P_{(Y, X), \mathcal{T}}(\mathcal{L}(\mathcal{T})(X) \neq Y \wedge \mathcal{L}(\mathcal{T})(X) \neq C_{\mathcal{L}, \mathcal{T}}^0(X)),$$

Table 6

Bias/variance comparison using Breiman's definition for AdaBoost, Bagging, Rotation Forest, NDA without subspaces and NDA with ($n_s = 10$).

		Bagging	NDA	Rot. Forest	NDA ($n_s = 10$)
<i>Testing error</i>					
AdaBoost	s	12/2/18	2/1/29	23/1/8	23/1/8
	p_s	0.3616	0.0000	0.0107	0.0107
	p_w	0.0772	0.0000	0.0081	0.0065
Bagging	s		4/1/27	24/1/7	24/1/7
	p_s		0.0000	0.0033	0.0033
	p_w		0.0000	0.0004	0.0001
NDA	s			30/1/1	31/1/1
	p_s			0.0000	0.0000
	p_w			0.0000	0.0000
Rot. Forest	s				21/2/9
	p_s				0.0428
	p_w				0.0308
<i>Bias</i>					
AdaBoost	s	3/1/28	2/1/28	15/3/14	19/1/12
	p_s	0.0000	0.0000	1.0000	0.2810
	p_w	0.0000	0.0000	0.9107	0.6006
Bagging	s		9/3/20	27/1/4	27/1/4
	p_s		0.0614	0.0000	0.0000
	p_w		0.0330	0.0002	0.0000
NDA	s			28/1/3	28/1/3
	p_s			0.0000	0.0000
	p_w			0.0000	0.0000
Rot. Forest	s				16/2/14
	p_s				0.8555
	p_w				0.2823
<i>Variance</i>					
AdaBoost	s	27/1/4	6/3/23	29/1/2	28/1/3
	p_s	0.0000	0.0023	0.0000	0.0000
	p_w	0.0000	0.0001	0.0000	0.0000
Bagging	s		3/2/27	18/1/13	18/1/13
	p_s		0.0000	0.4731	0.4731
	p_w		0.0000	0.1937	0.1421
NDA	s			29/1/2	28/1/3
	p_s			0.0000	0.0000
	p_w			0.0000	0.0000
Rot. Forest	s				18/1/13
	p_s				0.4731
	p_w				0.1937

where the central tendency, $C_{\mathcal{L},\mathcal{T}}^o(X)$, for learner $\mathcal{L}$ over the distribution of training datasets $\mathcal{T}$ is the class with the greatest probability of selection for instance x by classifiers learned by $\mathcal{L}$ from training sets drawn from $\mathcal{T}$, and is defined as follows:

$$C_{\mathcal{L},\mathcal{T}}^o(x) = \max_y P_{\mathcal{T}}(\mathcal{L}(\mathcal{T})(x) = y). \quad (17)$$

These definitions have the advantage that the bias term is a direct measure of the contribution of the central tendency to the total error, and variance is a measure of the contribution of the deviations from the central tendency to the total error.

We divided our dataset into four randomly selected partitions [42] to estimate these two measures. We selected each partition in turn to be used as the test set and trained the learner with the other three partitions. This method was repeated ten times with different random partitions for a total of 40 runs of the learning algorithm. We must take into account this estimation of error is different from the $5 \times 2cv$ setup used in the previous sections, as the following tables will show the same general tendency of previous comparison tables but with small variations due to the different setup for obtaining the performance of each method.

The central tendency was evaluated as the most frequent classification for an instance. The error was measured as the proportion of incorrectly classified instances. This experimental setup guarantees that each instance is selected for the test set the same number of times, and alleviates the effect of the random selection of instances on the estimations. Fig. 12 shows the estimation of the bias and variance terms of the error for AdaBoost, Bagging, Rotation Forest, NDA without using subspaces, and NDA ($n_s = 10$).

Table 6 shows a comparison of the three methods in terms of testing error and bias and variance terms of the testing error. Several previous empirical studies have shown that AdaBoost reduces the bias and variance components of the error [9–11]. On the other hand, Bagging seems to be more efficient in reducing variance than AdaBoost [11]. Regarding these two standard classifiers our results corroborate that behavior as is shown in Fig. 12 and Table 6. Table 6 shows that Bagging performs significantly better than AdaBoost in terms of variance of the error and significantly worse in terms of bias of the error.

The decomposition provides insights into the behavior of our method. NDA achieves a bias reduction that is as good as that achieved by AdaBoost. In addition, it is able to improve variance reduction with respect to AdaBoost. This finding suggests that our method is able to join the advantages of Bagging, variance reduction and noise tolerance, with the advantage of AdaBoost, bias reduction. Regarding the comparison with Rotation Forest, our method achieves a better performance improving both terms of the error, bias and variance.

It is interesting to note that Friedman [49] argued that, for classification, the decomposition of error in bias and variance terms is multiplicative instead of additive. The exact form of the dependence makes the dominating factor for decreasing estimated error the variance term. Thus, reducing variance is expected to be more effective to reduce the error than reducing bias. Thus, the good behavior of our method might be then partially explained by its better ability to reduce variance than the other methods.

6. Conclusions and future work

In this paper we have presented a new method for constructing classifier ensembles based on the use of supervised projections of random subspaces using different supervised projection methods. The projections are constructed using only misclassified instances to focus the next classifier on the most difficult instances of the dataset.

The experiments have shown that our method improves the results of AdaBoost and RSM using a variety of supervised projection methods. We have thoroughly compared these methods using 32 datasets from the UCI machine learning repository with different features in terms of the number of classes, number of inputs and number of instances. Furthermore, our proposed method was shown to be superior to the same method without subspaces and is able to improve the results of Rotation Forest using NDA as the supervised projection algorithm. Notably, the proposed method is more robust in the presence of noise in the class labels.

Further study of the method has shown that it reduces the bias term of the error as consistently as AdaBoost, and reduces the variance term more than does AdaBoost. Our diversity analysis supports the hypothesis that classifier accuracy may be more relevant than diversity for the attainment of better ensembles. This conclusion is based upon the finding that our method outperforms AdaBoost by improving the accuracy of the individual classifiers, even when the diversity of the ensemble is worsened. Our method proved to be more accurate and less diverse than AdaBoost, and more accurate and as diverse as Bagging.

We are currently adapting our method to account for boosting weights of the instances for the supervised projections instead of using only the misclassified instances.

References

- [1] N. García-Pedrajas, C. García-Osorio, C. Fyfe, Nonlinear boosting projections for ensemble construction, *Journal of Machine Learning Research* 8 (2007) 1–33.
- [2] I. Koprowska, J. Poon, J. Clark, J. Chan, Learning to classify e-mail, *Information Sciences* 177 (2007) 2167–2187.
- [3] A. Ulaş, M. Semerci, O.T. Yildiz, E. Alpaydin, Incremental construction of classifier and discriminant ensembles, *Information Sciences* 179 (2009) 1298–1318.

- [4] J. Hansen, Combining predictors: comparison of five meta machine learning methods, *Information Sciences* 119 (1–2) (1999) 91–105.
- [5] T.G. Dietterich, Ensemble Methods in Machine Learning, in: J. Kittler, F. Roli (Eds.), *Proceedings of the First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, vol. 1857, Springer-Verlag, 2000, pp. 1–15.
- [6] L.I. Kuncheva, Combining classifiers: Soft computing solutions, in: S.K. Pal, A. Pal (Eds.), *Pattern Recognition: From Classical to Modern Approaches*, World Scientific, 2001, pp. 427–451.
- [7] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996, pp. 148–156.
- [8] L. Breiman, Bagging predictors, *Machine Learning* 24 (2) (1996) 123–140.
- [9] L. Breiman, Bias, Variance, and Arcing Classifiers, Technical Report, 460, Department of Statistics, University of California, Berkeley, CA, 1996.
- [10] R.E. Schapire, Y. Freund, P.L. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, *Annals of Statistics* 26 (5) (1998) 1651–1686.
- [11] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning* 36 (1/2) (1999) 105–142.
- [12] L. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, *Machine Learning* 51 (2) (2003) 181–207.
- [13] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, *Machine Learning* 40 (2000) 139–157.
- [14] T.K. Ho, The random subspace method for constructing decision forests, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (8) (1998) 832–844.
- [15] M. Skurichina, R.P.W. Duin, Bagging and the random subspace method for redundant feature spaces, in: J. Kittler, R. Poli (Eds.), *Proceedings of the Second International Workshop on Multiple Classifier Systems MCS 2001*, Cambridge, UK, 2001, pp. 1–10.
- [16] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (10) (2006) 1619–1630.
- [17] N. García-Pedrajas, C. García-Osorio, Constructing ensembles of classifiers using supervised projection methods based on misclassified instances, *Expert Systems with Applications* 38 (1) (2010) 343–359.
- [18] D.F. Specht, A general regression neural network, *IEEE Transactions on Neural Networks* 2 (6) (1991) 568–576.
- [19] K. Fukunaga, J. Mantock, Nonparametric discriminant analysis, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 5 (6) (1983) 671–678.
- [20] B.-C. Kuo, D.A. Landgrebe, Nonparametric weighted feature extraction for classification, *IEEE Transactions on Geoscience and Remote Sensing* 42 (5) (2004) 1096–1105.
- [21] B.-C. Kuo, L.W. Ko, C.-H. Pai, D.A. Landgrebe, Regularized feature extractions for hyperspectral data classification, in: *International Geoscience and Remote Sensing Symposium*, vol. 3, 2003, pp. 1767–1769.
- [22] A. Sierra, A. Echevarria, Evolutionary discriminant analysis, *IEEE Transactions on Evolutionary Computation* 10 (1) (2006) 81–92.
- [23] X. Geng, D.-C. Zhan, Z.-H. Zhou, Supervised nonlinear dimensionality reduction for visualization and classification, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 35 (6) (2005) 1098–1107.
- [24] D. de Ridder, R. Duin, Locally Linear Embedding for classification (Technical Report PH-2002-01), Technical Report, Pattern Recognition Group, Department of Imaging Science and Technology, Delft University of Technology, 2002.
- [25] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science* 290 (5500) (2000) 2323–2326.
- [26] Q. Zhao, D. Zhang, H. Lu, Supervised LLE in ICA space for facial expression recognition, in: *Proceedings of the International Conference on Neural Networks and Brain ICNNB'05*, vol. 3, 2005, pp. 1970–1975. doi:10.1109/ICNNB.2005.1615010.
- [27] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N.L. Roux, M. Ouimet, Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, 2004, pp. 177–184.
- [28] O. Kouropteva, O. Okun, M. Pietikäinen, Incremental locally linear embedding, *Pattern Recognition* 38 (10) (2005) 1764–1767.
- [29] M.H.C. Law, A.K. Jain, Incremental nonlinear dimensionality reduction by manifold learning, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (3) (2006) 377–391.
- [30] S. Schuon, M. Durkovic, K. Diepold, J. Scheuerle, S. Markward, Truly incremental locally linear embedding, in: *1st International Workshop on Cognition for Technical Systems*.
- [31] S. Xiang, F. Nie, Y. Song, C. Zhang, C. Zhang, Embedding new data points for manifold learning via coordinate propagation, *Knowledge and Information Systems* 19 (2) (2009) 159–184.
- [32] L. Shi, Q. Yang, E. Liu, J. Li, Y. Dong, An incremental manifold learning algorithm based on the small world model, *Proceedings of the 2010 International Conference on Life System Modeling and Intelligent Computing*, and *2010 International Conference on Intelligent Computing for Sustainable Energy and Environment: Part I, LSMS/ICSEE'10*, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 324–332.
- [33] J. Yan, B. Zhang, S. Yan, N. Liu, Q. Yang, Q. Cheng, H. Li, Z. Chen, W.-Y. Ma, A scalable supervised algorithm for dimensionality reduction on streaming data, *Information Sciences* 176 (2006) 2042–2065.
- [34] A. Schlar, L. Rokach, Random projection ensemble classifiers, in: J. Filipe, J. Cordeiro (Eds.), *Proceedings of the 11th International Conference on Enterprise Information Systems*, Springer-Verlag, 2009, pp. 309–316.
- [35] J. Maudes, J.J. Rodríguez, C. García-Osorio, C. Pardo, Random projections for linear svm ensembles, *Applied Intelligence* 34 (3) (2011) 347–359.
- [36] X.-Z. Wang, C.-R. Dong, Improving generalization of fuzzy if-then rules by maximizing fuzzy entropy, *IEEE Transactions on Fuzzy Systems* (2009) 556–567.
- [37] L.I. Kuncheva, Full-class set classification using the hungarian algorithm, *International Journal of Machine Learning and Cybernetics* 1 (2010) 53–61.
- [38] B. Biggio, G. Fumera, F. Roli, Multiple classifier systems for robust classifier design in adversarial environments, *International Journal of Machine Learning and Cybernetics* 1 (2010) 27–41.
- [39] A. Frank, A. Asuncion, UCI machine learning repository (2010). <http://www.archive.ics.uci.edu/ml>.
- [40] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, *Neural Computation* 10 (7) (1998) 1895–1923.
- [41] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* 7 (2006) 1–30.
- [42] G.I. Webb, MultiBoosting: a technique for combining boosting and wagging, *Machine Learning* 40 (2) (2000) 159–196.
- [43] J.R. Quinlan, Bagging, boosting, and C4.5, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI Press and the MIT Press, 1996, pp. 725–730.
- [44] D.D. Margineantu, T.G. Dietterich, Pruning adaptive boosting, in: D.H. Fisher (Ed.), *Proceedings of the Fourteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, pp. 211–218.
- [45] Y. Zhang, S. Bhattacharyya, Genetic programming in classifying large-scale data: an ensemble method, *Information Sciences* 163 (2004) 85–101.
- [46] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: A. Priditis, J.F. Lemmer (Eds.), *Machine Learning: Proceedings of the Twelfth International Conference*, Elsevier Science Publishers, 1995, pp. 275–283.
- [47] R. Kohavi, D.H. Wolpert, Bias plus variance decomposition for zero-one loss functions, in: L. Saitta (Ed.), *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kaufman, 1996, pp. 275–283.
- [48] L. Breiman, Stacked regressions, *Machine Learning* 24 (1) (1996) 49–64.
- [49] J.H. Friedman, On bias, variance, 0/1-loss, and the curse-of-dimensionality, *Data Mining and Knowledge Discovery* 1 (1997) 55–77.