Neurocomputing 71 (2008) 2521–2528

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Class-switching neural network ensembles

Gonzalo Martínez-Muñoz *, Aitor Sánchez-Martínez, Daniel Hernández-Lobato, Alberto Suárez

Computer Science Department, Escuela Politécnica Superior, Universidad Autónoma de Madrid, C/Francisco Tomás y Valiente, 11, Madrid E-28049, Spain

ARTICLE INFO

Available online 8 May 2008

Keywords:
Ensembles of classifiers
Neural networks
Class-switching
Bagging
Boosting
Decision trees

ABSTRACT

This article investigates the properties of class-switching ensembles composed of neural networks and compares them to class-switching ensembles of decision trees and to standard ensemble learning methods, such as bagging and boosting. In a class-switching ensemble, each learner is constructed using a modified version of the training data. This modification consists in switching the class labels of a fraction of training examples that are selected at random from the original training set. Experiments on 20 benchmark classification problems, including real-world and synthetic data, show that class-switching ensembles composed of neural networks can obtain significant improvements in the generalization accuracy over single neural networks and bagging and boosting ensembles. Furthermore, it is possible to build medium-sized ensembles (≈ 200 networks) whose classification performance is comparable to larger class-switching ensembles (≈ 1000 learners) of unpruned decision trees.

1. Introduction

Ensemble methods in machine learning attempt to induce a collection of diverse predictors which are both accurate and complementary, so that, when the decisions of the different learners are combined, better prediction accuracy on previously unseen data is obtained. The goal is to generate from a given training data set a collection of diverse predictors whose errors are uncorrelated. Ensembles built in this manner often exhibit significant performance improvements over a single predictor in many regression and classification problems. Ensembles can be built using different base classifiers: decision stumps [27] decision trees [11,25,2–4,27,22,9,26,6], neural networks [13,31,23,28,22,26,8], support vector machines [16,29,30], etc.

To create diversity, ensemble methods introduce perturbations at some stage in the generation of individual predictors. These modifications, which often involve injecting some amount of randomness, can be either in the algorithm that is used to build the base learners or in the training data used as input for the induction process. The motivation for introducing an aleatory element in the base learning algorithm is that different executions of the randomized training algorithm on the same instance of a learning problem should generate diverse classifiers. For example, in *randomization* [17], the base learners are decision trees generated with a modified tree construction algorithm. This algorithm computes the best 20 splits for every internal node and then chooses one of them at random. Another algorithm of this type consists in generating diverse neural networks using different random initializations of the synaptic weights. This simple technique is sufficient to generate fairly accurate ensembles [22].

Perturbations in the training data set can be introduced in different ways: using bootstrap samples from the training data, modifying the empirical distribution of the data (either by resampling or reweighting examples), manipulating the input features or altering the output targets, etc. Bagging [3], one of the most widespread methods for ensemble learning, belongs to this group of techniques. In bagging, each individual classifier is generated using a surrogate training set of the same size as the original one. This surrogate set is obtained by random resampling with replacement from the original training data. In boosting [11], the empirical distribution is modified by reweighting the training examples depending on the performance of the generated classifiers on those examples. Initially, the weights of the training examples are equal. At each iteration of the boosting process, the weights of the training data are updated according to whether the classification produced by the classifier generated in the previous iteration is correct or not. The weights of correctly classified examples are decreased, and the weights of incorrectly classified ones are increased. In this manner, the subsequent base learner tends to focus on examples that are harder to classify. A weighted voting scheme, where the contribution made by predictors obtained later in the ensemble generally decreases, is finally used to combine the decisions of the individual learners.

Another strategy consists in manipulating the input features. For instance, one can randomly eliminate features of the input data before constructing each individual classifier. In *random subspaces* [14], each base learner is generated using a different random subset of the input features. Another data randomization

* Corresponding author. Tel.: +34 91 497 3364; fax: +34 91 497 2235.
E-mail addresses: gonzalo.martinez@uam.es (G. Martínez-Muñoz), aitor.sanchez@uam.es (A. Sánchez-Martínez), daniel.hernandez@uam.es (D. Hernández-Lobato), alberto.suarez@uam.es (A. Suárez).
URL: http://www.eps.uam.es/~gonzalo (G. Martínez-Muñoz).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.11.041

strategy consists in modifying the class labels. In particular, in classification problems with multiple classes, one can build each classifier in the ensemble using a different coding of the class labels [10,12].

The ensemble method analyzed in this work belongs to this last group of techniques. It is based on randomly modifying the class label of a fraction of instances of the training set to generate each classifier. This type of ensemble learning algorithms were introduced in [5] (flipping) and further analyzed in [18,19] (class-switching). In [19] it is shown that class-switching ensembles composed of a sufficiently large number of unpruned decision trees trained on data where a fairly large fraction of the class-labels are switched exhibit good generalization performance in a large number of benchmark classification problems. Typically, these class-switching ensembles obtain accuracies equivalent or better than boosting and much better than bagging [19]. In [18], the performance of the class-switching algorithm using neural networks as the base learners is analyzed. Because of the different properties of neural networks and decision trees, several modifications of the procedure described in [19] need to be made to generate efficient class-switching ensembles composed of neural networks. In particular, it is found that, in contrast with class-switching ensembles of decision trees, one should not attempt to train the base learners to provide an exact fit of the perturbed training data. Instead, the number of hidden units for the individual networks should be determined using standard architecture selection techniques. Another difference with class-switching ensembles of decision trees is that, when neural networks are used as base learners, the best overall results are obtained with relatively low class-switching rates ($\hat{p} = \frac{1}{5}$ or $\frac{2}{5}$).

In this work, we extend the analysis of class-switching ensembles of neural networks presented in [18] and compare their performance with class-switching ensembles of decision trees, bagging and boosting in 20 benchmark classification tasks. When the base learners are neural networks, class-switching ensembles significantly outperform bagging and boosting ensembles in most of the classification tasks investigated. Comparing class-switching ensembles of neural networks and of decision trees, a smaller number of networks (around 200 networks) need to be aggregated in the ensembles to achieve the lowest possible (asymptotic) error level. This has the advantage that the memory requirements to store the ensemble are lower than with ensembles of decision trees, which typically need around 1000 trees to converge.

The article is organized as follows: Section 2 introduces the class-switching algorithm and the modifications that are necessary to build accurate ensembles composed of neural networks. Section 3 presents the results of experiments that compare the classification performance of a single neural network, bagging, boosting and class-switching ensembles in 20 data sets. Finally, the conclusions of this research are summarized in Section 4.

2. Class-switching ensembles

Switching the class labels to generate ensembles of classifiers was first proposed by Breiman [5]. A modification of the initial flipping algorithm, denominated class-switching, was proposed and analyzed in [19] using decision trees as base learners. Class-switching ensembles are built by generating each classifier in the ensemble using different perturbed versions of the original training set. To generate a perturbed version of the training set, a fixed fraction p of the examples of the original training set are selected at random and the class label of each of these selected examples is randomly switched to a different one. The class label randomization can be characterized by a transition probability matrix

$$P_{j\leftarrow i} = p/(K-1) \quad \text{for } i \neq j, \\ P_{i\leftarrow i} = 1-p, \tag{1}$$

where $P_{j\leftarrow i}$ is the probability that an example whose label is i becomes labeled as belonging to class j and K is the number of classes in the problem.

The class-flipping procedure proposed by Breiman [5] is designed to ensure that, on average, the class proportions of the original training set are maintained in the modified training sets. In [19], it was shown that, for several benchmark problems in which the training data exhibits an imbalanced distribution of classes, the ensembles generated with class-flipping do not perform well. By contrast, class-switching ensembles composed of decision trees, where the class labels are switched at random, without attempting to maintain the original distribution of classes, are competitive with bagging and boosting ensembles for a large range of balanced and unbalanced classification tasks [19]. In order for class-switching to work, the fraction of examples whose class label is changed, p, should be small enough to ensure that, for any given class and for every region in the attribute space, there is still a majority of correctly labeled examples (i.e. examples whose class labels have not been switched). This condition is fulfilled on the training set (on average) if $P_{j\leftarrow i} < P_{i\leftarrow i}$. Hence, using (1), the switching rate p should fulfill

$$p < (K-1)/K. \tag{2}$$

From this equation, the ratio of the class-switching probability to its maximum value is defined as

$$\hat{p} = p/p_{\max} = pK/(K-1). \tag{3}$$

Using values of p over this limit would generate, for some regions in feature space, a majority of examples incorrectly labeled and, in consequence, those regions would be incorrectly classified by the ensemble.

Ref. [19] describes the conditions under which class-switching can generate accurate ensembles composed of $C4.5$ decision trees. In that investigation it was found that large ensembles of *unpruned* decision trees trained on data with fairly large class-switching rates $\hat{p}$ (but sufficiently small so that the perturbed problem bears a statistical resemblance to the original problem) exhibit a good generalization performance over a large range of benchmark classification tasks. Empirically, a value of $\hat{p} \approx \frac{3}{5}$ produced excellent results in all the classification tasks investigated [19]. The use of unpruned decision trees instead of pruned trees is motivated by their better performance when combined in the ensemble. Note that, provided that there are no training examples with identical attributes values belonging to different classes, an unpruned decision tree achieves perfect classification (0-error rate) on the perturbed training set: Each individual tree exhibits a large amount of overfitting. A consequence of using models that perfectly fit the perturbed training sets is that it is necessary to combine a large number of trees in the ensemble (≈ 1000) to ensure that the injected noise is averaged out and that the patterns of the original classification task are amplified by the aggregation of the individual learners.

Preliminary experiments were performed to check whether the prescription used for decision trees (i.e. 0 training error of the trees on the perturbed sets, large ensembles and high values of $\hat{p}$) can be directly applied to neural networks [18]. Note that the architecture and the weights of the neural network have to be finely tuned for each problem in order to obtain neural models with ≈ 0-error rates in the modified training sets. This is a drawback with respect to decision trees, where constructing 0-error models for the training data is straightforward and problem independent: it is sufficient to grow a decision tree until perfect classification is achieved.

Fig. 1 displays the value of the generalization error (averaged over 10 executions) as a function of the number of base classifiers for class-switching ensembles composed of neural networks (solid lines) and for decision trees (trait lines) in the *Waveform* data set [7]. In these experiments, the architecture and the connection weights of the network are chosen to achieve 0-error in the perturbed versions of the training data. In particular, networks with a single layer of 28 hidden units, trained over 1000 epochs are used. The bottom curves correspond to a $\hat{p}$ value of $\frac{3}{5}$ and the top curves correspond to $\hat{p} = \frac{4}{5}$. The learning curves displayed in this figure show that the generalization errors of the class-switching neural ensembles generated in this way are similar to those produced by class-switching ensembles of decision trees. However, since the baseline performance given by a single decision tree is different from the performance of a neural net, the conclusions for ensembles composed of decision trees and for ensembles of neural networks are different. The improvement obtained by decision tree class-switching ensembles over a single tree is substantial (the generalization error of a single decision tree is $\approx 30\%$). Hence, for decision trees, the strategy of generating 0-error base learners seems to perform well. The simple decision boundaries produced by single trees, which are based on making partitions parallel to coordinate axes in attribute space, evolve to more complex and convoluted boundaries when the decisions of the individual trees are combined in the ensemble. In contrast, class-switching ensembles composed of complex neural networks trained to classify the training data without errors do not significantly improve the generalization performance of a single network. In particular, for the *Waveform* problem single neural networks built with standard training algorithms (average of 5.2 hidden units and 570 training epochs), achieve an average error rate of around 20.0%. Class-switching ensembles composed of 1000 networks, each of which is designed and trained to achieve 0-error rates on the perturbed training sets, obtain an improvement of only about 1% point (19.0%) when the class-switching rate is $\hat{p} = \frac{4}{5}$. The results are slightly better (generalization errors of approximately 17.7%) when the class-switching rates are lower ($\hat{p} = \frac{2}{5}$ or $\frac{3}{5}$). By contrast, class-switching ensembles with class-switching rates $\hat{p} = [\frac{0}{5}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}]$ composed of 200 neural nets trained in the same conditions as the single network obtain average generalization errors around $\approx 16.5\%$, which represents a significant improvement over configurations that use more complex nets and larger ensembles, at a lower computational cost. Note that a neural net with this smaller number of hidden units does not necessarily obtain a 0-error model on the modified training data.

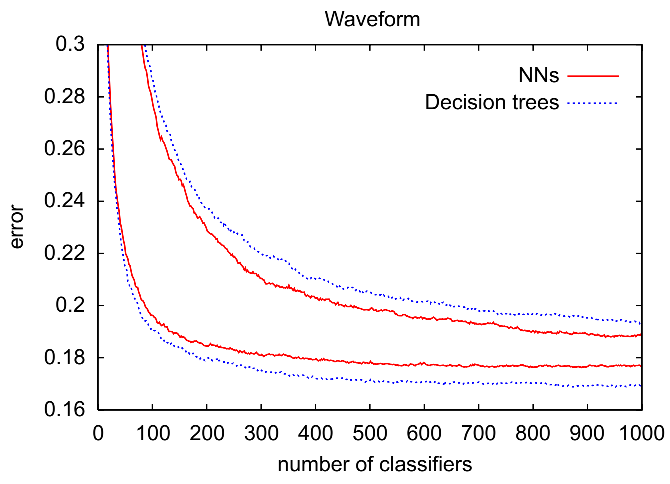


Fig. 1. Average test errors for class-switching ensembles composed of neural networks (solid lines) and decision trees (trait lines) using $\hat{p} = \frac{3}{5}$ (bottom curves) and $\hat{p} = \frac{4}{5}$ (top curves) for the *Waveform* data set.

In summary, a small ensemble of simple networks whose architecture is determined by standard procedures is trained much faster and exhibits better generalization performance than a large ensemble of complex networks trained to exhibit 0-error on the perturbed versions of the training set [18]. This is the prescription that will be used in the remainder of this work to build class-switching ensembles of neural networks.

3. Experiments

The performance of class-switching ensembles composed of feedforward neural networks with one hidden layer is investigated in a variety of classification tasks. Experiments are carried out in 20 classification problems from various fields of application and with different characteristics (number of classes, number of attributes, distribution of classes): 16 classification tasks from the UCI repository [1] and four synthetic data sets (*Led*24, *Ringnorm*, *Twonorm* and *Waveform* [7,4]). Table 1 displays the properties of the selected data sets, the protocol used for testing and the characteristics of the neural networks used as base learners in the ensemble.

Nominal features are binarized by assigning a different input unit for each label of each nominal attribute. In addition, all input features are normalized to have zero mean and unit standard deviation. The weights are randomly initialized between -0.1 and 0.1. Sigmoidal transfer functions for both the hidden and the output layers are used. The number of units in the output layer is equal to the number of classes in the classification task and the networks are trained to approximate the posterior probability of each class. The parameters of the network are determined using an improved RPROP batch algorithm [15]. The optimal architecture and number of training epochs for the neural networks are estimated for every partition of the data using cross-validation within the training data set. The same architecture and number of epochs is used in bagging, boosting and class-switching ensembles. For the neural networks, the FANN library [21] implementation is used.

The results given are averages over 100 experiments for each data set. In the real-world data sets these experiments consist in the execution of 10×10-fold cross-validation. For the synthetic data sets (*Led*24, *Ringnorm*, *Twonorm* and *Waveform*) different independent random samples for the training and testing sets are generated in each experiment. The sizes of these sets are given in Table 1. Each experiment involves the following steps:

(1) Obtain the training and testing sets from the corresponding cross-validation fold in the real-world data sets and by random sampling in the synthetic ones.
(2) Build a single neural network using the whole training data set. The configuration of the network is estimated using 10-fold cross-validation in the training data. Different architectures (3, 5, and 7 hidden units) and different values for the number of epochs (100, 300, 500 and 1000) are explored. The configuration that obtains on average the best accuracy on the separate folds of the training data is employed. For those data sets in which the cross-validation procedure selected most of the times the maximum number of epochs and hidden units, the range of possible hidden units considered is incremented. In the *Led24* and *Vehicle* data sets it was necessary to test 7, 11, 15 and 20 hidden units; for *E-coli*, *Ionosphere* and *New-thyroid* the architectures tested are 11, 15, 20 and 25; for *Glass* and

Table 1
Characteristics of the data sets, testing method, number of input units, average number (± standard deviation) of hidden units and average number of training epochs for the neural networks used in the experiments

Data set	Train	Test	Attrib.	Classes	Input units	Hidden units	Training epochs
Australian	690	10-fold-cv	14	3	42	4.76 ± 1.71	227
Breast W.	699	10-fold-cv	9	2	9	4.12 ± 1.49	328
Diabetes	768	10-fold-cv	8	2	8	5.36 ± 1.62	364
E-coli	332	10-fold-cv	7	8	7	21.11 ± 4.73	175
German	1000	10-fold-cv	20	2	61	4.98 ± 1.65	173
Glass	214	10-fold-cv	9	7	9	25.35 ± 5.09	448
Heart	270	10-fold-cv	13	2	23	4.84 ± 1.70	201
Ionosphere	351	10-fold-cv	34	2	34	20.26 ± 4.47	175
Labor	57	10-fold-cv	16	2	37	4.42 ± 1.54	405
Led24	200	5000 cases	24	10	48	13.8 ± 3.96	112
Liver	345	10-fold-cv	6	2	6	5.36 ± 1.64	282
New-thyroid	215	10-fold-cv	5	3	5	20.32 ± 4.17	522
Ringnorm	300	5000 cases	20	2	20	27.45 ± 3.98	570
Sonar	208	10-fold-cv	60	2	60	5.14 ± 1.46	331
Tic-tac-toe	958	10-fold-cv	9	2	27	4.38 ± 1.50	200
Twonorm	300	5000 cases	20	2	20	4.36 ± 1.61	330
Vehicle	846	10-fold-cv	18	4	18	11.7 ± 3.19	810
Votes	335	10-fold-cv	16	2	48	5.46 ± 1.50	220
Waveform	300	5000 cases	21	3	21	5.18 ± 1.48	570
Wine	178	10-fold-cv	13	3	13	5.76 ± 1.44	428

Ringnorm networks with 15, 20, 25 and 30 neurons in the hidden layer are considered. The values of the average number of hidden units and epochs used for training in the different data sets are displayed in Table 1.

(3) Generate a collection of neural networks to be included in the class-switching ensemble. Each network in the ensemble is trained on a different perturbed training set. These sets are obtained from the original training data using different class-switching rates ($\hat{p} = \frac{0}{5}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}$ and $\frac{4}{5}$). Note that class-switching with $\hat{p} = \frac{0}{5}$ cannot be considered a class-switching algorithm: the variability in the ensemble is achieved solely by the fact that the training process converges to different weight values because of their different random initializations.
(4) Build bagging and boosting ensembles composed of 200 neural networks. Boosting is implemented with resampling. The architecture of the networks and the number of learning epochs used for the neural networks in the class-switching, bagging and boosting ensembles is the same as the configuration selected using cross-validation for training a single neural network.
(5) As a reference, class-switching ensembles composed of 1000 C4.5 decision trees [24] with a switching rate of $\hat{p} = \frac{3}{5}$ are built. This configuration has the best overall results for class-switching ensembles composed of decision trees in a wide range of problems [19].
(6) The generalization errors of the different classifiers (a single neural network, bagging, boosting and class-switching) are estimated on the corresponding test sets.

Fig. 2 displays the evolution of the average generalization error for bagging, boosting and class-switching ensembles as a function of the number of neural networks aggregated in the ensemble for a representative subset of the classification problems investigated. As a reference, the average generalization error of a single net is displayed with a horizontal line in the plots. These graphs show that the convergence of the error of class-switching ensembles is related to the fraction of switched examples (i.e. $\hat{p}$): higher $\hat{p}$ values result in slower convergence rates. For most of the ensemble configurations, combining 200 networks seems to be sufficient for the error curves to level off and attain the asymptotic error rate of the ensemble. However, for some data sets (see for example the *German*, *Waveform* and *Wine* data sets in Fig. 2) using a high class-switching probability (class-switching with $\hat{p} = \frac{4}{5}$), 200 does not seem to be sufficient to reach the asymptotic ensemble error rate. In contrast, randomly initialized neural network ensembles ($\hat{p} = \frac{0}{5}$) reach their asymptotic error level after combining a fairly small number of networks (≈ 20). In most of the classification problems investigated the error rate of a single neural network is above the asymptotic error level of the different ensembles.

Table 2 presents the test errors averaged over the 100 executions for single networks, bagging, boosting, class-switching ensembles of neural networks for the different class-switching rates, $\hat{p}$, and for class-switching ensembles composed of decision trees using $\hat{p} = \frac{3}{5}$. For each data set, the generalization errors that are significantly better than bagging are highlighted in boldface. The results that are significantly better than boosting are underlined. The test used to determine whether differences are statistically significant is a paired t-test with an alpha-value 0.01. The cross-validation procedure used in the real-world problems tends to produce estimates of the p-values which are too large [20]. For this reason, a low alpha-value (1% instead of 5%) is used in the test. For the synthetic data sets the p-values are not biased because the experiments are carried out using independent sampling. The standard deviations of the values reported are given after the $\pm$ signs.

The best generalization error for neural based algorithms for each data set is marked with an asterisk. Considering only neural network ensembles, class-switching ensembles exhibit the best results in 14 of the 20 problems analyzed. Bagging has the best performance in five classification problems, boosting in three and ensembles with $\hat{p} = \frac{0}{5}$ in two. The performance of a single neural network is suboptimal in all the classification tasks analyzed and is generally poorer than most of the ensemble methods investigated. Table 2 also shows that most configurations of class-switching neural network ensembles reach similar generalization errors in many data sets. In particular, nearly the same results (within 0.2 points) are achieved in *Diabetes*, *E-coli*, *German*, *Tic-tac-toe*, *Twonorm*, *Votes*, *Waveform* and *Wine* by class-switching with $\hat{p} = \frac{1}{5}, \frac{2}{5}$ and $\frac{3}{5}$. The $\hat{p} = \frac{4}{5}$ configuration exhibits significantly worse results in several data sets (namely *German*, *Glass*,

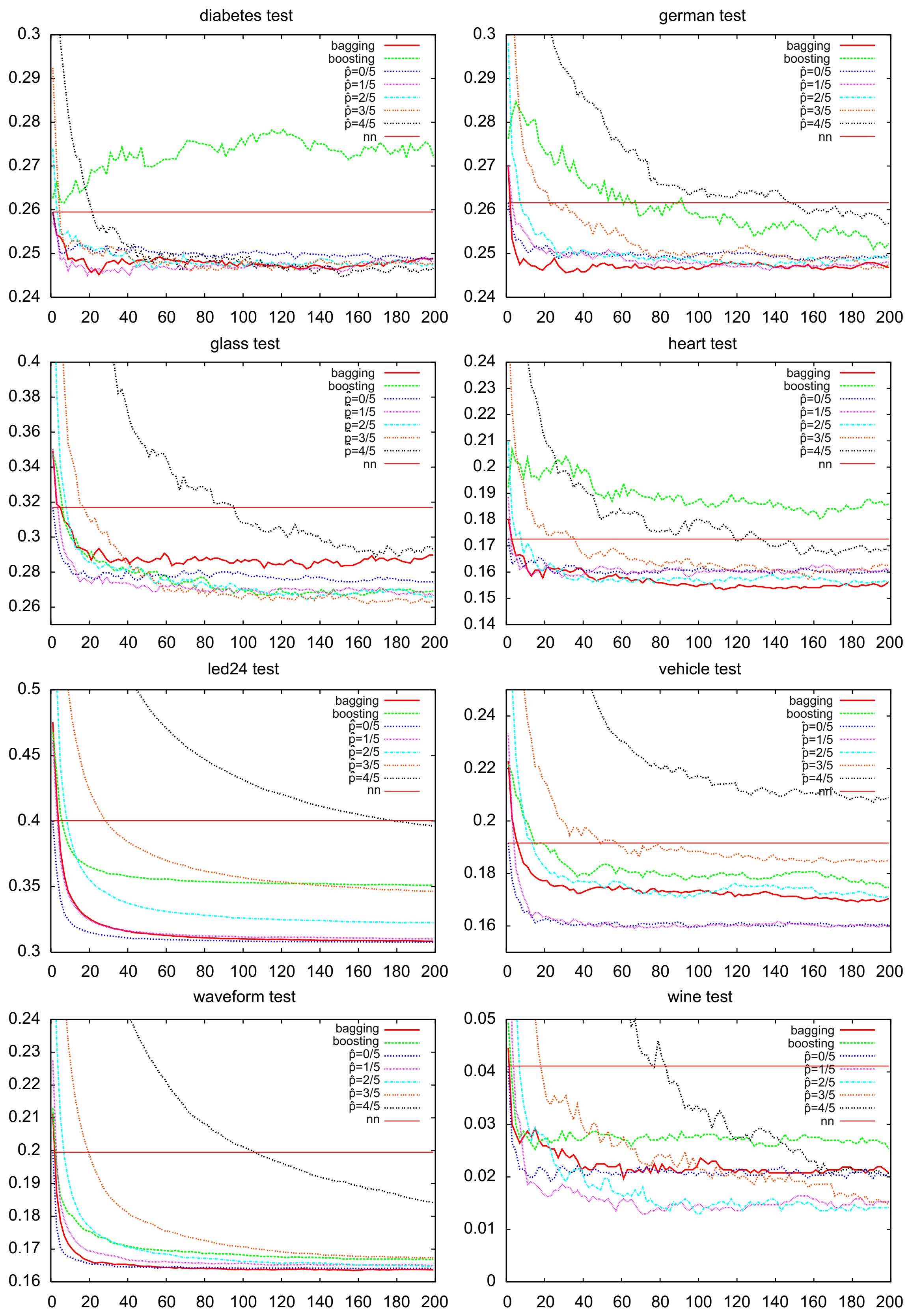


Fig. 2. Average test error as a function of the number of classifiers for bagging, boosting and class-switching ensembles for a representative subset of the classification problems investigated.

Table 2
Average generalization errors

Data sets	NN	Bag.	Boost.	Class-switching ($\hat{p}=$)					CS (trees)
				0/5	1/5	2/5	3/5	4/5	3/5
Australian	15.8 ± 4.0	$\underline{14.7} \pm \underline{3.7}$	15.5 ± 4.0	$\underline{14.7} \pm \underline{4.0}$	$\underline{14.7} \pm \underline{3.8}$	$\underline{14.5} \pm \underline{3.7}$	$\underline{\mathbf{14.0}} \pm \underline{\mathbf{3.7}}$	$\ast\underline{\mathbf{13.9}} \pm \underline{\mathbf{3.5}}$	$\underline{\mathbf{12.5}} \pm \underline{\mathbf{3.7}}$
Breast W.	3.9 ± 2.5	3.9 ± 2.4	4.3 ± 2.5	3.8 ± 2.4	$\underline{3.8} \pm \underline{2.4}$	$\underline{3.7} \pm \underline{2.2}$	$\ast\underline{\mathbf{3.3}} \pm \underline{\mathbf{2.3}}$	$\ast\underline{\mathbf{3.3}} \pm \underline{\mathbf{2.3}}$	$\underline{\mathbf{3.1}} \pm \underline{\mathbf{2.1}}$
Diabetes	$\underline{25.9} \pm \underline{4.7}$	$\underline{24.9} \pm \underline{4.6}$	27.6 ± 4.5	$\underline{24.8} \pm \underline{4.7}$	$\underline{24.9} \pm \underline{4.5}$	$\underline{24.8} \pm \underline{4.2}$	$\underline{24.8} \pm \underline{4.4}$	$\ast\underline{24.6} \pm \underline{4.7}$	$\underline{25.3} \pm \underline{3.9}$
E-coli	15.7 ± 5.9	14.6 ± 5.3	14.7 ± 5.3	14.3 ± 5.4	$\underline{\mathbf{13.7}} \pm \underline{\mathbf{5.3}}$	$\underline{\mathbf{13.8}} \pm \underline{\mathbf{5.3}}$	$\underline{\mathbf{13.8}} \pm \underline{\mathbf{5.2}}$	$\ast\underline{\mathbf{13.6}} \pm \underline{\mathbf{5.1}}$	13.7 ± 5.0
German	26.2 ± 5.4	$\ast 24.7 \pm 5.9$	25.2 ± 5.3	25.0 ± 6.2	24.8 ± 5.9	24.9 ± 5.9	24.7 ± 6.0	25.7 ± 6.4	24.5 ± 2.5
Glass	31.7 ± 9.6	28.9 ± 8.4	$\mathbf{27.0} \pm \mathbf{8.3}$	$\mathbf{27.5} \pm \mathbf{8.8}$	$\mathbf{26.7} \pm \mathbf{8.7}$	$\mathbf{26.6} \pm \mathbf{9.0}$	$\ast\mathbf{26.3} \pm \mathbf{8.3}$	29.4 ± 8.9	$\underline{\mathbf{21.0}} \pm \underline{\mathbf{9.2}}$
Heart	17.3 ± 7.0	$\ast\underline{15.6} \pm \underline{7.5}$	18.5 ± 7.1	$\underline{16.0} \pm \underline{7.2}$	$\underline{16.1} \pm \underline{7.3}$	$\ast\underline{15.6} \pm \underline{6.8}$	$\underline{16.3} \pm \underline{7.0}$	$\underline{17.0} \pm \underline{7.0}$	20.4 ± 7.2
Ionosphere	8.1 ± 5.1	7.9 ± 4.6	8.3 ± 4.7	$\underline{\mathbf{7.1}} \pm \underline{\mathbf{4.6}}$	$\ast\underline{\mathbf{7.0}} \pm \underline{\mathbf{4.2}}$	$\underline{7.5} \pm \underline{4.4}$	7.8 ± 4.0	9.3 ± 4.2	$\underline{\mathbf{6.0}} \pm \underline{\mathbf{3.7}}$
Labor	8.6 ± 11.9	8.4 ± 11.9	$\ast 6.3 \pm 11.2$	7.2 ± 10.7	6.6 ± 11.1	7.5 ± 12.0	10.8 ± 14.6	14.5 ± 16.5	11.0 ± 12.6
Led24	40.0 ± 3.4	$\ast\underline{30.8} \pm \underline{1.9}$	35.1 ± 2.1	$\ast\underline{30.8} \pm \underline{1.8}$	$\underline{31.0} \pm \underline{1.8}$	$\underline{32.2} \pm \underline{2.0}$	34.6 ± 2.0	39.6 ± 2.1	$\underline{34.2} \pm \underline{2.0}$
Liver	31.4 ± 8.3	29.8 ± 8.3	29.1 ± 6.9	30.0 ± 8.3	29.2 ± 7.7	$\ast\mathbf{28.4} \pm \mathbf{8.3}$	28.7 ± 7.8	28.7 ± 7.5	29.5 ± 7.2
New-thyroid	5.3 ± 4.7	5.1 ± 4.8	$\ast\mathbf{3.6} \pm \mathbf{3.5}$	4.7 ± 4.3	$\mathbf{4.0} \pm \mathbf{3.8}$	$\mathbf{3.9} \pm \mathbf{3.7}$	4.2 ± 3.9	4.8 ± 4.6	$\mathbf{2.9} \pm \mathbf{3.6}$
Ringnorm	15.7 ± 1.5	10.7 ± 2.0	$\mathbf{4.6} \pm \mathbf{0.8}$	10.6 ± 1.5	$\mathbf{4.4} \pm \mathbf{1.3}$	$\ast\underline{\mathbf{3.4}} \pm \underline{\mathbf{1.0}}$	$\underline{\mathbf{3.8}} \pm \underline{\mathbf{0.7}}$	$\mathbf{6.2} \pm \mathbf{0.9}$	$\mathbf{5.3} \pm \mathbf{0.6}$
Sonar	23.5 ± 8.8	20.2 ± 8.7	$\ast 18.7 \pm 8.4$	21.3 ± 8.4	21.0 ± 8.5	21.1 ± 9.2	21.6 ± 9.3	23.2 ± 9.6	21.6 ± 8.3
Tic-tac-toe	2.2 ± 1.8	$\underline{1.8} \pm \underline{1.3}$	2.0 ± 1.3	1.9 ± 1.4	$\underline{1.8} \pm \underline{1.3}$	$\underline{1.8} \pm \underline{1.2}$	$\ast\underline{1.7} \pm \underline{1.2}$	7.7 ± 5.5	2.2 ± 1.7
Twonorm	3.8 ± 0.7	$\underline{3.1} \pm \underline{0.4}$	3.4 ± 0.5	3.5 ± 0.6	$\underline{3.1} \pm \underline{0.4}$	$\ast\underline{\mathbf{2.9}} \pm \underline{\mathbf{0.4}}$	$\ast\underline{\mathbf{2.9}} \pm \underline{\mathbf{0.5}}$	3.3 ± 1.1	3.9 ± 0.3
Vehicle	19.2 ± 3.6	17.0 ± 3.9	17.5 ± 4.0	$\underline{\mathbf{16.1}} \pm \underline{\mathbf{3.9}}$	$\ast\underline{\mathbf{15.9}} \pm \underline{\mathbf{3.6}}$	17.1 ± 3.5	18.4 ± 3.5	20.9 ± 3.9	23.2 ± 3.6
Votes	5.2 ± 3.3	$\ast\underline{4.2} \pm \underline{3.3}$	4.9 ± 3.0	4.5 ± 3.0	4.8 ± 3.3	5.0 ± 3.4	5.0 ± 3.4	5.5 ± 3.5	4.2 ± 2.7
Waveform	20.0 ± 4.1	$\ast\underline{16.4} \pm \underline{1.0}$	16.7 ± 0.9	$\ast\underline{16.4} \pm \underline{1.0}$	16.5 ± 1.0	$\underline{16.5} \pm \underline{0.9}$	16.7 ± 1.0	18.4 ± 1.2	16.9 ± 0.8
Wine	4.1 ± 4.1	2.1 ± 3.7	2.6 ± 3.8	2.0 ± 3.4	$\underline{1.5} \pm \underline{2.8}$	$\ast\underline{1.4} \pm \underline{2.6}$	$\underline{1.6} \pm \underline{3.0}$	2.1 ± 3.5	1.4 ± 2.8
Average	16.2 ± 5.0	14.2 ± 4.7	14.3 ± 4.4	14.1 ± 4.6	13.6 ± 4.4	13.6 ± 4.5	14.1 ± 4.6	15.6 ± 5.1	14.1 ± 4.2

Table 3
Win/draw/loss records for ensembles composed of pruned trees

	NN	Bag.	Boost.	Class-switching ($\hat{p}=$)					CS-DT	Total
				0/5	1/5	2/5	3/5	4/5		
NN	X	0/5/15	1/8/11	0/2/18	0/3/17	0/4/16	0/6/14	4/7/9	2/8/10	7/43/110
Bag.	15/5/0	X	8/9/3	2/15/3	1/13/6	2/12/6	4/10/6	10/6/4	6/8/6	48/78/34
Boost.	11/8/1	3/9/8	X	4/9/7	1/8/11	1/7/12	3/8/9	9/6/5	6/8/6	38/63/59
0/5	**18/2/0**	3/15/2	7/9/4	X	1/14/5	2/11/7	5/9/6	8/8/4	5/10/5	49/78/33
1/5	17/3/0	**6/13/1**	11/8/1	5/14/1	X	2/15/3	5/11/4	11/7/2	**8/9/3**	65/80/15
2/5	16/4/0	6/12/2	**12/7/1**	**7/11/2**	**3/15/2**	X	**5/14/1**	**14/5/1**	7/10/3	**70/78/12**
3/5	14/6/0	6/10/4	9/8/3	6/9/5	4/11/5	1/14/5	X	11/9/0	5/11/4	56/78/26
4/5	9/7/4	4/6/10	5/6/9	4/8/8	2/7/11	1/5/14	0/9/11	X	3/8/9	28/56/76
CS-DT	10/8/2	6/8/6	6/8/6	5/10/5	3/9/8	3/10/7	4/11/5	9/8/3	X	46/72/42

The significance of the differences in performance is measured using a paired t-test with an alpha-value of 0.01.

Ionosphere, *Labor*, *Ringnorm*, *Vehicle*). In some cases this is due to the fact that larger ensembles ought to have been used.

The relative performance of neural network and decision tree ensembles is problem dependent. The differences between the generalization errors of the two types of class-switching ensembles are significant in many classification tasks. In *Australian*, *Glass*, *Ionosphere* and *New-thyroid*, class-switching ensembles of decision trees outperform all the configurations of neural network ensembles. By contrast, in *Heart* and *Vehicle*, and to a lesser extent in *Labor* and *Twonorm*, the decision tree ensembles are inferior to ensembles of neural networks.

Table 3 shows *win/draw/loss* records. The numbers displayed in each cell correspond to the number of data sets in which the algorithm displayed in the leftmost column wins/draws/losses when compared to the algorithm displayed in the top row. A difference is considered to be significant when a paired t-test has a p-value smaller than 0.01. For each column, the record with the largest value of *wins–losses* is highlighted in boldface. These results show that, for most of the data sets investigated, the ensembles considered have significantly better classification accuracies than single neural networks. The best performance with respect to a single neural network is the randomly initialized neural network ensemble: 18 significant wins and no significant losses. Single neural networks significantly outperform boosting in one data set, class-switching with $\hat{p}=\frac{4}{5}$ four times and class-switching ensembles of decision trees in two problems.

Neural network ensembles generated with class-switching rates of $\hat{p}=\frac{1}{5}$ and $\frac{2}{5}$ obtain the best overall results. These configurations rarely perform significantly worse than bagging and they perform better than bagging six times. Class-switching with $\hat{p}=\frac{3}{5}$ and $\frac{4}{5}$ improve the results of bagging in several data sets but also perform significantly worse than bagging in other data sets. The performance of randomly initialized neural network ensembles ($\hat{p}=\frac{0}{5}$) is very similar to the performance of bagging: both methods draw in 15 out of the 20 data sets. This observation confirms the results obtained by Optiz and Maclin [22]. The comparison of class-switching ensembles of neural networks with respect to boosting is more favorable. Class-switching with $\hat{p}=\frac{1}{5}$ and $\frac{2}{5}$ is significantly worse than boosting only in one data set and they outperform boosting 11 and 12 times, respectively. In addition, it can be observed that boosting combined with neural nets is not as effective as boosting of decision trees: for the investigated data sets bagging significantly outperforms boosting eight times and looses three times. This observation confirms previously observed results [22] and indicates that using standard boosting algorithms to combine strong learners is less effective than boosting weak learners. For class-switching ensembles composed of decision trees the results are problem dependent: Decision tree ensembles perform better than bagging and boosting in six problems and worse also in six problems. The comparison with class-switching ensembles of neural networks is less favorable for decision tree ensembles (eight significant losses and three wins with respect to the $\hat{p} = \frac{1}{5}$ neural network ensemble). According to the experiments performed, the best overall method (see last column of Table 3) is class-switching with $\hat{p} = \frac{2}{5}$: It wins in 70 out of 160 comparisons, draws 78 times and loses only on 12 occasions.

The relative classification speeds of class-switching ensembles composed of decision trees or of neural networks also vary for different problems: even though ensembles of neural networks are smaller, this advantage can be offset by the complexity of the classification process in the networks, which can be larger than in decision trees. The learning process in individual networks can also be slow. This implies that, regarding training speed, there is no clear overall advantage for any of the two types of class-switching ensembles.

4. Conclusions

Class-switching ensembles generate a diversity of classifiers using different perturbed versions of the training set [5,19]. To generate each perturbed set, a fraction of examples is randomly selected and their class labels are switched also at random to a different value. The prescription used for decision trees (generate individual classifiers that achieve 0-error in the perturbed training data sets) is found not to be the appropriate configuration for neural network ensembles constructed with class-switching [18]. Combining neural networks whose architecture is designed by standard architecture selection techniques (and that therefore do not necessarily achieve 0-error in the perturbed training data sets) produces significantly better results than ensembles composed of more complex nets that do achieve 0-error in the perturbed data sets. Since the networks in the ensemble are not constructed to have 0-error on the perturbed training sets, they seem to avoid overfitting to the noise injected, at least to a certain extent, and perform well in the original unperturbed problem. As a consequence, the number of base learners needed for the convergence of the ensemble errors to their asymptotic values is smaller than in class-switching ensembles composed of unpruned decision trees.

The classification accuracy of class-switching neural network ensembles is better or equivalent to bagging and better than boosting and single nets in the problems investigated. Ensembles generated with a class-switching rate of $\hat{p} = \frac{1}{5}$ or $\frac{2}{5}$ obtain the best overall results. These configurations (i.e. $p = \frac{1}{5}, \frac{2}{5}$) rarely obtain generalization errors significantly worse than bagging or boosting. This is not the case for other values of $\hat{p}$ (that is, $\hat{p} = \frac{0}{5}, \frac{3}{5}$ and $\frac{4}{5}$), where results both better and worse than boosting and bagging have been observed.

The question of whether to use neural networks or decision trees as base learners in a class-switching ensemble has an answer that is problem dependent both in terms of accuracy and of classification speed. Nevertheless, both types of ensembles have demonstrated an excellent overall performance in the classification tasks investigated. This establishes that class-switching can be a useful randomization mechanism for the creation of ensembles of classifiers with very different characteristics, such as decision trees and neural networks.

Acknowledgments

The authors acknowledge financial support from the Spanish *Dirección General de Investigación*, project TIN2004-07676-C02-02. Daniel Hernández-Lobato acknowledges support from the Consejería de Educación de la Comunidad Autónoma de Madrid and from the European Social Fund under an F.P.I. grant.

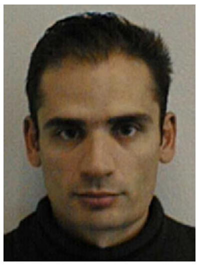
References

[1] A. Asuncion, D.J. Newman, UCI machine learning repository, 2007 ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[2] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1–2) (1999) 105–139.
[3] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[4] L. Breiman, Arcing classifiers, Ann. Statist. 26 (3) (1998) 801–849.
[5] L. Breiman, Randomizing outputs to increase prediction accuracy, Mach. Learn. 40 (3) (2000) 229–242.
[6] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[7] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Chapman & Hall, New York, 1984.
[8] I. Cantador, J.R. Dorronsoro, Balanced boosting with parallel perceptrons, in: J. Cabestany, A. Prieto, F.S. Hernández (Eds.), IWANN, Lecture Notes in Computer Science, Vol. 3572, Springer, 2005.
[9] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, Mach. Learn. 40 (2) (2000) 139–157.
[10] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. 2 (1995) 263–286.
[11] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in: Proceedings of the 2nd European Conference on Computational Learning Theory, 1995.
[12] J. Fürnkranz, Round robin classification, J. Mach. Learn. Res. 2 (2002) 721–747.
[13] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[14] T.K. Ho, C4.5 decision forests, in: Proceedings of the 14th International Conference on Pattern Recognition, vol. 1, 1998.
[15] C. Igel, M. Hüsken, Improving the RPROP learning algorithm, in: Proceedings of the 2nd International Symposium on Neural Computation, ICSC Academic Press, 2000.
[16] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, S.Y. Bang, Constructing support vector machine ensemble, Pattern Recognition 36 (12) (2003) 2757–2767.
[17] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: Proceedings of the 12th International Conference on Machine Learning, 1995.
[18] G. Martínez-Mu noz, A. Sánchez-Martínez, A. Suárez, Building ensembles of neural networks with class-switching, in: S.D. Kollias, A. Stafylopatis, W. Duch, E. Oja (Eds.), ICANN (1), Lecture Notes in Computer Science, Vol. 4131, Springer, 2006.
[19] G. Martínez-Muñoz, A. Suárez, Switching class labels to generate classification ensembles, Pattern Recognition 38 (10) (2005) 1483–1494.
[20] C. Nadeau, Y. Bengio, Inference for the generalization error, Mach. Learn. 52 (3) (2003) 239–281.
[21] S. Nissen, Implementation of a fast artificial neural network library (FANN), Technical Report, Department of Computer Science, University of Copenhagen, 2003. URL: ⟨http://fann.sourceforge.net/report/report.html⟩.
[22] D. Opitz, R. Maclin, Popular ensemble methods: an empirical study, J. Artif. Intell. Res. 11 (1999) 169–198.
[23] M.P. Perrone, L.N. Cooper, When networks disagree: ensemble methods for hybrid neural networks, in: R.J. Mammone (Ed.), Neural Networks for Speech and Image Processing, Chapman & Hall, London, 1993, pp. 126–142.
[24] J.R. Quinlan, C4.5 Programs for Machine Learning, Morgan Kaufmann, Los Altos, CA, 1993.
[25] J.R. Quinlan, Bagging boosting and C4.5, in: Proceedings of the 13th National Conference on Artificial Intelligence, Cambridge, MA, 1996.
[26] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (3) (2001) 287–320.
[27] R.E. Schapire, Y. Freund, P.L. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Statist. 12 (5) (1998) 1651–1686.
[28] A.J.C. Sharkey, Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, Springer, London, 1999.
[29] G. Valentini, T.G. Dietterich, Low bias bagged support vector machines, in: T. Fawcett, N. Mishra (Eds.), ICML, AAAI press, 2003.
[30] G. Valentini, T.G. Dietterich, Bias–variance analysis of support vector machines for the development of SVM-based ensemble methods, J. Mach. Learn. Res. 5 (2004) 725–775.
[31] D.H. Wolpert, Stacked generalization, Neural Networks 5 (2) (1992) 241–259.

Gonzalo Martínez-Muñoz received the degree of Licenciado in Physics and the M.Sc. and Ph.D. degrees in Computer Science from the Universidad Autónoma de Madrid (UAM), Madrid, Spain in 1995, 2001 and 2006, respectively. Currently, he is Assistant Professor in the Computer Science Department of the Universidad Autónoma de Madrid, Spain. From 1996 to 2002, he was with Geosys SL, a Spanish company specialized in Geographic Information Systems and remote sensing, as a Software Engineer developing Research and Development projects. His research interests include machine learning, pattern recognition, neural networks, decision trees, ensemble learning and genetic algorithms.

Aitor Sánchez-Martínez received the University Degree in Computer Science and Engineering from the Universidad Autónoma de Madrid (UAM) in 2006. He worked with a seven month internship (September 2005–March 2006) given by the Fundación General de la Universidad Autónoma de Madrid (FGUAM) in Risklab-Madrid company. He obtained the "Aprovechamiento académico excelente 2005/2006" fellowship by Comunidad Autónoma de Madrid to participate in a research project of pattern recognition and neural networks. From March 2006, he is working for Risklab-Madrid/QRR, a company specialized in Financial Risk consulting and software development, as a software designer and developer.

Daniel Hernández-Lobato received the Degree of Engineer in Computer Science from the Universidad Autónoma de Madrid (UAM) in 2004. He is currently a graduate student in the doctoral program of the Computer Science Department in the same university. He is the recipient of an FPI grant from the Consejería de Educación de la Comunidad de Madrid. His research interests include pattern recognition, machine learning and Bayesian inference.



Alberto Suárez received the degree of Licenciado in Chemistry from the Universidad Autónoma de Madrid (Spain) in 1988, and the Ph.D. degree in Physical Chemistry from the Massachusetts Institute of Technology (Cambridge, USA) in 1993. After holding postdoctoral positions at Stanford University (USA), the Université Libre de Bruxelles (Belgium), and the Katholieke Universiteit Leuven (Belgium), he is currently a professor in the Computer Science Department of the Universidad Autónoma de Madrid (Spain). He has worked on relaxation theory in condensed media, stochastic and thermodynamic theories of non-equilibrium systems, lattice-gas automata, and decision tree induction. His current research interests include machine learning, computational finance and information processing in the presence of noise.