Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition 38 (2005) 1483–1494
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Switching class labels to generate classification ensembles

Gonzalo Martínez-Muñoz*, Alberto Suárez

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

Received 4 August 2004; accepted 27 February 2005

Abstract

Ensembles that combine the decisions of classifiers generated by using perturbed versions of the training set where the classes of the training examples are randomly switched can produce a significant error reduction, provided that large numbers of units and high class switching rates are used. The classifiers generated by this procedure have statistically uncorrelated errors in the training set. Hence, the ensembles they form exhibit a similar dependence of the training error on ensemble size, independently of the classification problem. In particular, for binary classification problems, the classification performance of the ensemble on the training data can be analysed in terms of a Bernoulli process. Experiments on several UCI datasets demonstrate the improvements in classification accuracy that can be obtained using these class-switching ensembles.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Classification; Ensemble methods; Bagging; Boosting; Decision tree

1. Introduction

Classification methods based on pooling the decisions of an ensemble of classifiers have demonstrated great potential for improvement in many regression and classification problems [1–15]. To produce a reduction of the error rate, the classifiers generated must perform well on the proposed task and yet be sufficiently diverse. In order to achieve this diversity, ensemble algorithms introduce some systematic variation into the learning task, either by perturbing the training data or by taking advantage of instabilities in the learning algorithm.

One of the common procedures to generate classifier ensembles is bagging [3] (Bootstrap sampling and aggregation). In bagging, diversity is obtained by constructing each classifier in the ensemble with a different set of labelled examples, which is obtained from the original training set by re-sampling with replacement. Bagging then combines the decisions of the classifiers using unweighted voting. Bagging is believed to improve the performance of single classifiers mainly by reducing the variance error [4]. Breiman categorises bagging decision trees as a particular instance of *random forest* classification techniques [12]. A random forest is a tree-based ensemble that uses some kind of independent randomisation in the construction of every individual classifier. Many variants of bagging and random forests with excellent classification performance have been developed: In Ref. [9] trees in the ensemble are grown by randomly selecting among the best partitions at every tree node. In double-bagging [13] two classifiers are grown in each iteration by making use of the *out-of-bag* examples [16]. Attribute-bagging [14] selects a random subset of attributes at every iteration. IPG-ensembles [15] use different random partitions of the training data as input for the iterative growing and pruning tree construction algorithm of Gelfand et al. [17].

Another common algorithm for generating ensembles is boosting [1]. In boosting, the committee members are sequentially generated using weighted training data. Initially

* Corresponding author. Tel.: +34 91 497 3364; fax: +34 91 497 2235.
E-mail addresses: gonzalo.martinez@uam.es (G. Martínez-Muñoz), alberto.suarez@uam.es (A. Suárez).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2005.02.020

all example weights are equal to 1. At each iteration of the boosting process these weights are updated according to the classification given by the last committee member generated: weights of incorrectly classified examples are increased and weights of correctly classified ones are decreased. In this way the base learner focuses on the harder examples. This entails a reduction both in bias and variance [10]. A weighted vote is used to make the final class assignment. Boosting has demonstrated to be one of the most effective methods for constructing ensembles [2,7,9].

In this article we present a variant of the output flipping ensembles proposed by Breiman in Ref. [18], that belongs to the category of random forests. In Breiman's work, each classifier in the ensemble is generated using the original training set with randomised class labels: The class label of each example is switched according to a probability that depends on an overall switching rate (defined as the proportion of training examples that are switched on average) and on the proportions of the different class labels in the original training set. The switching probabilities are chosen to maintain the class distribution of the original training set. Error rates similar or better than bagging are reported by using ensembles with 100 classifiers.

In this article we show that still lower error rates can be achieved with ensembles generated by class switching provided that we use fairly large ensembles ($\approx$ 1000 classifiers) and relatively high class switching rates. In contrast to Ref. [18], we do not require that the original class distribution be maintained in the perturbed training data. This makes it possible to use larger values of the switching rate in unbalanced datasets. Larger values of the switching rate are sometimes needed to get better classification accuracy.

The paper is organised as follows: Section 2 introduces the algorithm for generating the ensemble by switching the class labels of the training examples. Section 3 describes a simple experiment that is used to analyse in detail the classification strategy of the proposed ensemble. The classification performance of the class switching ensemble algorithm is compared to that of Breiman's flipping ensemble algorithm, bagging and boosting in 15 datasets. Some of these problems are synthetic and some are real-world (taken from the UCI repository [19]). Finally, the conclusions of this research are summarised.

2. Switching outputs

In Ref. [18], Breiman proposes to generate diverse classifiers by randomly switching the class labels of the training dataset according to the transition matrix

$$\begin{aligned} P_{j\leftarrow i} &= wP_j \quad \text{for } i \neq j, \\ P_{i\leftarrow i} &= 1 - w(1 - P_i), \end{aligned} \tag{1}$$

where $P_{j\leftarrow i}$ is the probability that an example with label i gets the label j, P_i is the proportion of elements of class i in the training set, and w is proportional to the switching rate (average fraction of switched examples), p

$$w = \frac{p}{1 - \sum_j P_j^2} = \frac{p}{2\sum_j \sum_{k>j} P_j P_k}. \tag{2}$$

This form of the transition matrix, Eq. (1), is chosen to maintain the class proportions approximately constant.

In order for this method to work, the value of the switching rate p should be small enough to ensure that the training error tends to zero as the size of the ensemble grows. In a binary classification problem, the condition is

$$p < P_{min}, \tag{3}$$

where P_{min} is the proportion of examples that belong to the minority class. Inequality (3) ensures that, on average, the fraction of switched examples in the minority class is smaller than $\frac{1}{2}$. Switching rate values over this limit would flip the class label of more than half of the minority class examples. Hence, the minority feature space regions would be flooded with examples labelled as the majority class and consequently these regions would be classified incorrectly by the ensemble.

In this work we propose to generate ensembles of classifiers using different perturbed versions of the training set. In each perturbed set, a fixed fraction p of examples of the original training data is selected at random. The class label of each of these examples is randomly switched to a different one. This defines the following transition probability matrix:

$$\begin{aligned} P_{j\leftarrow i} &= p/(K-1) \quad \text{for } i \neq j, \\ P_{i\leftarrow i} &= 1 - p, \end{aligned} \tag{4}$$

where K is the number of classes. This label switching procedure produces training sets whose class distribution is usually different from that of the original training data. In fact, the class distribution of the perturbed set tends to equalise with increasing p for the unbalanced sets.

In order to guarantee the convergence of the ensemble in the training set there should be, for any given class, a majority of correctly labelled examples (i.e. not switched). This condition is fulfilled on the training set (on average) if $P_{j\leftarrow i} < P_{i\leftarrow i}$ and according to Eq. (4) we have

$$p < (K-1)/K \tag{5}$$

independently of the initial class distribution. Following this equation we define the maximum value of p:

$$pmax = (K-1)/K. \tag{6}$$

It is also convenient to define the ratio of the class switching probability to its maximum value

$$\hat{p} = p/pmax. \tag{7}$$

Thus, for unbalanced datasets, the proposed method increases the range of allowed values of p, with respect to the

class flipping method proposed by Breiman [18]. This is a key factor in improving the generalisation capacity of the ensemble, as will be shown in Section 4.

In order for the algorithm to work efficiently we need to use a base learner that manages to obtain a training error as low as possible. Note that a classifier that achieves perfect classification (0 error rate) on the perturbed training set exhibits an error rate in the original training set equal to the proportion of switched examples (p). An unpruned decision tree grown until all data have been separated fulfils this requirement. In fact, such decision tree always obtains a 0-error tree provided that there are no training examples with identical attributes values belonging to different classes in the perturbed set.

An interesting characteristic of the class-switching procedure is that the random selection of examples creates classifiers that have statistically independent errors and equal erring probability (note that in all cases the same number of examples are switched and that the base classifiers have nearly 0 error on the perturbed sets) on the original training set. Hence, we can obtain the estimation of the training error independently of the learning task at hand. In a binary classification problem the ensemble performance can be analysed in terms of a Bernoulli process, where each classifier has a $(1-p)$ probability of correctly classifying a given training example. The decision of a given classifier on a training example is, by construction, independent of that from the other classifiers. Thus, the probability of having a number of ensembles giving a correct classification is given by the binomial distribution. The ensemble training error can be estimated as the probability of having more than half of the classifiers misclassifying a given example

$$train_error(M) = \sum_{m=\lfloor 1+M/2 \rfloor}^{M} \binom{M}{m} p^m (1-p)^{M-m}, \quad (8)$$

where M is the number of classifiers in the ensemble (which is assumed to be odd to avoid ties). Based on the binomial distribution we can also estimate the margin curves for a class-switching ensemble on the training data for a two-class problem as

$$train_margin(x) = \sum_{m=0}^{\lfloor M(x+1)/2 \rfloor} \binom{M}{m} p^{M-m} (1-p)^m, \quad (9)$$

where x is the classification margin, defined as the fraction of correct classifiers minus the fraction of incorrect classifiers for the two-class problem [5]. For a given example, the margin is equal to -1 when all committee members agree on an incorrect classification and equal to 1 when all members agree on the correct classification.

The curves corresponding to Eqs. (8) and (9) are depicted in top and bottom plots of Fig. 1 , respectively. In Fig. 1 (top plot) the training error estimation is depicted for different values of p and for odd numbers of classifiers. Note that all the error curves tend to 0 (since we are considering a binary classification problem and the selected values of p are under 0.5). The plots show that for higher values of p more classifiers are needed for the ensemble training error to converge. Fig. 1 (bottom plot) shows the ensemble margin curves for the training set, with $p = 0.4$ and ensembles composed of 11, 101 and 1001 classifiers, respectively. Observe that all curves are centred at $x = 1 - 2p$ and that, with increasing M all examples tend to have margin equal to $1 - 2p$.

Eqs. (8) and (9) and plots in Fig. 1 are valid only for the training set. Nonetheless, we expect to observe some characteristics of these curves in the corresponding curves for the test set. In particular, the behaviour of the generalisation error rate depends on the value of p: Since the error on the test set is usually higher than on the train set, the effective threshold for p should be lower than the value given by Eq. (6). Below this value of p the error rate of the ensemble decreases with increasing ensemble size. The size of the ensemble needed to achieve convergence becomes larger as p approaches the threshold from below.

3. A simple classification experiment

In order to gain insight into the workings of the class-switching ensemble in comparison to other common ensemble procedures (bagging, boosting) we analyse in detail a simple learning task: Consider a linearly separable binary classification problem on a two-dimensional feature space, where the two classes are separated by the line $y = x$. This diagonal boundary is difficult to describe by a tree algorithm such as C4.5, that generates divisions only parallel to the feature space axis. This limitation implies that C4.5 usually finds poor separation boundaries for this kind of problems. The training set consists of 300 random feature vectors uniformly distributed at random in the unit square ($x \sim U[0, 1]$ and $y \sim U[0, 1]$). Using this data, bagging, boosting and class-switching ($p = 0.4$ and 0.2) ensembles of up to 1001 C4.5 trees are generated. The values of the parameters used in the tree generation are the same as those described in the experiments presented in Section 4.

The performance of the different ensembles is evaluated on a test set consisting of a 300×300 regular grid of points in the unit square. Fig. 2 depicts the classification results for different stages of the process. The first row shows the results using a single tree and the last row shows the ensembles using 1001 trees. Odd number of trees are used to avoid draws in the voting process. This figure shows that bagging and boosting converge faster than class-switching ensembles to their asymptotic classification behaviour. The class-switching algorithm starts rather poorly. In fact, a single classifier in this ensemble exhibits a classification pattern that does not resemble the original one. For value $p = 0.4$ of the class switching rate even when 101 classifiers are used the feature space is not properly separated. A large number of classifiers ($\geqslant 1000$) is needed to correctly define the boundary. This is coherent with Eq. (8), and

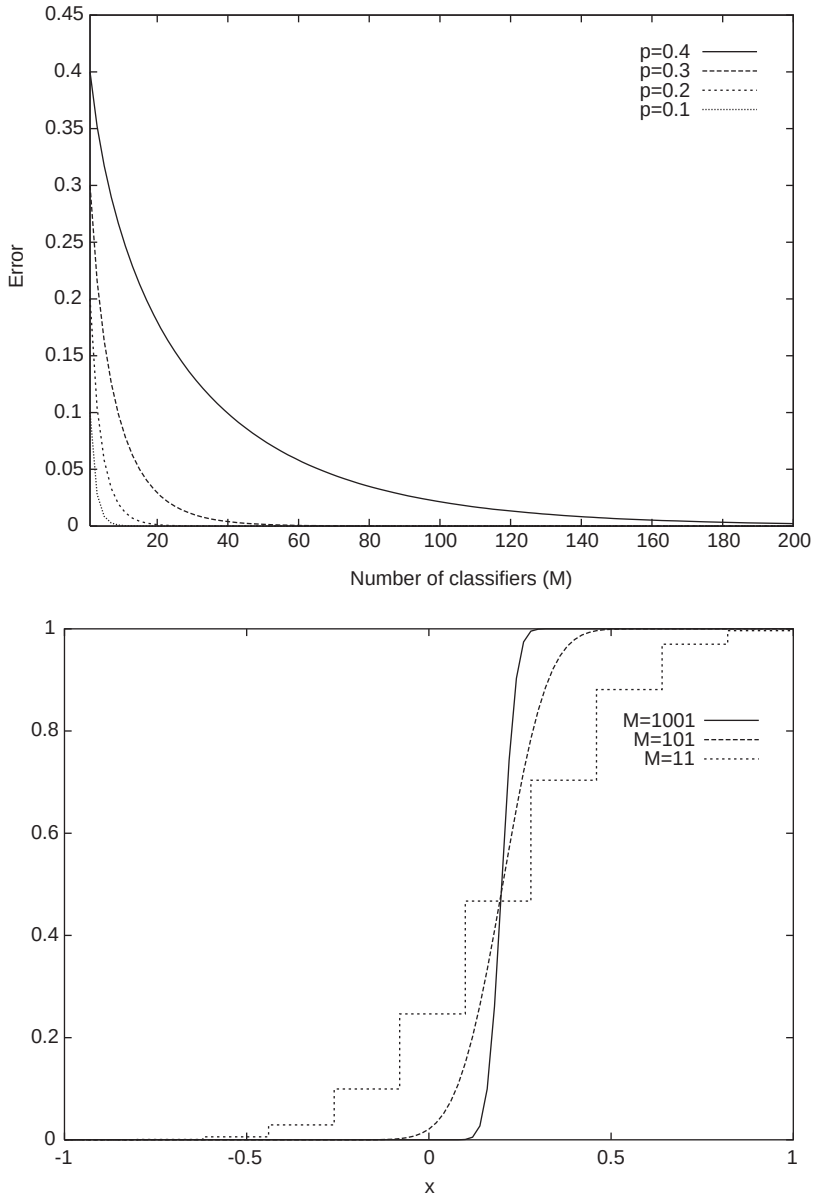


Fig. 1. (*Top plot*) Training error estimation in a binary classification problem versus ensemble size for ensembles with class switching rate $p = 0.1$ (*dotted line*), $p = 0.2$ (*short trait line*), $p = 0.3$ (*long trait line*) and $p = 0.4$ (*solid line*). (*Bottom plot*) Margin curves estimations in a binary classification problem for ensembles with class switching rate $p = 0.4$ for ensemble sizes 11(*short trait line*), 101(*long trait line*) and 1001(*solid line*).

further reinforces the conjecture that a large number of classifiers is needed for convergence. In spite of the slow convergence, the final classification accuracy is better than bagging or boosting. Even more interesting than the final accuracy is the profile of its final boundary. Bagging and boosting produce class boundaries that bear a strong resemblance to the boundaries produced by C4.5. Class switching ensembles produce a more convoluted decision boundary, whose shape is quite different from those generated by C4.5. This suggests that the class-switching algorithm can lead to a significant bias reduction of the base learner.

The origin of the differences in complexity of the boundaries can be traced to the fact that bagging and boosting, and class-switching ensembles pose different classification

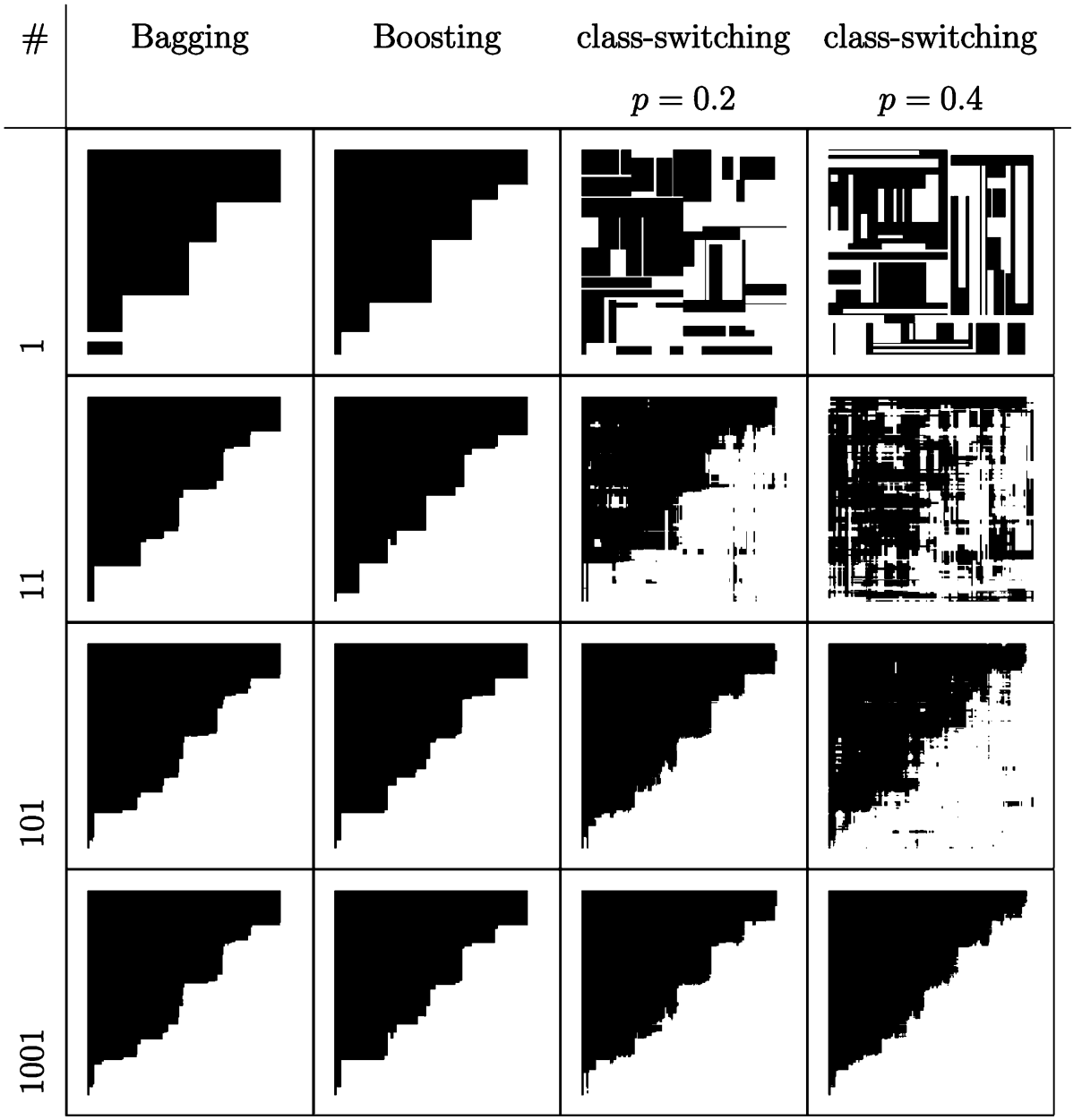


Fig. 2. Classification map for a perfectly separable linear problem for bagging, boosting and class-switching ensembles ($p = 0.2$ and 0.4). The number of trees used in each of the ensembles is marked on the left side of each line (1, 11, 101, 1001 trees, from top to bottom).

problems to the base learners. For each ensemble unit, bagging and boosting generate a learning problem that has a clear relation with the original problem. In fact, each of the classifiers in a bagging ensemble is a reasonably good solution of the problem. Boosting is different in this respect from bagging but still produces problems that are closely related to the original one. In fact, as the size boosting ensemble grows, the importance of the misclassified examples is increased. Thus, the base learner tends to focus on solving the harder parts of the problem. In contrast, the class-switching algorithm generates surrogate learning problems that can be quite different from the original one (especially for high values of p), whose resemblance to the original problem is only statistical: the decision boundary becomes defined only asymptotically, as the number of classifiers becomes large.

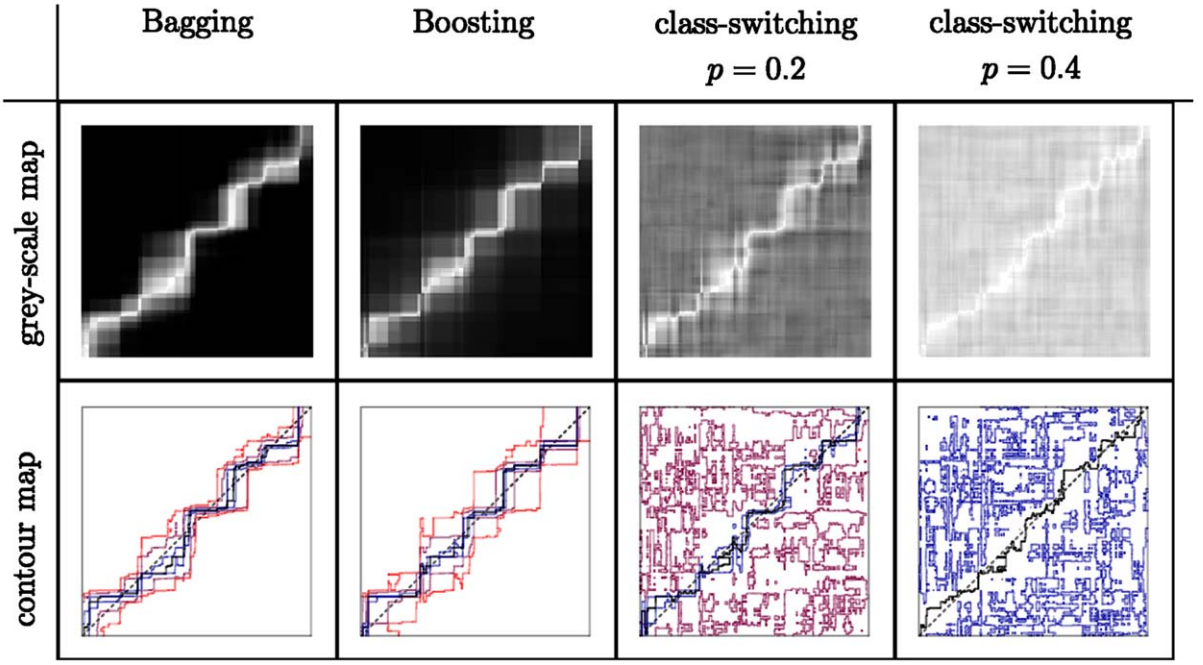


Fig. 3. Margin map for a perfectly separable linear problem for bagging, boosting and class-switching ($p = 0.2$ and 0.4) ensembles using 1001 classifiers (see details in the text).

Fig. 3 shows the margin map of the final decision ($M = 1001$) for the different ensembles (margin in the sense of votes difference (weighted or unweighted) between the most voted class and the second most voted class instead of using the margin based on the difference between the true class minus the incorrect class as defined in Ref. [5]). The first row represents the value of the margin using a raster of inverted grey-scale values map, where lighter grey values indicate lower margin values. The second row plots the margin values as an isoline map, i.e. each line represents locations in the feature space with equal margin values. The true problem boundary (the diagonal $y = x$) is also depicted, with a long trait line. Isolines for margins 0 (the ensemble decision boundary, marked by darker lines in the plots), and 0.2, 0.6 and 0.8 are shown. In the bagging and boosting ensembles, the isolines for values 0.2, 0.6 and 0.8 appear in pairs (one line for each class) at locations that are further away from the decision boundary as the margin value increases. In the class-switching ensemble with $p = 0.2$ the pairs of 0.2 and 0.6 margin isolines appear. The pair of lines for margin value 0.2 appear very near to the decision boundary, whereas for margin value 0.6 these lines have a rather convoluted structure and fill up the feature space. In the class-switching ensemble with $p = 0.4$ only the pair of 0.2 margin isolines appear. For this ensemble there are no points where the margin is 0.6 or higher. The differences between margin maps in the various ensembles are apparent in this figure: bagging generates large connected areas where all classifiers agree and a "narrow" uncertainty border. Boosting delineates a more precise frontier at the expense of lower margins on zones adjacent to the decision boundary (the uncertainty border region is broader). Both bagging and boosting exhibit progressively higher margin values when moving to regions that are further away from the boundary. The class-switching procedure generates a more delocalised margin map. Asymptotically, as the number of classifiers in the ensemble increases, the margin map consists in extended flat regions with constant margin $\approx 1 - 2p$ separated by narrow boundaries of lower margin values.

4. Experiments on UCI datasets

In order to evaluate the improvements in classification accuracy that can be obtained with class-switching ensembles, we compare the performance of this method with: C4.5, Breiman's class flipping ensembles [18], boosting and bagging. All ensemble methods are implemented using C4.5 Release 8 as the base classifier. For C4.5, bagging and boosting we use the default options of the C4.5 package. For class-switching and flipping ensembles we took away the major improvement of Release 8 in relation to Release 7; that is, a minimum description length (MDL)-penalty term that is applied to continuous attributes. Such a penalty generally produces better and smaller trees [20]. However, it stops the growth of C4.5 trees before all pure node leafs are

Table 1
Characteristics of the datasets used in the experiments

Dataset	Train	Test	Attributes	Classes	Class distribution
Australian	500	190	14	2	383/307
Breast W.	500	199	9	2	458/241
Diabetes	468	300	8	2	500/268
German	600	400	20	2	700/300
Heart	170	100	13	2	150/120
Horse-Colic	244	124	21	2	232/136
Ionosphere	234	117	34	2	225/126
New-thyroid	140	75	5	3	150/35/30
Segment	210	2100	19	7	*eq. distrib.*
Threenorm	300	5000	20	2	*eq. distrib.*
Tic-tac-toe	600	358	9	2	626/332
Twonorm	300	5000	20	2	*eq. distrib.*
Vowel	600	390	10	11	*eq. distrib.*
Waveform	300	5000	21	3	*eq. distrib.*
Wine	100	78	13	3	71/59/48

obtained. Hence, it does not fulfill the requirement for class-switching ensembles, i.e. nearly zero training error. We also set the minimum number of elements for a leaf to 1 and fully developed trees (i.e. no pruning) are used. This configuration is also similar to Breiman's implementation of class flipping ensembles [18] that used unpruned *CART* trees [21].

The boosting variant implemented is the one described in Ref. [10], based on Ref. [7]. In this algorithm reweighting (instead of resampling) is used, as suggested in Ref. [2], allowing all training examples to be included for the generation of every ensemble unit. A minimum weight limit for the training examples of 10^{-8} is set to avoid numeric underflow problems. The boosting process is continued even when a base learner obtains an error over 0.5 or equal to 0. In such cases the training set is substituted by a bootstrap sample from the original training set with all weights set to 1. This strategy prevents *Adaboost* from stopping too early. The last classifier is discarded if its error is over 0.5 and it is kept in the ensemble with a weight equal to $\ln(10^{10})$—equivalent to assigning a very small error ($\approx 10^{-10}$) to the classifier—if its error is equal to 0. In 5 of the studied datasets this last modification produced some differences that always increased the average accuracy of boosting.

We have tested the implemented algorithms in 15 different machine learning problems. Three of the sets are synthetic datasets (*Threenorm*, *Twonorm* and *Waveform*) proposed by Breiman [22,21]. The remaining problems are included in the UCI repository [19]: *Australian credit*, *Breast Cancer Wisconsin*, *Pima Indian Diabetes*, *German Credit*, *Heart*, *Horse Colic*, *Ionosphere*, *New-Thyroid*, *Image Segmentation*, *Tic-tac-toe*, *Vowel* and *Wine*. The datasets have been selected in order to sample a variety of problems: real and synthetic data; sets with different numbers of classes and attributes. Table 1 displays, for each dataset, the number of examples used for training and testing, the number of attributes, the number of classes and the number of examples for each class. The proportions of the training set is about $\frac{2}{3}$ of the total number of examples except for the synthetic sets and for the *Image Segmentation* set, in which the training set size described in its documentation was used.

For each dataset 100 experiments were carried out. Each experiment involved the following steps:

(1) Generate a stratified random partition of the data in training and testing for the real sets and a random sampling for the synthetic datasets (see Table 1 for sizes).
(2) Construct a C4.5 tree, and ensembles of 1000 trees using class-switching and flipping (with $\hat{p}$ values of: $\frac{1}{5}$, $\frac{2}{5}$, $\frac{3}{5}$ and $\frac{4}{5}$) and boosting and bagging.
(3) Compute the error of the classifiers on the testing set to obtain an estimate of the generalisation error.

Table 2 presents a summary of the average test errors obtained by C4.5 and the different ensembles using 1000 trees. The lowest average test error for every dataset is set in boldface and the second best is underlined. Standard deviation are only shown for C4.5. Except for some values displayed (marked with italics in the table), the standard deviations of the results are smaller than those reported for C4.5. The class-switching ensemble produces nine best results in 8 sets ($2\times\hat{p}=\frac{4}{5}$, $5\times\hat{p}=\frac{3}{5}$ and two for $\hat{p}=\frac{2}{5}$). Flipping produces the best results in 4 sets (for $2\times\hat{p}=\frac{3}{5}$ and $2\times\hat{p}=\frac{2}{5}$). Boosting produces the best results in the synthetic sets *Threenorm* and *Twonorm* and in *Tic-tac-toe*. Bagging is better in the difficult sets *Diabetes* and *Heart*.

A summary of the overall performance of the studied algorithms is shown in Table 3. This is displayed in form of *win/draw/loss* records, where the first (second/third) numbers displayed in each cell correspond to the number of sets in which the algorithm displayed in the first column on the

Table 2
Average test error (in %) using 1000 classifiers for C4.5, class-switching, flipping, boosting and bagging. The average best result for each problem is highlighted in **bold** type. The second best results are underlined. Means with standard deviation greater than that reported for C4.5 are shown in *italics*

	C4.5	Class-switching				Flipping				Boosting	Bagging
		$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$	$\hat{p}=\frac{2}{5}$	$\hat{p}=\frac{1}{5}$	$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$	$\hat{p}=\frac{2}{5}$	$\hat{p}=\frac{1}{5}$		
Australian	14.3±2.2	*14.8*	**13.0**	**13.0**	13.5	*20.8*	13.6	**13.0**	13.6	13.4	13.3
Breast W.	5.4±1.4	**3.0**	3.1	3.1	3.6	*34.4*	*7.1*	3.8	3.8	3.2	3.9
Diabetes	27.0±2.6	25.7	25.6	25.4	25.8	34.7	29.2	26.2	25.7	26.1	**24.6**
German	28.9±2.2	26.7	**25.0**	25.1	26.8	30.0	*29.9*	26.7	26.3	25.5	25.7
Heart	23.6±3.5	22.4	21.2	21.7	*22.8*	*29.0*	22.1	21.8	*23.1*	19.5	**19.1**
Horse-colic	15.9±2.9	15.8	16.1	16.0	15.8	36.7	18.4	**15.3**	15.6	17.1	16.0
Ionosphere	10.9±2.8	8.1	6.9	**6.2**	6.3	35.9	*18.7*	7.0	6.3	6.4	7.5
New-thyroid	8.4±3.1	**3.9**	4.0	4.2	5.1	30.2	30.3	*10.8*	4.5	5.7	*6.1*
Segment	10.3±1.4	7.6	**5.5**	5.7	7.0	7.5	**5.5**	5.7	7.1	6.5	8.1
Threenorm	31.7±1.2	18.7	17.7	18.2	*19.9*	18.7	17.7	18.2	*20.0*	**15.7**	19.1
Tic-tac-toe	17.3±2.3	6.7	3.4	3.9	6.3	34.8	19.1	6.5	6.2	**1.2**	8.9
Twonorm	21.6±0.7	4.6	3.8	4.0	*5.5*	4.6	3.8	4.0	*5.6*	**3.7**	*6.6*
Vowel	26.5±2.4	4.9	4.7	6.1	8.4	5.0	**4.6**	6.0	8.4	7.5	13.2
Waveform	29.0±1.3	19.2	**16.9**	17.3	19.3	22.5	17.5	17.6	19.4	17.4	19.4
Wine	9.2±4.0	2.6	**1.2**	1.8	3.1	*7.7*	1.5	1.5	3.0	4.1	6.4

Table 3
Win/draw/loss records summary. For each column, the record with higher *wins–losses* is highlighted in bold face (if positive)

	C4.5	Class-switching				Flipping				Boosting	Bagging
		$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$	$\hat{p}=\frac{2}{5}$	$\hat{p}=\frac{1}{5}$	$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$	$\hat{p}=\frac{2}{5}$	$\hat{p}=\frac{1}{5}$		
C4.5	X	1/0/14	1/0/14	1/0/14	0/0/15	9/0/6	7/0/8	1/0/14	0/0/15	1/0/14	1/0/14
Class-switching											
$\hat{p}=4/5$	14/0/1	X	3/0/12	4/0/11	10/1/4	12/2/1	7/0/8	4/1/10	8/1/6	6/0/9	10/0/5
$\hat{p}=3/5$	14/0/1	**12/0/3**	X	**10/2/3**	13/0/2	**15/0/0**	**11/3/1**	**13/1/1**	13/0/2	10/0/5	12/0/3
$\hat{p}=2/5$	14/0/1	11/0/4	3/2/10	X	**14/0/1**	14/0/1	10/0/5	8/4/3	**14/0/1**	**11/0/4**	**12/1/2**
$\hat{p}=1/5$	**15/0/0**	4/1/10	2/0/13	1/0/14	X	12/0/3	8/0/7	5/0/10	7/2/6	5/0/10	10/0/5
Flipping											
$\hat{p}=4/5$	6/0/9	1/2/12	0/0/15	1/0/14	3/0/12	X	1/0/14	1/0/14	3/0/12	1/0/14	4/0/11
$\hat{p}=3/5$	8/0/7	8/0/7	1/3/11	5/0/10	7/0/8	14/0/1	X	5/1/9	7/1/7	3/0/12	6/0/9
$\hat{p}=2/5$	14/0/1	10/1/4	1/1/13	3/4/8	10/0/5	14/0/1	9/1/5	X	9/1/5	5/0/10	11/0/4
$\hat{p}=1/5$	**15/0/0**	6/1/8	2/0/13	1/0/14	6/2/7	12/0/3	7/1/7	5/1/9	X	5/0/10	9/1/5
Boosting	14/0/1	9/0/6	5/0/10	4/0/11	10/0/5	14/0/1	12/0/3	10/0/5	10/0/5	X	11/0/4
Bagging	14/0/1	5/0/10	3/0/12	2/1/12	5/0/10	11/0/4	9/0/6	4/0/11	5/1/9	4/0/11	X

left of the table wins (draws/losses) with respect to the algorithm displayed in the topmost row. For each column, the record with a higher *wins–losses* value is highlighted in bold face, if positive. In this table we see that the only algorithm that outperforms all the rest is class-switching together with $\hat{p}=\frac{3}{5}$. Moreover, class-switching with $\hat{p}=\frac{3}{5}$ and $\frac{2}{5}$ are the only two algorithm configurations that outperform boosting.

Fig. 4 shows the dependence of the average train error (top plot) and test error (bottom plot) on the ensemble size in class-switching ensembles for different values of $\hat{p}$ in the *Breast Cancer Wisconsin* dataset. The training error curves shown on the top plot are similar to those on Fig. 1, and confirm the analysis of the train error based on Eq. (8). Note, however, that the coincidence is not exact: the train error curves for the *Breast Cancer Wisconsin* dataset do not start for one classifier in the value of p. This is due to the fact that the *Breast Cancer Wisconsin* set has several examples with equal feature values, which cannot be separated if the class-switching algorithm changes the class label of some (and not all) of these examples.

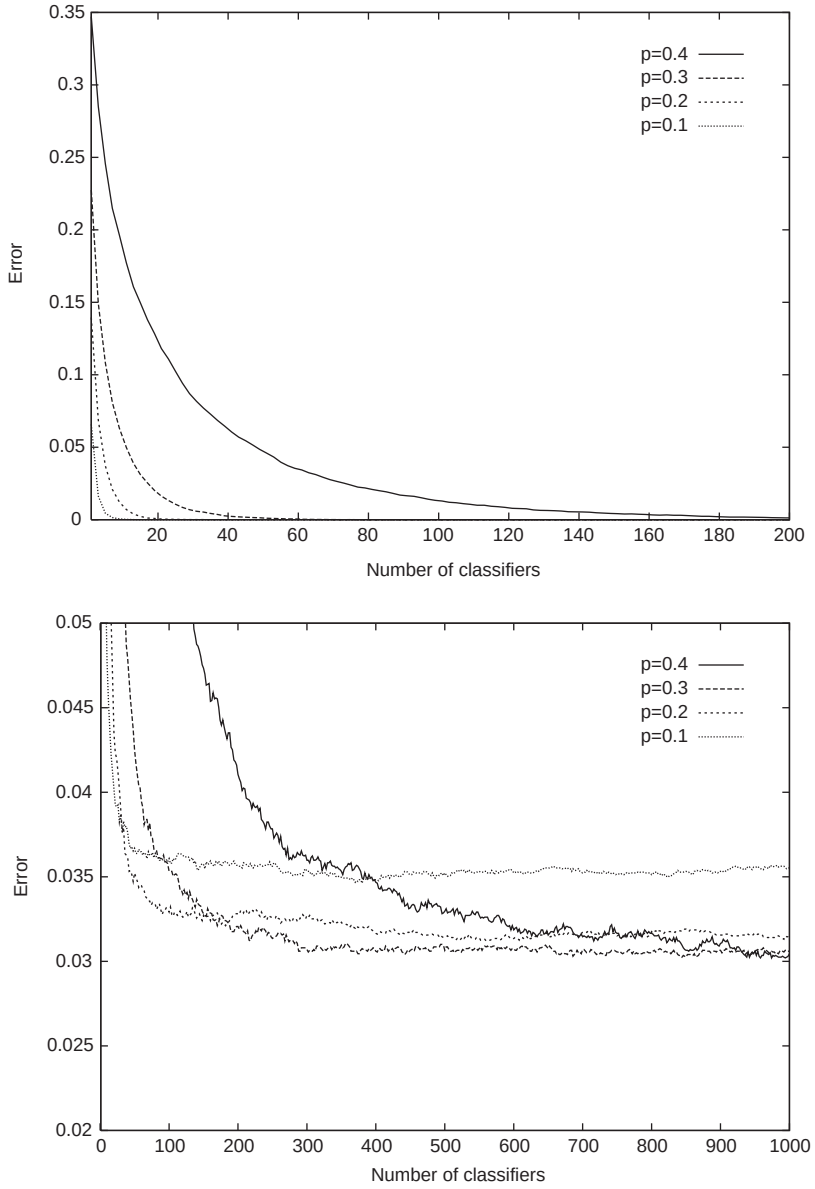


Fig. 4. Average train (*top plot*) and test (*bottom plot*) errors for the *Breast Cancer Wisconsin* dataset.

Table 2 and Fig. 4 confirm that for both the train error and the test error the convergence of the error class-switching ensembles is related to $\hat{p}$, the ratio of the actual switching probability and the maximum switching probability, defined in Eq. (7): Higher $\hat{p}$ values result in slower convergence. In the *Breast Cancer Wisconsin* set, for example, the class-switching ensemble with $\hat{p} = \frac{4}{5}$ obtained the best accuracy, but it needed 200, 400 and 800 classifiers to arrive to error rates equivalent to bagging, class-switching ensemble with $\hat{p} = \frac{1}{5}$ and boosting, respectively. Slight improvements can be obtained if more classifiers are generated (see the bottom plot in Fig. 4). In some other sets (*Threenorm*, *Tic-tac-toe* and *Twonorm*) the class-switching ensemble with $\hat{p} = \frac{4}{5}$ can produce better results if larger ensembles are generated (In *Twonorm* an error of 3.8 is obtained when using 2000 trees and in *Tic-tac-toe* an error of 4.9 is obtained for 5000 trees).

It is important to note that also the classification accuracy is related to $\hat{p}$. From Table 2 and Fig. 4 we see that higher values of $\hat{p}$ tend to produce higher accuracies. However, when using values of $\hat{p}$ near to 1 ($\hat{p} = \frac{4}{5}$), the class-switching test errors are generally worse than $\hat{p} = \frac{3}{5}$. This

Table 4
Average number of base classifier (in %) with test error higher than p_{max}

Dataset	Class-switching	
	$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$
Australian	9.2	0.1
Breast W.	0.3	0.0
Diabetes	12.1	1.0
German	12.9	1.1
Heart	21.4	6.2
Horse-Colic	9.4	0.2
Ionosphere	12.1	0.5
New-thyroid	4.2	0.0
Segment	0.0	0.0
Threenorm	3.5	0.0
Tic-tac-toe	5.2	0.0
Twonorm	0.2	0.0
Vowel	0.0	0.0
Waveform	0.1	0.0
Wine	5.3	0.1

can be explained by looking at generalisation errors of the generated individual classifiers. Table 4 presents the average number of classifiers that have generalisation errors higher than p_{max} (i.e. those that impair the overall ensemble performance) for class-switching ensembles with $\hat{p}=\frac{3}{5}$ and $\frac{4}{5}$. In the *German Credit* dataset, class-switching ensembles with $\hat{p}=\frac{4}{5}$ have an average of 12.9% of the classifiers with individual test error rate over 0.5 (even though the ensemble has an average error rate of 26.7), whereas with $\hat{p}=\frac{3}{5}$ this value is reduced to 1.1% (producing a generalisation error of 25.0).

The flipping and class-switching algorithms produce very similar generalisation errors in the datasets with equal distributed classes. However, and as expected from Eq. (3), using flipping together with p values that are above the proportion of the minority class produces too many examples labelled with the majority class in the feature space region where the minority class examples are located. This misleads the base learning algorithm, which labels these regions incorrectly. In Table 2 this effect can be observed for $P_{min}\approx p$ and $P_{min}\leqslant p$. In those cases the ensemble labels all the feature space as belonging to the majority class, producing test errors equal to the proportion of the minority class.

Since the main error differences between our algorithm and Breiman's occur mainly when the datasets are unbalanced, we have carried out a detailed comparison between both methods for the synthetic set *Threenorm* (Table 5). We select the *Threenorm* set because the results obtained by both algorithms are very similar when a balanced dataset (approximately equal number of examples from both classes) is used. Furthermore, using a synthetic set allows us to modify the a priori class probabilities when generating the training and testing sets. Both algorithms are tested using P_{min} (i.e. the fraction of examples belonging to the minority class) values of: 0.4, 0.3, 0.2 and 0.1. For each value of P_{min} we generated 10 training sets of 300 examples and one test set composed of 5000 with the same class proportions. The average errors for both algorithms in the test sets for different values of $\hat{p}$ and P_{min} are shown in Table 5 together, for reference, with the results for the balance sets ($P_{min}=0.5$) from Table 2. The results show that with decreasing P_{min} the range of available values of p for the flipping algorithm is reduced and its generalisation capability is also reduced. Flipping and class-switching obtain similar results for the balanced set. Nevertheless, it is significantly worse than class-switching (1.9 error percentage points worse) when slightly unbalanced sets are used ($P_{min}=0.4$), considering the best accuracy of each algorithm across the different values of $\hat{p}$. This difference increases for $P_{min}=0.3$ and $P_{min}=0.2$ to 2.0 and 2.4, respectively. For $P_{min}=0.1$ the difference decreases to 0.4 error percentage points. However, this last experiment configuration is quite difficult since only 30 examples of the minority class are used.

Table 5
Average test errors for *Threenorm* using unbalance sets for class-switching/flipping algorithms

	$\hat{p}=\frac{4}{5}$	$\hat{p}=\frac{3}{5}$	$\hat{p}=\frac{2}{5}$	$\hat{p}=\frac{1}{5}$
$P_{min}=0.5$	18.7/18.7	**17.7**/**17.7**	18.2/18.2	19.9/20.0
$P_{min}=0.4$	18.9/37.8	**17.9**/23.8	**17.9**/19.8	19.5/19.6
$P_{min}=0.3$	18.0/30.0	**17.1**/29.6	17.3/22.7	18.0/19.1
$P_{min}=0.2$	15.1/20.0	14.6/20.0	**14.4**/19.9	14.9/17.0
$P_{min}=0.1$	9.7/10.0	**9.6**/10.0	**9.6**/10.0	**9.6**/10.0

5. Conclusions

In this article we show that randomly switching the class labels can be used to construct ensembles with error accuracies better than bagging and comparable or better than boosting on several UCI datasets. These improvements of classification are only reached for relatively high class switching rates and large ensembles.

Randomising outputs as a method to generate ensembles of classifiers was first proposed by Breiman [18]. In this reference, the classification experiments are carried out with ensembles of 100 classifiers, which are too small to exhibit the method's full potential. In this work, we have shown that fairly large ensembles (with up to 1000 units) are needed to attain the asymptotic ensemble error rate, especially for high class switching values (p). Contrary to Breiman's prescription, the probability of switching class label is kept constant (i.e. independent of the original label and class distribution) for every training example. This has the advantage that higher switching rates can be used for unbalanced datasets and, as the experiments show, better accuracies can be produced. In balanced datasets, Breiman's method and

the class switching ensembles presented here have comparable classification accuracies.

Another important issue studied in this article is the relation between the switching rate parameter p with the accuracy and convergence of the ensemble. Higher p values introduce more noise in the individual classification problems that the base algorithm tackles. This means that, for higher p's, every learner represents a classification pattern that bears less resemblance to the original problem, and that, as a consequence, larger ensembles are needed to capture the regularities in the original unperturbed problem. Notwithstanding, far from being a drawback, high p values produce more complex boundaries that can produce better generalisation accuracy. There still exists an upper limit for p, which is reached when the generated individual classifiers exhibit generalisation errors close to random guessing. The experiments carried out show that class-switching ensembles with relative switching rates of $\frac{3}{5}$ achieve the best average performance on the tested datasets.

The simple procedure used for the generation of perturbed training sets allows a statistical analysis of the ensemble training in terms of a Bernoulli process. Provided that the base classifiers are sufficiently flexible to obtain a perfect classification on the perturbed training sets, the learning curves displaying the dependence of the training error on the size of the class-switching ensemble can be described with a sum of binomial terms. Furthermore, these curves are independent of the learning problem, except in sets that contain several examples characterised by the same feature vector.

Acknowledgements

This work has been supported by the Spanish "Dirección General de Investigación", project TIC2001-0572-C02-02.

References

[1] 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, pp. 23–37.

[2] J. Quinlan, Bagging, boosting, and C4.5, in: Proceedings of the 13th National Conference on Artificial Intelligence, Cambridge, MA, 1996, pp. 725–730.

[3] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.

[4] L. Breiman, Arcing classifiers, Ann. Stat. 26 (3) (1998) 801–849.

[5] R. Schapire, Y. Freund, P. Bartlett, W. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Stat. 12 (5) (1998) 1651–1686.

[6] M. Skurichina, R.P.W. Duin, Bagging for linear classifiers, Pattern Recognition 31 (7) (1998) 909–930.

[7] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1–2) (1999) 105–139.

[8] A.J.C. Sharkey, Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, Springer, London, 1999.

[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] G.I. Webb, Multiboosting: a technique for combining boosting and wagging, Mach. Learn. 40 (2) (2000) 159–196.

[11] G. Ratsch, T. Onoda, K.-R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (3) (2001) 287–320.

[12] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[13] T. Hothorn, B. Lausen, Double-bagging: combining classifiers by bootstrap aggregation, Pattern Recognition 36 (6) (2003) 1303–1309.

[14] R. Bryll, R. Gutierrez-Osuna, F. Quek, Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets, Pattern Recognition 36 (6) (2003) 1291–1302.

[15] G. Martínez-Muñoz, A. Suárez, Using all data to generate decision tree ensembles, IEEE Trans. Syst. Man Cyber. C 34 (4) (2004) 393–397.

[16] L. Breiman, Out-of-bag estimation, Technical Report, Statistics Department, University of California, 1996.

[17] S. Gelfand, C. Ravishankar, E. Delp, An iterative growing and pruning algorithm for classification tree design, IEEE Trans. Pattern Anal. Mach. Intell. 13 (2) (1991) 138–150.

[18] L. Breiman, Randomizing outputs to increase prediction accuracy, Mach. Learn. 40 (3) (2000) 229–242.

[19] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, 1998, URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

[20] J.R. Quinlan, Improved use of continuous attributes in C4.5, J. Artificial Intell. Res. 4 (1996) 77–90.

[21] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Chapman & Hall, New York, 1984.

[22] L. Breiman, Bias, variance, and arcing classifiers, Technical Report 460, Statistics Department, University of California, 1996.

About the Author—GONZALO MARTÍNEZ-MUÑOZ received the university degree in Physics and the M.Sc degree in Computer Science from the Universidad Autónoma de Madrid (UAM), Madrid, Spain, in 1995 and 2001, respectively. He is currently pursuing the Ph.D. degree. Currently, he is Assistant Professor of Object Oriented Programming and Experimental Algorithmics in the Computer Science Department of the Universidad Autónoma de Madrid, Madrid, Spain. From 1996 to 2002, he was with Geosys SL, a Spanish company specialized in geographical information systems and remote sensing, as a Software Designer and Developer in the framework of research and development projects. His research interests include pattern recognition, decision trees, machine learning, and genetic algorithms.

About the Author—ALBERTO SUÁREZ received the degree of Licenciado in Chemistry from the Universidad Autónoma de Madrid, Madrid, Spain, in 1988, and the Ph.D. degree in physical chemistry from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 1993. Currently, he is a Professor in the Computer Science Department of the Universidad Autónoma de Madrid, Madrid, Spain. He has held postdoctoral positions at Stanford University, Stanford, CA, the Université Libre de Bruxelles, Brussels, Belgium, and the Katholieke Universiteit Leuven, Leuven, Belgium. He has worked on relaxation theory in condensed media, stochastic and thermodynamic theories of nonequilibrium systems, lattice-gas automata, and decision tree induction. His research interests include machine learning, computational finance, and information processing in the presence of noise.