Pattern Recognition 83 (2018) 119–133

Contents lists available at ScienceDirect

Pattern Recognition

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

Vote-boosting ensembles

Maryam Sabzevari*, Gonzalo Martínez-Muñoz, Alberto Suárez

Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Escuela Politécnica Superior, C/Francisco Tomás y Valiente, 11, Madrid 28049, Spain

ARTICLE INFO

Article history:
Received 5 August 2017
Revised 4 May 2018
Accepted 20 May 2018
Available online 22 May 2018

Keywords:
Ensemble learning
Boosting
Uncertainty-based emphasis
Robust classification

ABSTRACT

Vote-boosting is a sequential ensemble learning method in which the individual classifiers are built on different weighted versions of the training data. To build a new classifier, the weight of each training instance is determined in terms of the degree of disagreement among the current ensemble predictions for that instance. For low class-label noise levels, especially when simple base learners are used, emphasis should be made on instances for which the disagreement rate is high. When more flexible classifiers are used and as the noise level increases, the emphasis on these uncertain instances should be reduced. In fact, at sufficiently high levels of class-label noise, the focus should be on instances on which the ensemble classifiers agree. The optimal type of emphasis can be automatically determined using cross-validation. An extensive empirical analysis using the beta distribution as emphasis function illustrates that vote-boosting is an effective method to generate ensembles that are both accurate and robust.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

In ensemble learning, the outputs of a collection of diverse classifiers are combined to exploit their complementarity, in the expectation that the global ensemble prediction be more accurate than the individual ones [17]. The complementarity of the classifiers is either an indirect consequence of diversity, as in bagging [9], random forests [13], and class-switching [45], or can be explicitly favored by design, as in negative correlation learning [41] and boosting [26,27,56,57]. In this manuscript, we present vote-boosting, an ensemble learning method of the latter type, in which the progressive focus on a particular training instance depends on the degree of disagreement among the predictions of the ensemble classifiers. By contrast to standard boosting algorithms, the strength of this emphasis is independent of whether the instance is correctly or incorrectly classified. The optimal emphasis profile depends on the characteristics of the classification problem considered and on the complexity of the base learners. In problems with no or low levels of noise in the class labels of the training instances the appropriate focus is on instances on which the classifiers disagree (i.e. instances for which the ensemble prediction has a high degree of uncertainty). As the noise level increases, especially when more flexible base learners (e.g. unpruned CART or random trees) are used, the emphasis on uncertain instances should be reduced. In fact, in problems with sufficiently high levels of class-label noise, the optimal strategy is to assign larger weights to instances on which the ensemble classifiers agree. In practice, the determination of the optimal emphasis strategy can be automatically made through parameter selection (e.g. through cross-validation on the training data). The results of an extensive empirical are used to illustrate that vote-boosting is an effective method to build ensembles that are both accurate and robust to class-label noise.

The article is organized as follows: In Section 2, we provide a review of ensemble methods that are related to the current proposal. Vote-boosting is described in Section 3. In this section, we show that the ensemble construction algorithm can be viewed as an optimization by gradient descent in the functional space of linear combinations of hypothesis. In Section 4, the properties and performance of vote-boosting ensembles are analyzed in an extensive empirical evaluation on synthetic and real-world classification tasks from different domains of application. Finally, the conclusions of this study are summarized in Section 5.

2. Previous work

There are a wide variety of methods to build ensembles. In this work, we focus on homogeneous ensembles, which are composed of predictors of the same type. Each of the predictors in the ensemble is built from a training set composed of labeled instances. Once the individual predictors have been built, their outputs are combined to reach a global ensemble decision. A wide range of alternatives can be used to carry out this combination [63]. Nevertheless, simple strategies, such as averaging real-valued outputs,

* Corresponding author.
E-mail address: maryam.sabzevari@uam.es (M. Sabzevari).

https://doi.org/10.1016/j.patcog.2018.05.022
0031-3203/© 2018 Elsevier Ltd. All rights reserved.

or majority voting, if the individual classifiers yield class-labels, are generally effective [28].

Randomization techniques can be used to generate collections of diverse classifiers. The objective is to build predictors that err on different examples. If the errors of these classifiers are independent, they can be averaged out by the combination process. An example of these types of ensembles is bagging [9]. The individual classifiers in a bagging ensemble are built by applying a fixed learning algorithm to independent bootstrap samples drawn from the original training data. In class-switching ensembles, each member is built using a perturbed versions of the original training set, in which the class labels of a fraction of instances are modified at random [12,45]. Alternatively, diverse classifiers can be built by including some randomized steps in the learning algorithm itself. For instance, in one of the earliest works on ensembles [35], one takes advantage of the presence of multiple local minima in the optimization process used to determine the synaptic weights of a multilayer perceptron. Starting from different initial seeds, it is possible to build neural networks with the same architecture, but different weights. Each of these different networks yields a different prediction. The final ensemble prediction can be obtained by pooling the individual decisions of the neural networks that result from the different weight initializations. Random forests [13], which are one of the most effective ensemble methods [20], are built using a combination of data randomization and randomization in the learning algorithm: The ensemble classifiers are random trees trained on bootstrap replicates of the original training dataset. The individual ensemble trees are generated using the random subspace method [37]. Other effective classifiers of this type are rotation forests [55], ensembles of extremely randomized trees [30], and other variants of random forest [64,65].

An alternative to simply generating diversity is to explicitly aim to increase the complementarity of the ensemble classifiers. An example of such strategy is Negative Correlation Learning [41]. In this method, complementarity is favored by simultaneously training all the classifiers in the ensemble: The parameters of the individual classifiers and the weights of the combination of their outputs are determined globally by minimizing a cost function that penalizes coincident predictions. One can also build ensembles of base learners that are trained to focus on different regions in feature space [5]. Boosting is another ensemble method in which complementarity among the classifiers is explicitly favored. Boosting originally refers to the problem of building a strong learner out of a collection of weak learners; i.e. learners whose predictive accuracy is only slightly better than random guessing [27,56,57]. AdaBoost is one of the most widely used boosting algorithms [26]. In its original formulation, AdaBoost considered only binary classification tasks. Nonetheless, there are numerous extensions to deal with multiclass problems (see e.g. the references in [21]). In AdaBoost an ensemble is grown by incorporating classifiers that progressively focus on instances that are misclassified by the previous classifiers in the sequence. The individual classifiers are built by applying a learning algorithm that can handle individual instance weights. Alternatively, weighted resampling in the training set can be used. The first classifier is obtained by assuming equal weights for all instances. The subsequent classifiers are built using different emphasis on each of the training instances. Specifically, to build the tth classifier in the sequence, the weights of instances that are misclassified by the most recent classifier in the ensemble are increased. Correspondingly, the weights of the correctly classified instances are reduced. The final prediction of the ensemble is determined by weighted majority voting. The weight of an individual classifier in the final ensemble prediction depends on the weighted accuracy of this classifier on the training set. The margin of an instance is defined as the sum of weighted votes for the correct class minus the sum of weighted votes for the most voted incorrect class. Therefore, misclassified instances have negative margins. In AdaBoost, the evolution of the weight of a particular instance is a monotonically decreasing function of its margin [25].

AdaBoost is one of the most effective ensemble methods [18,20,44]. However, it is not robust to class-label noise [7,39,53]. Specifically, AdaBoost gives unduly high weights to noisy instances, whose class labels are incorrect. There are numerous studies that address this excessive sensitivity of AdaBoost to class-label noise [1,14,15,19,23,24,27,31,33,38,43,54,59,61,62]. A possible strategy is to identify and either remove noisy instances in the training data, or correct their class-labels [1,29,43]. Modified weight update rules can be used for instances that are identified as noisy [14,60]. Another alternative is to apply explicit or implicit regularization techniques to avoid assigning excessive weight to a reduced group of instances [19,31,34,38,48,59]. For instance, the logistic loss function employed in LogitBoost [27] gives less emphasis to instances with large negative margins than the exponential loss function used in AdaBoost. In consequence, LogitBoost is generally more robust to class-label noise [49]. In other studies, penalty terms are used in the cost function to avoid focusing on outliers or on instances that are difficult to classify [33,54,61,62]. It is possible to also use hybrid weighting methods that modulate the emphasis on instances according to their distance to the decision boundary [2,3,31,32]. Most boosting algorithms use convex loss functions. This has the advantage that the resulting optimization problem can be solved efficiently using, for instance, gradient descent. However, as shown in [42], the generalization capacity of boosting variants that use convex loss functions can be severely affected by class-label noise. Alternative non-convex loss functions are used in BrownBoost and other robust boosting variants [15,23,24,46,50]. In these methods, the evolution of the weights is *not* a monotonic function of the margin: instances with small negative margins (i.e. misclassified instances that are close to the decision boundary) are assigned higher weights, as in AdaBoost. However, instances whose margin is negative and large receive lower weights. The rationale for using this type of emphasis is that instances in regions with a large class overlap tend to have small margins. Focusing on these instances is beneficial because the classification boundary can be modeled in more detail. By contrast, large negative margins correspond to misclassified instances that are far from the classification boundary. A robust boosting algorithm should therefore avoid emphasizing these instances, which are likely to be noisy.

In the next section we introduce vote-boosting, a novel boosting algorithm in which the weights of the instances are determined in terms of the degree of agreement or disagreement among the predictions of the ensemble members, irrespective of their actual class labels. As illustrated by the results of empirical evaluation presented in Section 4, the optimal type of emphasis (that is, whether the focus should be placed on instances on which the classifiers disagree, or on instances on which they agree) can be determined from the training data alone using, for instance, cross-validation. Since the instance weights do not depend on whether the predictions by the ensemble classifiers are correct, it is possible to avoid unduly emphasizing incorrectly classified instances that are outliers, which is one of the weaknesses of standard boosting algorithms, such as AdaBoost. In this manner, one can build accurate ensembles that are robust to class-label noise.

3. Vote-boosting

Consider the problem of automatic induction of a classification system from labeled data. The original training set is composed of N_{train} attribute class-label pairs $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$, where $\mathbf{x} \in \mathcal{X}$.

In this article, we focus on binary classification tasks, in which $y \in \{-1, 1\}$. Problems with multiple classes can be addressed with a number of strategies, such as the ones used in combination with AdaBoost for this purpose [57].

Let $\{f_\tau(\cdot)\}_{\tau=1}^{t}$ be a partially-grown ensemble of size t. The τth classifier in the ensemble is a function $f_\tau : \mathcal{X} \to \{-1, 1\}$ that maps a vector of attributes $\mathbf{x} \in \mathcal{X}$ to a class label $f_\tau(\mathbf{x}) \in \{-1, 1\}$. This function is obtained by applying a base learning algorithm to a training set, taking into account the individual instance weights $\left\{w_i^{[\tau]}\right\}_{i=1}^{N_{train}}$.

To obtain the prediction of the ensemble, the predictions of the individual classifiers are aggregated by weighted averaging

$$F_t(\mathbf{x}) = \sum_{\tau=1}^{t} \alpha_\tau^{[t]} f_\tau(\mathbf{x}), \quad F_t \in [-1, 1], \tag{1}$$

where $\alpha_\tau^{[t]} \geq 0$ is the weight of the prediction of the τ classifier. These weights are normalized $\sum_{\tau=1}^{t} \alpha_\tau^{[t]} = 1$. In (unweighted) majority voting one assumes that all the predictions have the same weight , $\alpha_\tau^{[t]} = 1/t$. This simple voting scheme provides good overall results. For this reason, it will be used in our implementation. Based on this aggregated output, the final prediction of the ensemble of size t is

$$H_t(\mathbf{x}) = sign(F_t(\mathbf{x})). \tag{2}$$

Assuming that t is odd, $H_t(\mathbf{x}) \in \{-1, 1\}$. At this stage of the ensemble construction process, instance $\mathbf{x}$ can be characterized by $t_+(\mathbf{x})$ and $t_-(\mathbf{x}) = t - t_+(\mathbf{x})$, the counts of positive and negative votes, respectively. The fractions of votes in each class are

$$\pi_\pm^{[t]}(\mathbf{x}) = \frac{t_\pm(\mathbf{x})}{t}; \quad \pi_+^{[t]}(\mathbf{x}) + \pi_-^{[t]}(\mathbf{x}) = 1. \tag{3}$$

These values can be used to quantify the level of certainty of the ensemble prediction. Values $\pi_+^{[t]}(\mathbf{x})$ close to 0 or 1 correspond to instances for which the predictions of most ensemble classifiers coincide. Instances whose classification by the ensemble is uncertain are characterized by $\pi_+^{[t]}(\mathbf{x})$ close to 1/2. In contrast to standard boosting algorithms, in vote-boosting, the instance weights depend on the degree of agreement or disagreement among the predictions of the individual classifiers, not on whether these predictions are erroneous. The pseudo-code of the proposed vote-boosting algorithm is presented in Fig. 1.

The final ensemble is composed of T classifiers, each of which is built by applying the base learning algorithm $\mathcal{L}$ on the training data with different sets of instance weights. The weights of the instances can be taken into account using weighted resampling with replacement

$$\begin{aligned} \mathcal{D}_{train}^{[t]} &= \text{Sample}\left(\mathcal{D}_{train}, \mathbf{w}^{[t]}\right) \\ f_t(\cdot) &\leftarrow \mathcal{L}\left(\mathcal{D}_{train}^{[t]}\right). \end{aligned} \tag{4}$$

For the induction of the first ensemble classifier all instances are assigned the same weight. These weights are updated at each iteration according to the tally of votes: Assuming that instance $\mathbf{x}_i$ has received $t_+(\mathbf{x}_i)$ votes for the positive class at the tth iteration, we use the Laplace estimator of the probability that a classifier in the ensemble outputs a particular class prediction

$$\pi_\pm^{[t]}(\mathbf{x}_i) = \frac{t_\pm(\mathbf{x}_i) + 1}{t + 2}. \tag{5}$$

Finally the weights are updated

$$w_i^{[t+1]} = \frac{1}{\mathcal{Z}_{t+1}} g\left(\pi_+^{[t]}(\mathbf{x}_i)\right), \quad \text{for } i = 1, \ldots, N_{train}, \tag{6}$$

where $g : [0, 1] \to \mathbb{R}^+$ is an emphasis function, which is non-negative, and

$$\mathcal{Z}_{t+1} = \sum_{i=1}^{N_{train}} g\left(\pi_+^{[t]}(\mathbf{x}_i)\right) \tag{7}$$

is a normalization constant. These weights are then used to build the following classifier in the ensemble.

A natural choice for the emphasis function is the probability density of the beta distribution with shape parameters a, b

$$g(p) \equiv \beta(p; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1 - p)^{b-1}, \quad 0 \leq p \leq 1. \tag{8}$$

For this particular emphasis function, the weights at the $(t + 1)$th iteration are updated according to

$$w_i^{[t+1]} = \frac{1}{Z_{t+1}} \beta\left(\pi_+^{[t]}(\mathbf{x}_i); a, b\right) \tag{9}$$

If the class distributions are not strongly imbalanced, the choice $a = b$, in which the two classes are handled in a symmetrical manner, is generally appropriate. In problems with a large class imbalance, an asymmetric choice of the emphasis function may be preferable. In Fig. 1 the density profiles of the symmetric beta distribution for different values of $a = b \in \{0.25, 0.75, 1, 1.5, 2.5, 5, 10, 20, 40\}$ are shown. If $a = b = 1.0$, the distribution is uniform. Therefore, all instances are given the same importance (plot in the first row, third column of Fig. 1). In this case, the proposed algorithm is equivalent to bagging [9]. For $a = b > 1.0$ the distribution becomes unimodal, with a maximum at 0.5. In this range, the higher the values of $a = b$, the more concentrated becomes the probability around the mode. In consequence, vote-boosting emphasizes uncertain instances and reduces the importance of those instances on which most classifiers agree. For simple classifiers, the emphasis on uncertain instances that one obtains is similar to the error-based emphasis of AdaBoost. The reason is that uncertain instances are generally more difficult to classify and, in consequence, are more likely to be incorrectly classified. In the regime $a = b < 1.0$, vote-boosting progressively focuses on instances on which classifiers agree. As will be illustrated in the section on experiments, this strategy is effective in complex or noisy problems, especially when the ensemble is composed of flexible classifiers, because of its regularizing effects.

It is common, especially in the first iterations of the algorithm, when the ensemble is still small, that all the classifiers predict the same class label for some instances. In such cases, the fraction of positive votes is either 0 or 1. Except for $a = b = 1$, the value of the beta distribution at these points is either zero or infinity. In the case of zero density values, those instances would be assigned zero weight in the next iteration of the algorithm. Thus, they would be effectively removed from the sample. In the other extreme, some instances would have infinite weights. To avoid these evaluations of the beta distribution at the boundaries of its support, the Laplace correction has been used in the estimation of class prediction probabilities (5).

Finally, we note that vote-boosting can be used with base learners that achieve zero or low error rates in the training data. In such cases, the weights that AdaBoost assigns to the training instances are ill-defined. By contrast, even if the training error of a single ensemble classifier is zero or close to zero, there can be disagreement among the individual predictions. Therefore, provided that the Laplace correction is used in the estimation of class prediction probabilities, the weights given by Eq. (9), are always well defined. This feature allows us to build vote-boosting ensembles of unpruned CART or random trees, which, as illustrated by the results of Section 4 are both accurate and robust to class-label noise.

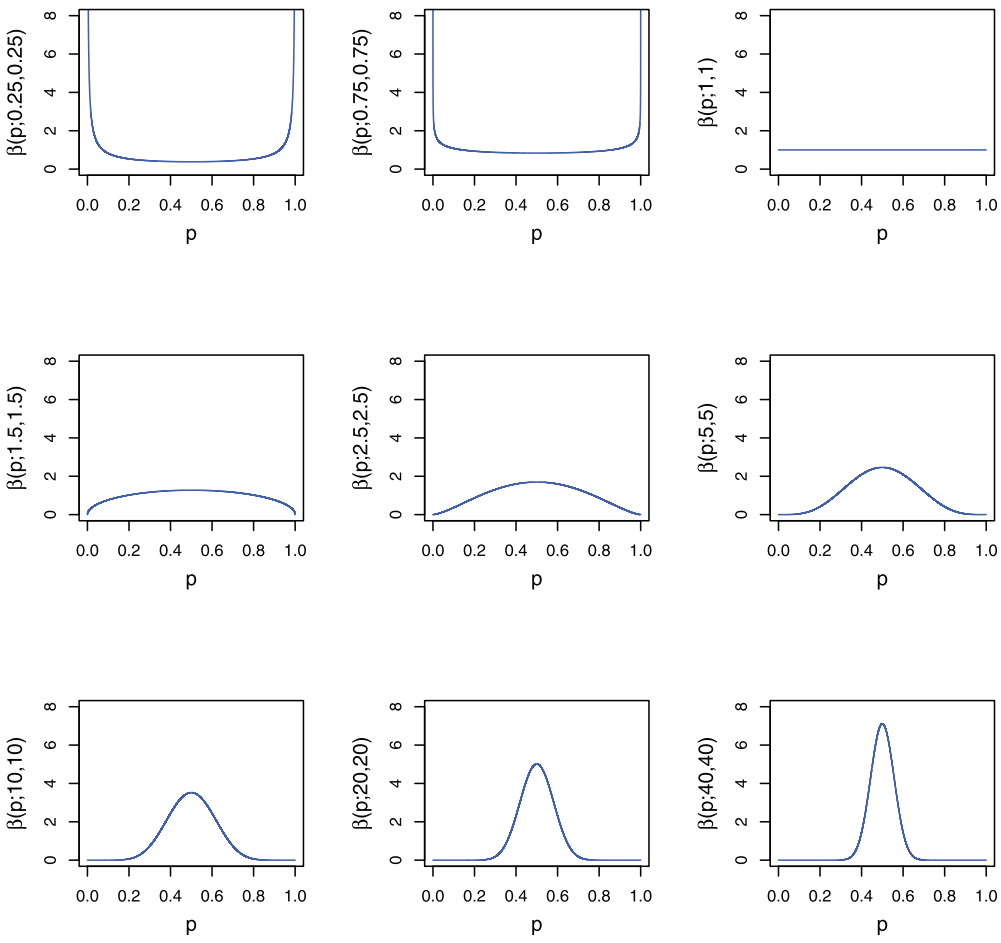


Fig. 1. Symmetric beta distribution with $a = b = [0.25, 0.75, 1, 1.5, 2.5, 5, 10, 20, 40]$.

3.1. An interpretation of vote-boosting as functional gradient descent

Similarly to other boosting methods, vote-boosting can be viewed as a gradient descent algorithm in the hypothesis space of linear combinations of predictors [47]. Consider an ensemble of t predictors $\{f_\tau\}_{\tau=1}^{t}$. The global ensemble prediction on instance $\mathbf{x}$ is of the form

$$H_t(\mathbf{x}) = sign[F_t(\mathbf{x})], \tag{10}$$

where

$$F_t(\mathbf{x}) = \frac{1}{t}\sum_{\tau=1}^{t} f_\tau(\mathbf{x}), \quad F_t(\mathbf{x}) \in [-1, 1]. \tag{11}$$

The fraction of votes for the positive class can be expressed in terms of this quantity

$$\pi_+^{[t]}(\mathbf{x}) = \frac{1 + F_t(\mathbf{x})}{2}. \tag{12}$$

Consider the predictor $F : \mathbf{x} \in \mathcal{X} \to F(x) \in [-1, 1]$. Let the cost functional for this predictor be

$$C[F] = \frac{1}{N_{train}}\sum_{i=1}^{N_{train}} y_i c(F(\mathbf{x}_i)), \tag{13}$$

where $c(z)$ is a monotonically non-increasing function of $z \in [-1, 1]$, such that $c(0) = 0$. These properties ensure that when the prediction of F for the ith example is class -1 (i.e. $F(\mathbf{x}_i) < 0$), then $c(F(\mathbf{x}_i)) > 0$. Similarly, when the prediction is class $+1$ (i.e. $F(\mathbf{x}_i) > 0$), then $c(F(\mathbf{x}_i)) < 0$. Therefore, when the prediction $F(\mathbf{x}_i)$ is incorrect (i.e. $y_i F(\mathbf{x}_i) < 0$), the contribution to the cost functional $y_i c(F(\mathbf{x}_i))$ is positive. When the prediction is correct, the corresponding contribution is negative. From these properties ones concludes that $C[F]$ achieves its global minimum when the training error is zero. Furthermore, this quantity increases with each incorrect prediction. In consequence, the minimizer of $C[F]$ also minimizes the error of predictor F in the training set.

If $|F(\mathbf{x}_i)|$ is a measure of how certain the prediction of F for instance $\mathbf{x}_i$ is, the value $|c(F(\mathbf{x}_i))|$ provides also a measure of such certainty. The magnitude of contribution $y_i c(F_t(\mathbf{x}_i))$ to the cost functional increases with the margin of the prediction. It is largest when $|F_t(\mathbf{x}_i)| = 1$; that is, when all ensemble classifiers agree.

In vote-boosting, the first classifier in the ensemble is built by assuming equal weights for all the instances in the training set. Then, the ensemble is grown in a sequential manner by incorporating to $\{f_\tau\}_{\tau=1}^{t}$, the ensemble of size t, the classifier that minimizes the value of cost functional for the enlarged ensemble $\{f_\tau\}_{\tau=1}^{t} \cup \{f_{t+1}\}$

$$f_{t+1} = \arg\min_{f \in \mathcal{F}} C\left[F_{t+1}^{[f]}\right], \tag{14}$$

where

$$\begin{aligned} F_{t+1}^{[f]}(\mathbf{x}) &= \frac{1}{t+1}\sum_{\tau=1}^{t} f_\tau(\mathbf{x}) + \frac{1}{t+1} f(\mathbf{x}) \\ &= F_t(\mathbf{x}) + \frac{1}{t+1}(f(\mathbf{x}) - F_t(\mathbf{x})). \end{aligned} \tag{15}$$

Assuming the change in the value of the cost functional when the ensemble incorporates the new classifier f is small

$$\begin{aligned}
&\delta C[F_t] \\
&\equiv C\left[F_{t+1}^{[f]}(\mathbf{x})\right] - C[F_t(\mathbf{x})] \\
&= \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} y_i \left[c\left(F_t(\mathbf{x}_i) + \frac{1}{t+1}(f(\mathbf{x}_i) - F_t(\mathbf{x}_i))\right) - c(F_t(\mathbf{x}_i)) \right] \\
&\approx \frac{1}{(t+1)N_{train}} \left[\sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) f(\mathbf{x}_i) - \sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) F_t(\mathbf{x}_i) \right],
\end{aligned} \tag{16}$$

to lowest order in the Taylor expansion. The second term in the last expression does not depend on f. Therefore, to lowest order, minimizing the cost functional is equivalent to minimizing

$$\begin{aligned}
\sum_{i=1}^{N_{train}} y_i c'(F_t(\mathbf{x}_i)) f(\mathbf{x}_i) &= \sum_{i:y_i = f(x_i)} c'(F_t(\mathbf{x}_i)) - \sum_{i:y_i \neq f(x_i)} c'(F_t(\mathbf{x}_i)) \\
&= \sum_{i=1}^{N_{train}} c'(F_t(\mathbf{x}_i)) - 2 \sum_{i:y_i \neq f(x_i)} c'(F_t(\mathbf{x}_i)) \\
&= 2 \sum_{j=1}^{N_{train}} c'(F_t(\mathbf{x}_j)) \left(\frac{1}{2} - \sum_{i:y_i \neq f(x_i)} w_i^{[t+1]} \right),
\end{aligned}$$

where

$$w_i^{[t+1]} = \frac{c'(F_t(\mathbf{x}_i))}{\sum_{j=1}^{N_{train}} c'\left(F_t(\mathbf{x}_j)\right)}. \tag{17}$$

Since $c(z)$ is a monotonic non-increasing function, then $-c'(z)$ is non-negative, and the values $\left\{w_i^{[t+1]}\right\}_{i=1}^{N_{train}}$ defined in Eq. (17) can be thought of as a set of instance weights. The denominator in (17) ensures that these weights are normalized

$$\sum_{i=1}^{N_{train}} w_i^{[t+1]} = 1. \tag{18}$$

Using this expression for the instance weights, Eq. (16) becomes

$$\delta C[F_t] \propto \sum_{i:y_i \neq f(x_i)} w_i^{[t+1]} - \sum_{i:y_i \neq F_t(x_i)} w_i^{[t+1]}. \tag{19}$$

Note that $\delta C[F_t] < 0$ only if the weighted training error of the newly built classifier is lower than the corresponding error for the ensemble.

Under these conditions, the $(t+1)$th predictor in the ensemble is the minimizer of the *weighted* training error

$$f_{t+1} = \arg\min_{f \in \mathcal{F}} \sum_{i:y_i \neq f(x_i)} w_i^{[t+1]}, \tag{20}$$

where $\mathcal{F}$ is the functional space of the base learners. In contrast to standard boosting algorithms, the weights given by (17) depend only on the ensemble predictions, irrespective of whether these predictions are correct.

Assuming that the function $c(z)$ in (13) is bounded, it is convenient to express it in terms of a cumulative distribution function $G(\pi)$ defined in the unit interval $\pi \in [0, 1]$

$$c(F(\mathbf{x})) = 2\, K \left[G(1/2) - G\left(\frac{1 + F(\mathbf{x})}{2} \right) \right], \tag{21}$$

where K is a positive constant. Without loss of generality, this constant is set to one ($K = 1$). Because of the monotonicity of $G(p)$, when the prediction $F(\mathbf{x}_i)$ is incorrect (i.e. $y_i F(\mathbf{x}_i) < 0$), the contribution to the cost functional $y_i c(F(\mathbf{x}_i))$ is positive. Assuming this form for $c(F(\mathbf{x}))$, its derivative is

$$c'(F(\mathbf{x})) = -g\left(\frac{1 + F(\mathbf{x})}{2} \right), \tag{22}$$

where $g(p) = G'(p)$ is the corresponding probability density, which is non-negative. With these assumptions, the weights of the training instances are

$$w_i^{[t+1]} = \frac{g\left(\pi_+^{[t]}(\mathbf{x}_i)\right)}{\sum_{j=1}^{N_{train}} g\left(\pi_+^{[t]}(\mathbf{x}_j)\right)}, \quad i = 1, 2, \ldots, N_{train}, \tag{23}$$

where the density $g(p)$ plays the role of an emphasis function. Note that this density need not be symmetric around $\pi_+^{[t]}(\mathbf{x}_i) = \frac{1}{2}$. In fact, asymmetries in the emphasis could be useful in classification problems with unbalanced classes. In contrast to most boosting algorithms, including AdaBoost, the weights given by Eq. (23) do not depend on the actual class label of the instance. Therefore, it does not seem possible to derive error bounds similar to those enunciated in Theorem 6 (e.g. Eq. (21)) of [26].

4. Empirical evaluation

In this section we present the results of an empirical analysis of vote-boosting. Different sets of experiments have been performed to analyze the properties ensembles built with this method and evaluate their accuracy in a wide range of classification tasks from different areas of application. In these experiments, the symmetric beta distribution has been used as the emphasis function. A first set of experiments is carried out to investigate the relationship of vote-boosting with bagging and AdaBoost. The results of these experiments illustrate that, when simple (e.g. decision stumps) or regularized learners (e.g. puned CART trees) are used as base learners, vote-boosting performs an interpolation between bagging ($a = b = 1.0$) and AdaBoost (high values of $a = b$). In a second set of experiments, we investigate the behavior of vote-boosting composed of different classifiers. In particular, we compare the accuracies of vote-boosting ensembles composed of decision stumps, pruned CART trees, unpruned CART trees, and (unpruned) random trees. The best overall results in terms of predictive accuracy are obtained with random trees. However, the differences with vote-booting ensembles composed of pruned or unpruned CART trees are not statistically significant. Finally, the accuracy of vote-boosting ensembles composed of random trees is compared with bagging, AdaBoost and random forest. From the results of this benchmarking exercise, we conclude that, in the problems investigated, vote-boosting ensembles composed of random trees achieve state-of-the-art classification accuracy rates. These rates are comparable or superior to random forest and AdaBoost. A final batch of experiments is carried out to analyze the differences among the optimal emphasis profiles for different classification problems using random trees as base learners. This analysis illustrates that in problems with low levels of noise in the class labels, new classifiers should focus on instances whose classification by the current ensemble is uncertain By contrast, in problems with contaminated labels, the optimal emphasis is to reduce the weights of such uncertain instances, which are likely to be noisy.

4.1. Vote-boosting as an interpolation between bagging and AdaBoost

The objective of the experiments presented in this subsection is to analyze how the behavior of vote-boosting ensembles composed of simple or regularized learners, such as decision stumps, or pruned CART trees, changes when different levels of emphasis on the uncertain training instances are considered. As discussed earlier, when uniform emphasis is made, vote-boosting is equivalent to bagging. In most of the problems analyzed, when such simple base learners are used, stronger emphasis on uncertain instances (i.e. instances for which the degree of disagreement among the ensemble predictions is largest) results in a behavior

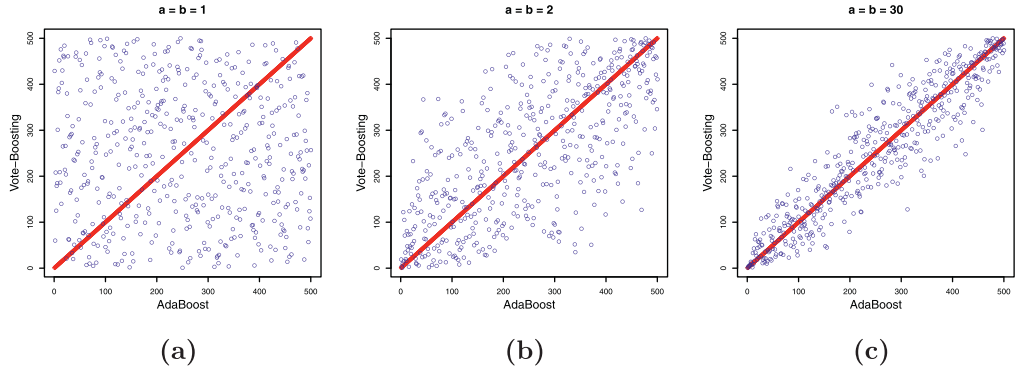


Fig. 2. Weight ranks of the training instances for vote-boosting and AdaBoost of decision stumps in *Twonorm* (a) $a = b = 1.0$, (b) $a = b = 2.0$, (d) $a = b = 30.0$.

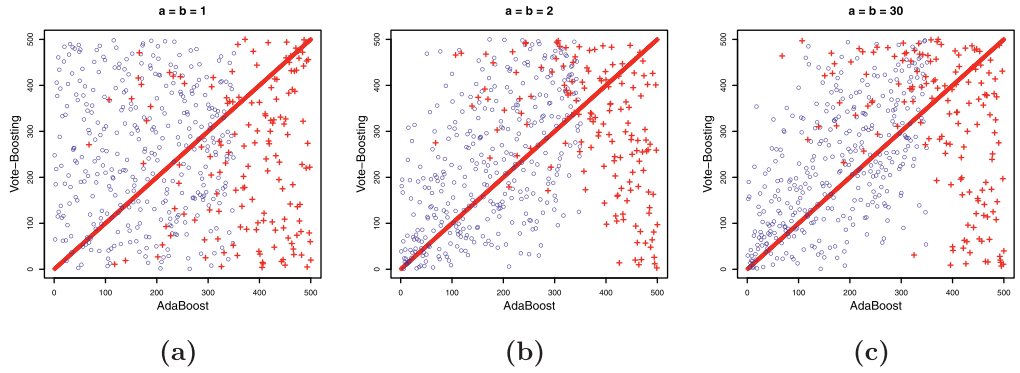


Fig. 3. Weight ranks of the training instances for vote-boosting and AdaBoost of decision stumps in *Twonorm* with 30% class-label noise, (a) $a = b = 1.0$, (b) $a = b = 2.0$, (c) $a = b = 30.0$.

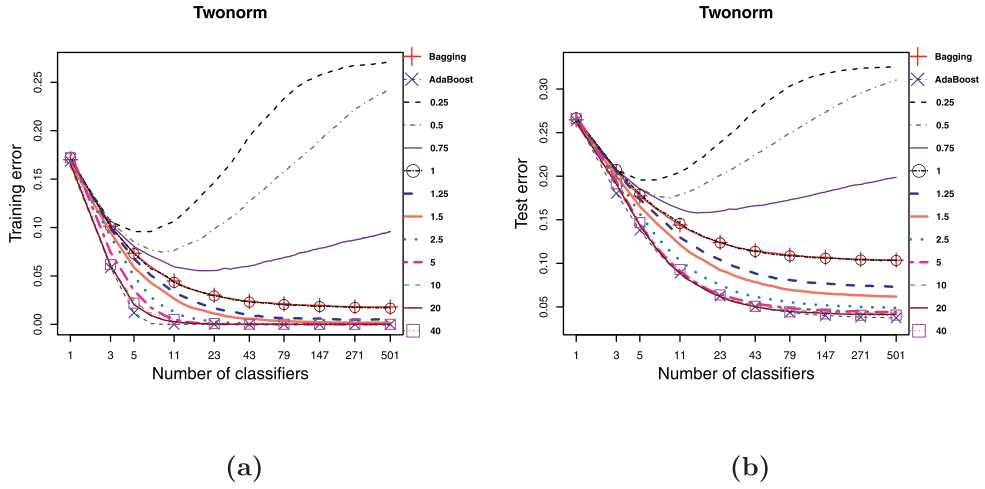


Fig. 4. Error rate as a function of number of classifiers in *Twonorm* for bagging, AdaBoost, and vote-boosting ensembles of pruned CART trees: (a) training error (b) test error.

that is similar to AdaBoost. In such cases, vote-boosting provides an interpolation between bagging and AdaBoost, depending on the strength of the emphasis on uncertain instances.

To investigate this relationship between vote-boosting and AdaBoost, we first present the results of an experiment in the binary classification problem *Twonorm* using decision stumps as base learners. In *Twonorm*, instances are drawn from two unit-variance Gaussians in 20 dimensions whose means are $(a, a, \ldots, a)$ for class 1 and $(-a, -a, \ldots, -a)$ for class 2, with $a = 2/\sqrt{20}$ [10]. This is not a trivial task for decision stumps because, as individual classifiers, they can model only class boundaries that are parallel to the axes. In the experiments performed, the training set is composed of 500 independently generated instances. Different vote-boosting ensembles composed of 100 stumps were built using the symmetric beta distribution for emphasis, with $a = b \in \{1.0, 2.0, 30.0\}$. An AdaBoost ensemble composed of 100 stumps was also built using the same training data. The final weights given to the training instances in the different ensembles were recorded and subsequently ranked. Ties were resolved by randomizing the corresponding ranks. In Fig. 2, a scatter plot of these ranks is shown. The position along the horizontal axis corresponds to the rank assigned by AdaBoost. Correspondingly, the location along the vertical axis corresponds to the ranks assigned by vote-boosting with $a = b = 1.0$ in (a), $a = b = 2.0$ in (b), and $a = b = 30.0$ in (c). When

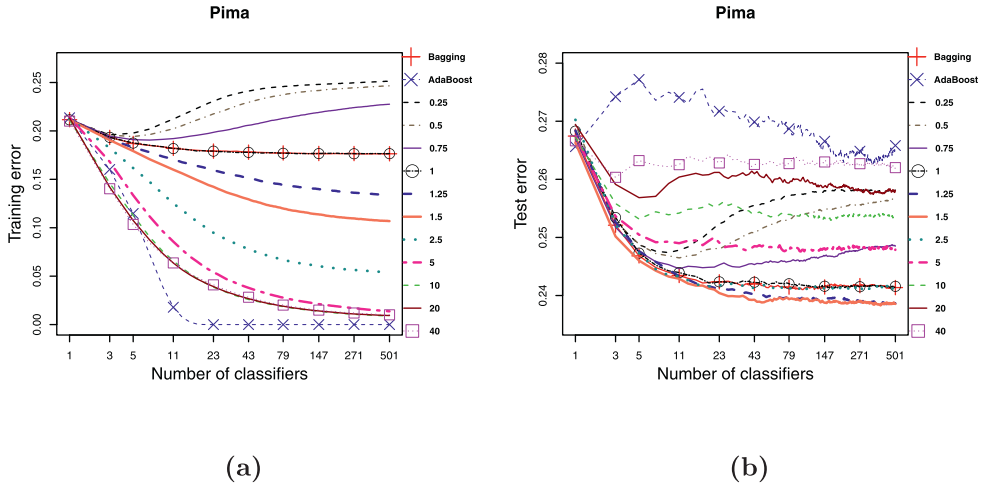


Fig. 5. Error rate as a function of the number of classifiers in *Pima* for bagging, AdaBoost, and vote-boosting ensembles of pruned CART trees: (a) training error (b) test error.

Table 1
Characteristics of the classification problems analyzed and training / test partitions.

Dataset	Instances	Training	Test	Number of attributes
Adult	32,561	21,707	10,854	15
Australian	690	460	230	14
Breast W.	699	466	233	9
Blood	748	499	249	5
Boston	506	337	169	14
Chess	3196	2131	1065	36
German	1000	667	333	20
Heart	270	178	92	13
Hepatitis	155	104	51	19
Horse-colic	368	246	122	21
Ionosphere	351	234	117	34
Liver	345	230	115	6
Magic	19,020	12,680	6340	11
Musk	6598	4399	2199	168
Ozone	2536	1691	845	74
Parkinsons	197	132	65	24
Pima	768	512	256	8
Ringnorm	2300	300	2000	20
Sonar	208	139	69	60
Spambase	4601	3068	1533	58
Threenorm	2300	300	2000	20
Tic-tac-toe	958	639	319	9
Twonorm	2300	300	2000	20

Table 2
Test error rates of vote-boosting ensembles composed of decision stumps, pruned CART trees, unpruned CART trees and random trees.

Dataset	Stump	Pruned	Unpruned	Random tree
Adult	20.9 ± 2.9	13.6 ± 0.3	**13.5 ± 6.2**	13.6 ± 0.2
Australian	14.6 ± 2.1	13.6 ± 2.0	13.5 ± 2.1	**13.3 ± 2.1**
Breast W.	3.9 ± 1.0	3.8 ± 1.1	4.0 ± 1.2	**3.1 ± 1.1**
Blood	22.7 ± 1.5	22.2 ± 1.7	**21.8 ± 1.8**	21.9 ± 1.7
Boston	15.2 ± 2.3	**12.7 ± 2.5**	13.0 ± 2.6	13.1 ± 2.3
Chess	17.7 ± 14.1	0.6 ± 0.6	0.7 ± 0.6	**0.6 ± 0.3**
German	25.4 ± 1.6	24.1 ± 2.1	**24.1 ± 2.0**	24.3 ± 1.7
Heart	**16.4 ± 3.5**	17.8 ± 3.6	18.4 ± 3.9	16.8 ± 3.5
Hepatitis	16.4 ± 4.3	16.2 ± 4.8	16.9 ± 4.9	**14.0 ± 3.9**
Horse-Colic	**13.9 ± 2.7**	14.1 ± 2.9	13.9 ± 2.7	15.0 ± 2.5
Ionosphere	8.1 ± 1.7	6.7 ± 2.0	6.7 ± 1.8	**6.6 ± 1.8**
Liver	**27.5 ± 3.9**	28.4 ± 3.6	28.6 ± 3.9	28.4 ± 3.7
Magic	15.4 ± 0.3	12.7 ± 0.3	12.9 ± 0.3	**11.6 ± 0.2***
Musk	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
Ozone	6.3 ± 0.0	**5.8 ± 0.4**	5.8 ± 0.4	6.0 ± 0.2
Parkinsons	11.5 ± 4.3	**7.4 ± 3.6**	7.6 ± 3.2	9.3 ± 3.6
Pima	24.1 ± 2.0	24.2 ± 2.2	23.8 ± 2.3	**23.4 ± 1.8**
Ringnorm	9.3 ± 0.9	4.5 ± 0.8	4.6 ± 0.8	**4.4 ± 0.8**
Sonar	18.9 ± 4.6	16.0 ± 4.4	15.8 ± 4.8	**15.5 ± 4.5**
Spambase	6.1 ± 0.6	4.5 ± 0.3	4.5 ± 0.3	**4.1 ± 0.5**
Threenorm	20.1 ± 1.0	17.1 ± 1.1	17.0 ± 1.0	**16.2 ± 1.0***
Tictactoe	22.0 ± 2.9	**0.7 ± 0.6**	0.7 ± 0.6	1.1 ± 0.7
Twonorm	4.6 ± 0.7	4.2 ± 0.6	4.2 ± 0.6	**3.6 ± 0.5***

$a = b = 1.0$ is used, all instances are assigned the same weight and no correlations between the weight ranks given by vote-boosting and AdaBoost is observed. Therefore, the points corresponding to the training instances appear uniformly distributed in Fig. 2 (a). As the values of $a = b$ increase, points tend to cluster around the diagonal. This is a consequence of the fact that the ranks of the weights given by both types of ensembles become more similar. Comparing Fig. 2 (a)–(c), it is apparent that the correlations between the weight ranks become stronger as $a = b$ increases. In particular for $a = b = 30.0$ vote-boosting and AdaBoost give similar emphasis, even though the former does not make use of class labels to decide whether an instance should be given more weight, whereas the latter does. The reason for this coincident emphasis is that, in this simple problem, the ensemble classifiers are more likely to disagree precisely in the instances that are incorrectly classified.

If class-label noise is injected in the problem, the weighting schemes of AdaBoost and vote-boosting become different: Vote-boosting maintains the focus on instances in the boundary region, in which classes overlap and the disagreement rates among the ensemble predictions are highest. By contrast, AdaBoost tends to give more weight to those instances whose class label has been modified. Focusing on these noisy instances is misleading and eventually impairs the generalization capacity of AdaBoost. To illustrate this observation, the experiment was repeated injecting class-label noise in the training data. Specifically, 30% of the training examples were selected at random and their class labels flipped, as in the noisy completely at random model described in [22]. The results of these experiments are shown in Fig. 3. Instances whose class label has been switched are marked with a red cross in these plots. For $a = b = 1.0$, no correlation is observed between the ranks of the weights given by vote-boosting and AdaBoost. However, instances whose class label has been switched, which are distributed uniformly in the vertical direction, tend to appear on the right-hand side of the plots. This means that they receive special emphasis in AdaBoost but not in vote-boosting. Increasing the value of $a = b$ pushes the unperturbed instances towards the diagonal but not the perturbed ones. However, the correlation between the ranks is less marked than in the noiseless case, because of the interference of the noisy instances.

A second batch of experiments was carried out to analyze the behavior of vote-boosting composed of pruned CART trees as a function of the strength of the emphasis that is applied to uncertain instances. Using the symmetric beta distribution as emphasis function, we analyze how the learning curves, which trace the dependence of the error as a function of the size of the ensemble, depend on the value of the shape parameter. The values explored are $a = b \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.5, 5, 10, 20, 40\}$. The experiments were made on the classification tasks *Twonorm* and *Pima*. These tasks have been chosen because of the different prediction accuracies of bagging and AdaBoost on those datasets. In *Twonorm*, AdaBoost significantly outperforms bagging. By contrast in *Pima*, which is a very noisy task, bagging is more accurate than AdaBoost.

Figs. 4 and 5 display the learning curves for bagging, AdaBoost, and vote-boosting using different values of the shape parameter in the classification tasks *Twonorm* and *Pima*, respectively. The plots on the left-hand side show the results for the training error. The plots on the right-hand side correspond to the test error curves. When $a = b = 1.0$, all instances are given equal weights. In this case, if weighted resampling is used, vote-boosting is equivalent to bagging. This is apparent from Figs. 4 and 5: the error curves of both methods are very close to each other. When values $a = b < 1.0$ are used, emphasis is made on instances on which most predictors agree. If regularized classifiers, such as pruned CART trees, are used as base learners, this type of emphasis is in general not effective. Nonetheless, as will be illustrated by the results presented in Section 4.3, focusing on these types of instances can lead to improvements in the generalization capacity when the data are very noisy and the ensemble is composed of flexible classifiers that overfit (e.g. unpruned CART or random trees).

Values of $a = b > 1.0$ correspond to emphasizing instances in which the ensemble prediction is uncertain. For *Twonorm*, the learning curves of vote-boosting using $a = b = 40.0$ and AdaBoost are quite similar. In this problem, the classification errors are mainly due to the overlap between the distributions of the two classes. Therefore, the incorrectly classified instances are close to the decision boundary, where the ensemble predictions are also more uncertain. Using a beta distribution sharply peaked around $\pi = 0.5$ (see Fig. 1) gives more weight to these uncertain instances. In consequence, the resulting emphasis is similar to AdaBoost's. In *Pima*, which is a noisy problem, the learning curves of vote-boosting with large $a = b$ and AdaBoost are different. Still, the closest performance to AdaBoost is vote-boosting with large $a = b$. Finally, we observe that the optimal value for the shape parameter of the beta distribution in vote-boosting is problem dependent: Values of $a = b \approx 1.25 - 1.5$ perform well in *Pima*. The best performance in *Twonorm* requires using a large values of $a = b \approx 40$. For each problem, the optimal value can be determined using cross-validation.

4.2. Vote-boosting with different base learners

In this section, we present the results of a comparison of vote-boosting ensembles composed of different base learners. These are, in order of increasing complexity, decision stumps, pruned and unpruned CART trees, and unpruned random trees. Ensembles composed of 501 classifiers are built. This fairly large ensemble size is needed in some problems to achieve convergence to the asymptotic error level [36]. Weighted resampling with replacement is used to generate the bootstrap samples on which the individual classifiers are trained. In vote-boosting, the symmetric beta distribution is used for emphasis. The shape parameter is determined as the value among those in $a = b \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 2.5, 5, 10, 20, 40\}$ that minimizes the 10-fold cross-validation in the training set. When uniform emphasis in all the training instances ($a = b = 1.0$) is used, the results are equivalent to bagging or, if random trees are used as base learners, to random forest. For values of the shape parameter above 1.0, the symmetric beta distribution has a single mode at $\pi = 0.5$, which implies that more emphasis is made on training instances on which the degree of disagreement among the different classifiers is large. By contrast, when the shape parameter is smaller than 1 the focus is on training instances on which most classifiers agree.

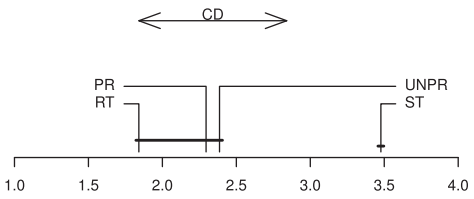


Fig. 6. Comparison of the average ranks of decision stumps (ST), pruned CART trees (PR), unpruned CART trees (UNPR), and random trees (RT) using a Nemenyi test. Horizontal lines connect methods whose average ranks are not significantly different (p-value $< .05$).

Table 3
Median of the beta distribution parameter ensembles composed of decision stumps, pruned CART trees, unpruned CART trees, and random trees. An asterisk is shown when all values of the beta parameter yield the same cross-validation error.

Dataset	Stump	Pruned	Unpruned	Random tree
Adult	40.0	40.0	20.0	1.0
Australian	*	5.0	2.5	1.5
Breast W.	10.0	5.0	10.0	1.0
Blood	40.0	1.25	0.75	0.5
Boston	40.0	20.0	10.0	2.0
Chess	20.0	20.0	20.0	10.0
German	40.0	5.0	2.5	2.5
Heart	10.0	2.5	1.5	0.5
Hepatitis	10.0	10.0	5.0	2.5
Horse-Colic	20.0	1.5	1.5	1.5
Ionosphere	40.0	5.0	10.0	2.5
Liver	40.0	5.0	5.0	1.5
Magic	40.0	40.0	40.0	20.0
Musk	*	*	*	0.7
Ozone	40.0	10.0	5.0	1.5
Parkinsons	40.0	15.0	10.0	5.0
Pima	15.0	1.5	1.0	0.5
Ringnorm	40.0	20.0	20.0	20.0
Sonar	40.0	20.0	20.0	10.0
Spambase	40.0	20.0	20.0	20.0
Threenorm	40.0	10.0	10.0	5.0
Tictactoe	40.0	20.0	20.0	10.0
Twonorm	20.0	20.0	20.0	5.0

For the empirical evaluation carried out in this an the following section, different binary classification tasks from the UCI repository [6] and other sources [11] are considered. The characteristics of the datasets used in this study are summarized in Table 1. For each dataset, the table displays the total number of labeled instances available, the number of those instances used for training and for testing, and the number of attributes. The test error rates reported are averages, followed by the corresponding standard deviations after the $\pm$ sign, over 100 realizations of the training and test sets. For classification problems in which only a finite collection of labeled instances is available, 2/3 of the data are selected at random for training and the remaining 1/3 for testing. For synthetic problems (namely, *Ringnorm, Twonorm*, and *Threenorm*), instances are generated independently at random: 300 instances are used for training and 2000 for testing. In all cases, stratified sampling is used to ensure that the class distributions in the test and training sets are similar.

In Table 2, the average test errors over the 100 realizations are shown for all base learners. For each dataset, the lowest average generalization error is highlighted in boldface. The second

Table 4
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a = b (median)
Adult	0	14.6 ± 0.2	13.8 ± 0.2	13.6 ± 0.2	**13.6 ± 0.2**	1.0
	10	15.5 ± 0.3	14.3 ± 0.3	14.1 ± 0.3	**14.0 ± 0.3**	0.75
	20	17.0 ± 0.3	14.8 ± 0.2	14.8 ± 0.3	**14.3 ± 0.2**	0.25
	30	20.3 ± 0.3	15.1 ± 0.3	17.8 ± 0.3	**15.1 ± 0.2**	0.25
Australian	0	13.3 ± 2.1	13.7 ± 1.9	**13.0 ± 2.0**	13.3 ± 2.1	1.5
	10	14.9 ± 2.3	17.6 ± 2.4	**13.6 ± 2.0**	13.6 ± 2.1	0.75
	20	18.2 ± 2.9	24.3 ± 3.0	15.8 ± 2.2	**14.3 ± 2.4***	0.25
	30	24.6 ± 3.8	32.0 ± 3.5	21.6 ± 3.0	**18.2 ± 3.3***	0.25
Breast W.	0	4.6 ± 1.2	3.6 ± 1.0	**3.0 ± 1.0**	3.1 ± 1.1	1.0
	10	6.3 ± 1.6	6.9 ± 1.8	4.1 ± 1.2	**3.5 ± 1.2***	0.5
	20	9.6 ± 2.5	12.0 ± 2.6	6.5 ± 1.9	**4.1 ± 1.4***	0.25
	30	17.7 ± 3.0	20.9 ± 3.4	13.0 ± 2.7	**6.8 ± 2.6***	0.25
Blood	0	26.3 ± 1.9	25.4 ± 2.1	24.5 ± 2.2	**21.9 ± 1.7***	0.5
	10	28.5 ± 2.2	27.2 ± 2.5	26.6 ± 2.4	**22.6 ± 2.0***	0.25
	20	31.6 ± 2.7	30.2 ± 2.6	29.7 ± 2.6	**23.8 ± 2.5***	0.25
	30	35.8 ± 3.6	34.6 ± 3.8	34.2 ± 3.6	**29.0 ± 4.5***	0.25
Boston	0	14.2 ± 2.4	**12.7 ± 2.4**	13.0 ± 2.4	13.1 ± 2.3	2.0
	10	16.2 ± 2.8	16.9 ± 2.5	**14.5 ± 2.6**	14.9 ± 2.5	0.5
	20	19.1 ± 3.3	22.2 ± 3.6	17.5 ± 3.2	**16.2 ± 2.8**	0.5
	30	26.4 ± 4.2	31.4 ± 4.0	24.6 ± 3.8	**21.3 ± 4.4***	0.25
Chess	0	0.6 ± 0.3	**0.5 ± 0.3**	1.6 ± 0.4	0.6 ± 0.3	10.0
	10	5.1 ± 0.7	2.3 ± 0.5	2.1 ± 0.5	**2.1 ± 0.5**	1.25
	20	11.9 ± 1.2	**3.6 ± 0.7**	4.3 ± 0.9	4.0 ± 0.9	0.75
	30	22.4 ± 1.5	**5.5 ± 0.9***	10.2 ± 1.3	7.9 ± 1.5	0.25
German	0	24.5 ± 2.0	24.8 ± 2.1	**24.0 ± 1.8**	24.3 ± 1.7	2.5
	10	26.2 ± 2.1	27.9 ± 2.1	25.3 ± 1.8	**25.3 ± 1.9**	1.0
	20	28.8 ± 2.4	32.2 ± 2.5	**27.3 ± 2.2**	27.4 ± 2.3	0.75
	30	32.3 ± 3.0	37.4 ± 2.8	31.0 ± 3.1	**29.9 ± 3.2**	0.5
Heart	0	19.5 ± 3.7	20.2 ± 3.4	17.2 ± 3.0	**16.8 ± 3.5**	0.5
	10	22.7 ± 4.2	24.9 ± 4.5	19.3 ± 3.7	**18.7 ± 4.0**	0.5
	20	26.0 ± 5.0	30.6 ± 5.1	22.7 ± 4.2	**20.8 ± 4.0**	0.375
	30	32.1 ± 5.4	36.3 ± 6.0	28.5 ± 5.8	**25.8 ± 6.6**	0.25
Hepatitis	0	15.8 ± 4.7	15.8 ± 4.0	**13.6 ± 3.5**	14.0 ± 3.9	2.5
	10	17.1 ± 4.7	20.0 ± 5.0	**14.2 ± 3.9**	15.3 ± 3.9	1.0
	20	21.0 ± 5.8	26.0 ± 6.9	17.6 ± 4.8	**17.3 ± 4.5**	0.5
	30	27.8 ± 7.8	33.6 ± 7.2	23.8 ± 7.3	**21.9 ± 7.0**	0.5
Horse-colic	0	**14.9 ± 2.8**	15.4 ± 2.6	15.0 ± 2.5	15.0 ± 2.5	1.5
	10	17.8 ± 3.5	21.2 ± 3.5	**17.3 ± 2.9**	17.4 ± 3.0	1.0
	20	23.4 ± 4.2	27.6 ± 4.7	21.3 ± 3.6	**20.9 ± 3.7**	0.625
	30	30.0 ± 5.6	34.9 ± 5.1	28.4 ± 4.6	**28.2 ± 5.3**	0.75

best result is underlined. If the differences between the two lowest test errors is statistically significant at a significance level $\alpha = 0.05$ the lowest error value is marked with an asterisk (*). A resampled paired t-test is used for synthetic problems. When random train/test partitions of the same dataset are employed a corrected resampled paired t-test [8,51] is used instead.

To provide an overall comparison of the accuracies of vote-boosting using the four tested base learners, we apply the framework proposed in [16]. To this end, the average rank of the classifier average errors is computed for the 23 classification problems investigated. The rank of a classifier in a specific classification problem is determined by ordering the different methods according to their test errors. A lower rank corresponds to a smaller test error and, therefore, better accuracy. The average ranks of vote-boosting using the four different base learners are displayed in Fig. 6. In this diagram, the differences of average ranks between methods that are connected by a horizontal solid line are not statistically significant according to a Nemenyi test (p-value $< .05$).

From the results presented in Table 2 and Fig. 6, one concludes that vote-boosting composed of random trees has the best overall accuracy: except in *Boston, Horse-colic*, and *Parkinsons*, these types of ensembles achieve the lowest or second lowest average test errors. Notwithstanding, according to the Nemenyi test, the average rank differences are statistically significant only with respect to the use of stumps (see Fig. 6).

The values of the shape parameters of the symmetric beta distribution ($a = b$) selected by cross-validation are shown in Table 3. The figures reported are medians over the 100 realizations of the training and test data. An asterisk is shown in the Table when ensembles built using the different values have the same within-train cross-validation error. For decision stumps, this occurs in *Australian* and *Musk* because the individual stumps are the same irrespective of the type of emphasis employed. In *Musk*, when more complex base classifiers are used, the ensembles built with the different values of $a = b$ all achieve zero error. In most cases, the value of the shape parameter of the beta distribution ($a = b$) decreases as the complexity of the base classifier increases. The reason of this is that more emphasis is needed in order to *boost* more stable base classifiers. Thus, the largest values of $a = b$ are selected for decision stumps. Hence, for these types of base learners the focus is on uncertain instances, on which most ensemble classifiers disagree. The lowest values of $a = b$ correspond to random trees. In this case, the emphasis on uncertain instances is reduced. In fact, for *Blood, Heart, Musk*, and *Pima*, the focus should be on instances on which the ensemble classifiers agree.

In summary, vote-boosting ensembles composed of random trees provide the best overall accuracy in the problems investigated. Since this type of ensemble can also be built efficiently, they will be used for further evaluation in the following set of experiments.

Table 5
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a = b (median)
Ionosphere	0	8.0 ± 2.2	**6.6 ± 1.8**	6.6 ± 1.6	6.6 ± 1.8	2.5
	10	9.5 ± 2.7	10.5 ± 2.6	7.9 ± 2.2	**7.9 ± 2.2**	0.75
	20	13.1 ± 3.1	17.0 ± 3.5	10.9 ± 2.9	**9.9 ± 3.1**	0.5
	30	19.7 ± 4.5	26.4 ± 4.5	17.8 ± 4.6	**15.7 ± 5.1**	0.25
Liver	0	29.5 ± 3.9	30.5 ± 3.9	**27.7 ± 3.8**	28.4 ± 3.7	1.5
	10	31.7 ± 4.1	33.8 ± 4.5	**31.0 ± 4.0**	31.4 ± 3.8	1.0
	20	36.1 ± 4.3	37.8 ± 4.5	**35.1 ± 4.4**	35.7 ± 4.4	1.0
	30	39.9 ± 4.7	41.5 ± 4.7	**39.6 ± 4.5**	39.7 ± 4.9	0.875
Magic	0	12.3 ± 0.3	12.9 ± 0.3	12.0 ± 0.3	**11.6 ± 0.2***	20.0
	10	12.9 ± 0.3	13.9 ± 0.3	**12.4 ± 0.2**	12.5 ± 0.2	1.25
	20	14.2 ± 0.4	14.4 ± 0.3	13.6 ± 0.4	**13.3 ± 0.4***	0.5
	30	17.4 ± 0.5	15.0 ± 0.5	16.7 ± 0.4	**14.6 ± 0.4***	0.25
Musk	0	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**	0.7
	10	1.4 ± 0.3	0.6 ± 0.2	1.6 ± 0.3	**0.1 ± 0.1***	0.25
	20	4.5 ± 0.4	0.9 ± 0.6	5.7 ± 0.4	**0.8 ± 0.3**	0.25
	30	12.0 ± 1.0	**2.5 ± 1.2***	14.4 ± 1.2	5.1 ± 0.7	0.25
Ozone	0	6.0 ± 0.3	**5.7 ± 0.4***	6.0 ± 0.2	6.0 ± 0.2	1.5
	10	6.2 ± 0.3	6.4 ± 0.5	6.0 ± 0.2	**6.0 ± 0.3**	5.0
	20	6.6 ± 0.6	9.7 ± 1.0	**6.1 ± 0.4**	6.2 ± 0.4	1.0
	30	9.1 ± 0.9	18.3 ± 1.5	7.9 ± 1.0	**7.5 ± 1.2**	0.5
Parkinsons	0	10.5 ± 4.0	**7.1 ± 3.3***	10.2 ± 3.4	9.3 ± 3.6	5.0
	10	13.8 ± 4.2	12.9 ± 4.0	**12.2 ± 3.7**	12.7 ± 4.2	1.5
	20	18.6 ± 5.7	20.1 ± 5.9	**16.2 ± 4.8**	17.0 ± 4.9	0.75
	30	24.1 ± 5.5	28.0 ± 6.0	23.0 ± 5.8	**22.6 ± 6.1**	0.5
Pima	0	23.9 ± 1.9	25.9 ± 1.9	**23.2 ± 2.0**	23.4 ± 1.8	0.5
	10	25.9 ± 2.3	28.9 ± 2.3	24.7 ± 2.2	**24.1 ± 2.4**	0.25
	20	28.4 ± 2.6	32.9 ± 3.3	26.7 ± 2.5	**25.3 ± 2.5***	0.25
	30	32.6 ± 3.0	38.4 ± 3.3	31.5 ± 2.8	**29.8 ± 3.7***	0.5
Ringnorm	0	8.9 ± 1.8	**4.3 ± 0.6***	6.0 ± 1.1	4.4 ± 0.8	20.0
	10	9.6 ± 1.7	7.6 ± 1.0	6.7 ± 1.2	**6.1 ± 1.4***	10.0
	20	10.8 ± 1.8	13.0 ± 1.7	**7.9 ± 1.4***	8.4 ± 1.8	1.25
	30	15.6 ± 2.9	22.2 ± 2.3	**12.4 ± 2.4**	12.5 ± 3.0	0.75
Sonar	0	22.4 ± 5.1	**15.1 ± 4.6**	17.9 ± 4.8	15.5 ± 4.5	10.0
	10	23.3 ± 5.5	20.7 ± 5.1	20.6 ± 5.4	**19.7 ± 4.6**	5.0
	20	26.9 ± 5.7	26.3 ± 5.3	**24.2 ± 5.7**	24.5 ± 5.6	1.25
	30	32.2 ± 5.0	34.6 ± 5.5	**30.4 ± 5.4**	30.4 ± 5.3	0.75
Spambase	0	5.9 ± 0.5	4.3 ± 0.4	5.0 ± 0.5	**4.1 ± 0.5**	20.0
	10	8.2 ± 0.8	**6.1 ± 0.6**	6.5 ± 0.6	6.3 ± 0.6	0.75
	20	11.5 ± 1.0	7.5 ± 0.7	9.4 ± 0.8	**7.1 ± 0.7**	0.25
	30	16.5 ± 1.1	10.3 ± 1.1	14.3 ± 1.0	**8.6 ± 0.8***	0.25

Table 6
Test error rates for bagging (unpruned CART trees), boosting (pruned CART trees), random forest, and vote-boosting (random trees).

Dataset	Noise (in %)	Bagging	AdaBoost	Random forest	Vote-boosting	a = b (median)
Threenorm	0	19.0 ± 1.7	16.8 ± 0.8	16.6 ± 0.9	**16.2 ± 1.0***	5.0
	10	20.6 ± 1.7	20.2 ± 1.2	**18.5 ± 1.1**	18.6 ± 1.2	1.5
	20	23.3 ± 1.8	24.9 ± 1.6	**21.2 ± 1.3***	21.6 ± 1.5	1.25
	30	28.8 ± 2.1	32.0 ± 1.9	**26.9 ± 2.0***	27.2 ± 2.5	0.62
Tic-tac-toe	0	2.1 ± 0.9	**0.7 ± 0.6***	2.1 ± 1.0	1.1 ± 0.7	10.0
	10	5.7 ± 1.6	9.9 ± 1.8	5.9 ± 1.6	**5.6 ± 1.6**	2.5
	20	**12.5 ± 2.2**	20.6 ± 2.7	12.7 ± 2.6	13.0 ± 2.6	1.25
	30	23.5 ± 2.8	30.3 ± 2.9	**22.2 ± 2.7**	22.9 ± 2.9	0.75
Twonorm	0	6.4 ± 1.5	4.0 ± 0.5	3.9 ± 0.5	**3.6 ± 0.5***	5.0
	10	7.3 ± 1.6	7.1 ± 0.9	**4.8 ± 0.7***	4.9 ± 0.8	1.25
	20	9.3 ± 2.1	12.6 ± 1.6	**6.6 ± 1.1**	6.7 ± 1.2	0.75
	30	14.2 ± 2.3	21.5 ± 2.4	10.7 ± 1.7	**9.6 ± 2.5***	0.5
win/draw/loss	0	12/11/0	6/13/4	9/14/0	–	
	10	16/7/0	17/6/0	4/18/1	–	
	20	18/5/0	16/7/0	7/14/2	–	
	30	19/4/0	19/2/2	11/11/1	–	

4.3. Comparison of vote-boosting with other ensemble methods

In this section, we carry out an comparison of vote-boosting ensembles and other related methods: bagging, AdaBoost and random forest. The classification problems considered and experimental protocol followed are the same as in the previous section. In each type of ensemble, the base classifier that performs best is used: pruned CART trees in AdaBoost, unpruned CART trees in bagging, and random trees in vote-boosting and random forest. Note that unpruned CART or random trees cannot be used in combination with AdaBoost because they generally achieve zero training error, which gives rise to singularities in the weight updates. As illustrated in the previous section, similar results are obtained with vote-boosting ensembles composed of unpruned or pruned CART trees. The adabag [4], ipred [52] and ranfomForest [40] packages in R have been used for AdaBoost, bagging, and random forest imple-

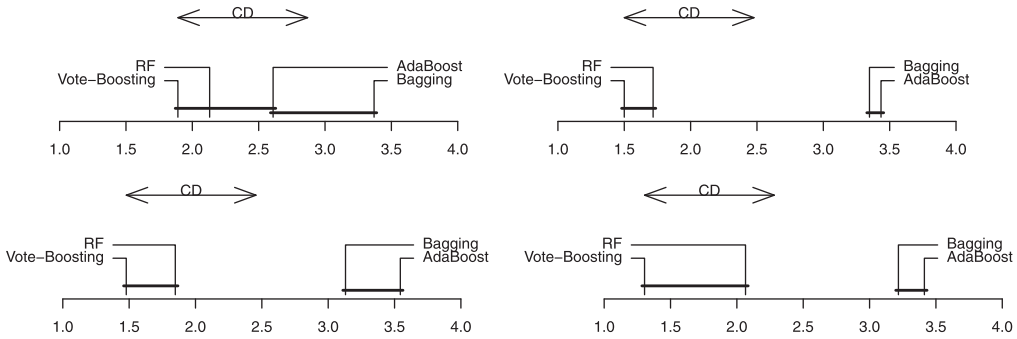


Fig. 7. Comparison of the average ranks of bagging, AdaBoost, random forest and vote-boosting using a Nemenyi test. Horizontal lines connect methods whose average ranks are not significantly different (p-value $<$ .05). The plots correspond to datasets without injected noise (top left), with 10% (top right), 20% (bottom left), and 30% class-label noise (bottom right).

Algorithm 1: Vote-boosting algorithm with resampling.

Input:

$\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}},\ \mathbf{x}_i \in \mathcal{X},\ y_i \in \{-1, 1\}$

T % Ensemble size

$\mathcal{L}$ % Base learning algorithm

$g(p)$ % Non-negative emphasis function $(0 \leq p \leq 1)$

$F_0(\cdot) \leftarrow 0$

$t_+(\mathbf{x}_i) \leftarrow 0\ \forall i = 1, \ldots, N_{train}$

$w_i^{[1]} \leftarrow \frac{1}{N_{train}}\ \forall i = 1, \ldots, N_{train}$

for $t \leftarrow 1$ ***to*** T **do**

  $f_t(\cdot) \leftarrow \mathcal{L}\left(\mathcal{D}_{train}, \mathbf{w}^{[t]}\right)$

  $t_+(\mathbf{x}_i) \leftarrow t_+(\mathbf{x}_i) + \mathbb{I}(f_t(\mathbf{x}_i) > 0) \forall i = 1, \ldots, N_{train}$

  $\pi_+^{[t]}(\mathbf{x}_i) = \frac{t_+(\mathbf{x}_i)+1}{t+2}\ \forall i = 1, \ldots, N_{train}$

  $w_i^{[t+1]} \leftarrow g\left(\pi_+^{[t]}(\mathbf{x}_i)\right)\ \forall i = 1, \ldots, N_{train}$

  Normalize $\mathbf{w}^{[t]}$

  $F_t(\cdot) \leftarrow F_{t-1}(\cdot) + f_t(\cdot)$

Output: $H_T(\cdot) = sign(F_T(\cdot))$

mentations, respectively. As in the previous subsection, ensembles of 501 classifiers are used to guarantee convergence to the asymptotic error level. In both vote-boosting and AdaBoost, weighted resampling with replacement is used to take into account the different emphasis on the training instances. In AdaBoost, reweighting was considered as an alternative. However, as reported in the literature, similar or slightly better results are obtained when resampling instead of reweighting is used [18,58].

The experiments are carried out on the original problem, and also with 10%, 20% and 30% class-label noise. Class-label noise is injected into the training set by randomly switching the class label of a random subset (10%, 20% and 30%) of the training instances. This type of label noise is known as completely at random noise (NCAR) [22]. An interesting observation is that, as the noise level increases, the values of the shape parameter that are selected tend to decrease.

The test error rates of the different ensembles and classification problems considered are displayed in Tables 4–6. The results reported are averages, followed by the corresponding standard deviations after the $\pm$ sign, over 100 realizations of the training and test set partitions. The median value for $a = b$ used in vote-boosting is reported in the last columns of Tables 4–6. In these tables, the best and second best results for each classification problem are highlighted using bold face and underlined, respectively. In addition, the lowest test error rate is marked with an asterisk (*) if the improvement over to the second best is statistically significant, at a significance level $\alpha = 0.05$. The significance of these differences is determined using a paired resampled t-test for synthetic problems, and to a corrected resampled paired t-test [8,51] when random train/test partitions are used. Finally, the number of significant wins and losses when one compares the average accuracy of vote-boosting with each of the remaining ensembles, according to the aforementioned statistical test, are shown in the last row of Table 6. Draws correspond to differences of average accuracy that are not statistically significant.

From the results presented in these tables, it is clear that vote-boosting ensembles exhibit the best overall performance. In particular, it has the largest number of statistically significant wins at all noise levels. The tally is very favorable when one compares the average accuracy of vote-boosting with bagging: vote-boosting significantly outperforms bagging in 12 out of the 23 datasets (without injected noise). For 10%, 20% and 30% noise levels, the differences in number of wins become larger: vote-boosting significantly outperforms bagging in 16, 18 and 19 out of the 23 tested datasets, respectively. The comparison with AdaBoost on the explored datasets is also favorable to vote-boosting. In the original datasets (i.e. without injected noise) vote-boosting outperforms AdaBoost in 6 out of 23 datasets and is inferior in 4 datasets: *Ozone, Parkinsons, Ringnorm*, and *Tic-tac-toe*). In this problems vote-boosting is second best; furthermore, its test error rates of vote-boosting are fairly close to AdaBoost. In these classification problems, except for *Ozone*, the shape parameter of the beta distribution used as emphasis function in vote-boosting is fairly high. This indicates a strong emphasis on uncertain examples, which has a similar effect as the emphasis on incorrectly classified instances that is characteristic of AdaBoost. On the other hand, in problems such as *Blood, Heart, Liver* or *Pima*, which are difficult for AdaBoost, vote-boosting selects low values for $a = b$, which implies that less emphasis is made on uncertain examples.

It is remarkable that, for some datasets, such as *Blood, Heart*, and *Pima*, and in most of the problems, for sufficiently high levels of class-label noise, values of $a = b$ below 1.0 provide the best accuracy. In such cases, emphasis is made on instances on which the individual ensemble classifiers agree the most. The effectiveness of this type of emphasis, which is somewhat counter-intuitive, is a consequence of the large intrinsic variability of random trees. In fact, this effect is less prominent in ensembles of more stable base learners, such as decision stumps or pruned CART trees. As the noise level increases, the performance of AdaBoost rapidly deteriorates. By contrast vote-boosting is robust to class-label noise: it outperforms AdaBoost in 17, 16 and 19 datasets for 10%, 20% and 30% injected noise, respectively.

Finally, vote-boosting is more accurate than random forest in 9 out of the 23 classification problems investigated for the noiseless case. In some cases, the improvements can be fairly large (e.g. *Blood, Chess, Ringnorm, Sonar* or *Tic-tac-toe*). In 14 datasets the two

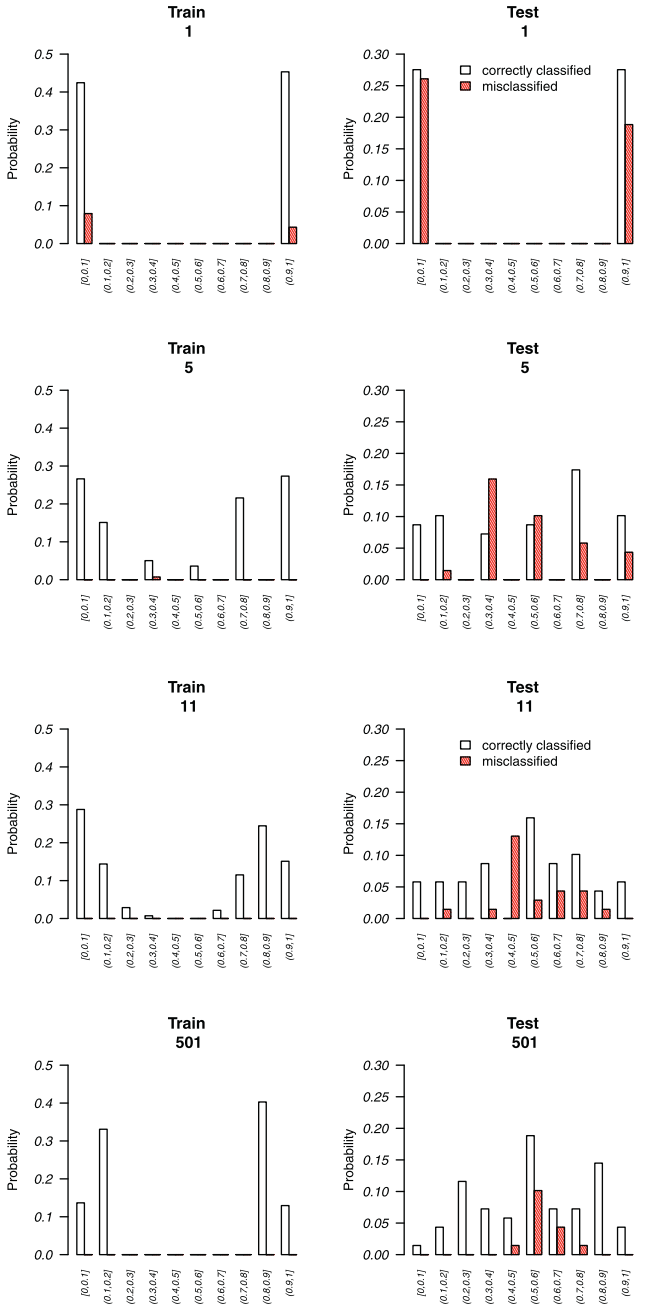


Fig. 8. Histograms of vote fractions for correctly (white) and incorrectly (red) classified instances in *Sonar* for the training set (left column) and test set (right column). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

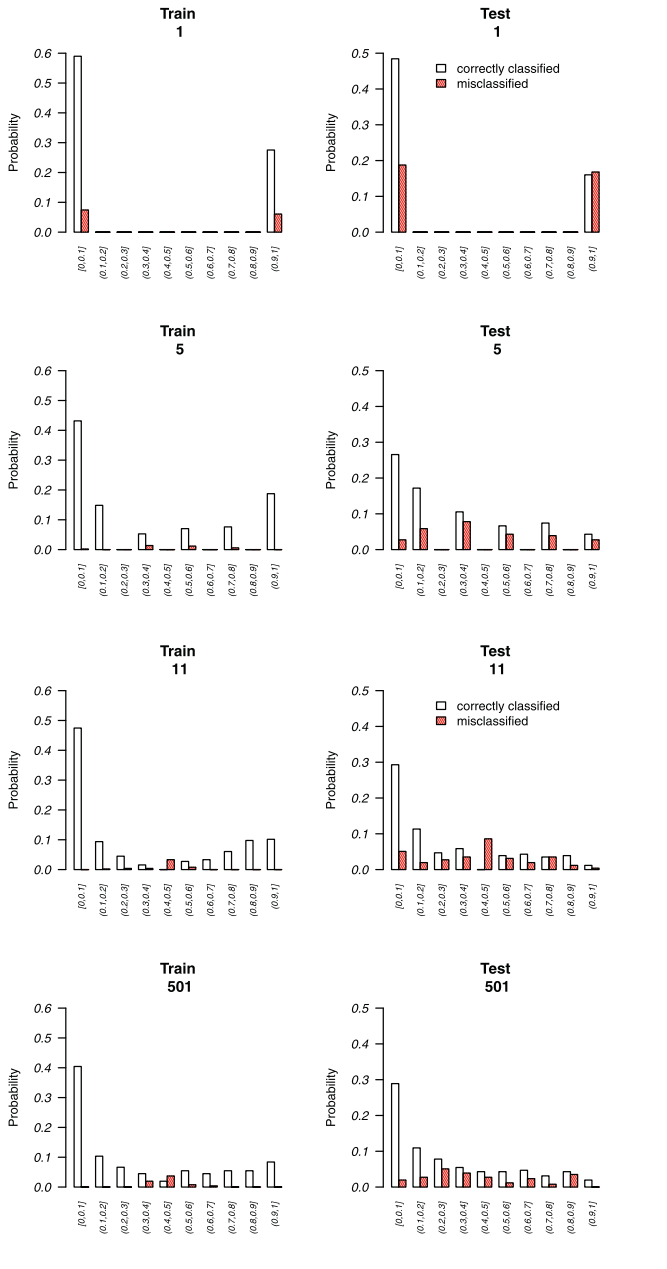


Fig. 9. Histograms of vote fractions for correctly (white) and incorrectly (red) classified instances in *Pima* for the training set (left column) and test set (right column). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

methods have comparable accuracies. The accuracy improvements of random forests over vote-boosting are not statistically significant in any of the problems investigated. When the class labels are contaminated, vote-boosting performs significantly better in 4, 7 and 11, and worse in 1, 2 and 1 datasets for 10%, 20%, and 30% noise levels, respectively. Again, in the noisy datasets we see that, when random forest wins, the differences are typically small. By contrast, when vote-boosting wins, the differences are, in general, large.

In addition, the overall performance of the accuracies of the different ensembles are summarized in Fig. 7 using the procedure described in [16]. The plots in this figure display the average ranks of bagging, AdaBoost, random forest (RF) and vote-boosting for the original datasets (top left), and for 10% (top right), 20% (bottom left), 30% (bottom right) injected class-label noise. In these plots, a horizontal solid line connects methods for which the differences of average ranks are not statistically significant using a Nemenyi test with p-value < 0.05. In the original problems, vote-boosting has the best average rank. However, the differences with random forest and AdaBoost are not statistically significant. The difference with bagging, which has the worst performance in terms average rank, is statistically significant. For problems contaminated with noise in the class labels, vote-boosting has the best average rank. The differences with respect to random forest increase for higher noise levels. However, they are not statistically significant. In all noisy problems, the average ranks of both random forest and vote-boosting ensembles are significantly better than bagging and AdaBoost.

4.4. Emphasis profiles

From the values of the shape parameter of the beta distribution reported in the last column of Tables 4–6, it is apparent that the type of emphasis and its strength need to be adapted to the classification task considered. To further investigate this point, we have carried out a detailed analysis of the distribution of class votes for *Sonar* and *Pima*, in which the optimal emphasis strategies are very different.

In *Sonar*, the vote-boosting ensemble is built using a beta distribution with $a = b = 10.0$. Thus, the optimal emphasis is to focus on training instances in which the disagreement rates are largest. In Fig. 8, the histograms of the distribution of votes are plotted for ensembles of 1, 5, 11 and 501 random trees. The height of the white bars indicate the fraction of correctly classified instances for the corresponding range of voting distributions. The red stripped bars correspond to incorrectly classified instances. The plots on the left are for the training set and on the right for the test set. In the training set, the strong focus on uncertain instances (those for which the fraction of class votes is close to 0.5) leads to a markedly bimodal distribution, in which most predictions are by clear majority. Incorrectly classified instances disappear because ensembles that are sufficiently large achieve zero training error. The distribution of class votes in the test set is markedly different: It covers the whole interval, and exhibits a low peak for intermediate class vote frequencies, especially for instances that are misclassified.

A very different picture is obtained in *Pima* (Fig. 9). In this classification task, the selected shape parameter for the symmetric beta distribution is $a = b = 0.5$. In consequence, the optimal strategy is to avoid focusing on training instances in which the disagreement rates are large. For correctly classified instances, the histograms in training and test sets are similar. Misclassified instances in the training set appear mostly around 0.5. By contrast, in the test set, they appear in the whole [0,1] interval. This is consistent with the observation that *Pima* has high levels of class-label noise.

5. Conclusions

Vote-boosting is a novel ensemble learning method in which individual classifiers are built using different weighted versions of the training data. To build a new classifier, the weights of the training instances are determined in terms of the disagreement rate among the classifiers that make up the ensemble. The optimal weighing scheme depends on the complexity of the base classifiers and on the level of noise in the class labels of the training data. For simple or regularized classifiers, such as decision stumps or pruned CART trees, vote-boosting interpolates between bagging and AdaBoost. When the level of class-label noise is small, prediction errors are more likely to occur near the classification boundary, where the uncertainty, as measured by the disagreement among ensemble predictions, is largest. Therefore, it is possible to build more accurate ensembles by focusing on uncertain instances. Since these instances are more likely to be misclassified, the emphasis given by vote-boosting is similar to AdaBoost's. For noisy classification problems, a softer emphasis on uncertain instances is generally preferable. In this case, the most accurate predictions are obtained by means of ensembles that are fairly similar to bagging. When more variable individual learners are used (e.g. unpruned CART or random trees) a milder emphasis on uncertain instances is generally needed to achieve the best generalization performance. This is a consequence of the fact that some of the uncertainty in the predictions is due to the intrinsic variability of the base learners. For problems in which the level of class-label noise is high it is in fact advantageous to progressively focus on instances in which the ensemble classifiers agree. In practice, the optimal type of emphasis can be readily determined using cross-validation within the training data.

Note that, since AdaBoost is based on emphasizing incorrectly classified instances, it cannot be used to improve the performance of base learners whose training error is small, such as unpruned CART or random trees. By contrast, vote-boosting does not have this limitation and can be used to build boosted ensembles composed of these types of classifiers that are both accurate and robust to noise in the class labels.

Acknowledgments

The research has been supported by the *Spanish Ministry of Economy, Industry, and Competitiveness* (projects TIN2016-76406-P, TIN2013-42351-P and TIN2015-70308-REDT), and *Comunidad de Madrid*, project CASI-CAM-CM (S2013/ICE-2845).

References

[1] N. Abe, B. Zadrozny, J. Langford, Outlier detection by active learning, in: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, USA, 2006, pp. 767–772.
[2] A. Ahachad, A. Omari, A.R. Figueiras-Vidal, Neighborhood guided smoothed emphasis for real AdaBoost ensembles, Neural Process. Lett. 42 (1) (2015) 155–165.
[3] A. Ahachad, L. lvarez Prez, A.R. Figueiras-Vidal, Boosting ensembles with controlled emphasis intensity, Pattern Recognit. Lett. 88 (2017) 1–5.
[4] E. Alfaro, M. Gámez, N. García, adabag: An R package for classification with boosting and bagging, J. Stat. Softw. 54 (2) (2013) 1–35.
[5] G. Armano, E. Tamponi, Building forests of local trees, Pattern Recognit. 76 (2018) 380–390.
[6] Dua, D. and Karra Taniskidou, E., 2017. Machine Learning Repository, http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.
[7] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Mach. Learn. 36 (1) (1999) 105–139, doi:10.1023/a:1007515423169.
[8] R.R. Bouckaert, E. Frank, Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–12.
[9] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, doi:10.1023/A:1018054314350.
[10] L. Breiman, Bias, Variance, and Arcing Classifiers, Technical Report, 460, Statistics Department, University of California, 1996.
[11] L. Breiman, Arcing classifiers, Ann. Stat. 26 (1998) 801–823.
[12] L. Breiman, Randomizing outputs to increase prediction accuracy, Mach. Learn. 40 (3) (2000) 229–242, doi:10.1023/A:1007682208299.
[13] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[14] J. Cao, S. Kwong, R. Wang, A noise-detection based AdaBoost algorithm for mislabeled data, Pattern Recognit. 45 (12) (2012) 4451–4465.
[15] S. Cheamanunkul, E. Ettinger, Y. Freund, Non-convex boosting overcomes random label noise, Comput. Res. Repos. (2014). arXiv:1409.2905.
[16] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30 URL:http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.
[17] T.G. Dietterich, Ensemble methods in machine learning, Lect. Notes Comput. Sci. 1857 (2000) 1–15.
[18] 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.
[19] C. Domingo, O. Watanabe, MadaBoost: a modification of AdaBoost, in: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, in: COLT '00, 2000, pp. 180–189.
[20] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (2014) 3133–3181.
[21] A. Fernndez-Baldera, L. Baumela, Multi-class boosting with asymmetric binary weak-learners, Pattern Recognit. 47 (5) (2014) 2080–2090.
[22] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn.Syst. 25 (5) (2014) 845–869.
[23] Y. Freund, An adaptive version of the boost by majority algorithm, Mach. Learn. 43 (3) (2001) 293–318.
[24] Y. Freund, A more robust boosting algorithm (2009). URL:http://arxiv.org/abs/0905.2138.
[25] Y. Freund, R. Schapire, N. Abe, A short introduction to boosting, J. Jpn. Soc. Artif. Intell. 14 (771–780) (1999) 1612.
[26] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.

[27] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), Ann. Stat. 28 (2) (2000) 337–407, doi:10.1214/aos/1016218223.
[28] G. Fumera, F. Roli, A theoretical and experimental analysis of linear combiners for multiple classifier systems, IEEE Trans. Pattern Anal. Mach. Intell. 27 (6) (2005) 942–956, doi:10.1109/TPAMI.2005.109.
[29] Y. Gao, F. Gao, Edited AdaBoost by weighted KNN, Neurocomputing 73 (16) (2010) 3079–3088.
[30] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[31] V. Gómez-Verdejo, J. Arenas-García, A.R. Figueiras-Vidal, A dynamically adjusted mixed emphasis method for building boosting ensembles, IEEE Trans. Neural Netw. 19 (1) (2008) 3–17.
[32] V. Gómez-Verdejo, M. Ortega-Moral, J. Arenas-García, A.R. Figueiras-Vidal, Boosting by weighting critical and erroneous samples, Neurocomputing 69 (7–9) (2006) 679–685.
[33] Y. Guo, P.L. Bartlett, A.J. Smola, R.C. Williamson, J. Baxter, Norm-based Regularization of Boosting, Technical Report, Australian National University, 2002.
[34] V. Gmez-Verdejo, J. Arenas-Garca, A.R. Figueiras-Vidal, Committees of AdaBoost ensembles with modified emphasis functions, Neurocomputing 73 (7) (2010) 1289–1292.
[35] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001, doi:10.1109/34.58871.
[36] D. Hernández-Lobato, G. Martínez-Muñoz, A. Suárez, Inference on the prediction of ensembles of infinite size, Pattern Recognit. 44 (7) (2011) 1426–1434. https://doi.org/10.1016/j.patcog.2010.12.021.
[37] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844.
[38] W. Jiang, Is Regularization Unnecessary for Boosting?, Department of Statistics, Northwestern University, Morgan Kaufmann, 2001, pp. 57–64.
[39] T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEE Trans. Syst. Man Cybern. Part A 41 (3) (2011) 552–568, doi:10.1109/TSMCA.2010.2084081.
[40] A. Liaw, M. Wiener, Classification and regression by randomforest, R News 2 (3) (2002) 18–22.
[41] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Netw. 12 (1999) 1399–1404.
[42] P.M. Long, R.A. Servedio, Random classification noise defeats all convex potential boosters, Mach. Learn. 78 (3) (2010) 287–304.
[43] A. Loureiro, L. Torgo, C. Soares, Outlier detection using clustering methods: a data cleaning application, in: Proceedings of KDNet Symposium on Knowledge-based Systems for the Public Sector. Bonn, Germany, 2004.
[44] R. Maclin, D.W. Opitz, Popular ensemble methods: an empirical study, Comput. Res. Repos. (2011). arXiv:1106.0257.
[45] G. Martínez-Muñoz, A. Suárez, Switching class labels to generate classification ensembles, Pattern Recognit. 38 (10) (2005) 1483–1494.
[46] H. Masnadi-shirazi, N. Vasconcelos, On the design of loss functions for classification: theory, robustness to outliers, and SavageBoost, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Advances in Neural Information Processing Systems 21, Curran Associates, Inc., 2009, pp. 1049–1056.
[47] L. Mason, J. Baxter, P. Bartlett, M. Frean, Boosting algorithms as gradient descent, in: Advances in Neural Information Processing Systems 12, MIT Press, 2000, pp. 512–518.
[48] E. Mayhua-Lopez, V. Gomez-Verdejo, A.R. Figueiras-Vidal, Real AdaBoost with gate controlled fusion, IEEE Trans. Neural Netw. Learn. Syst. 23 (12) (2012) 2003–2009.
[49] R.A. McDonald, D.J. Hand, I.A. Eckley, An empirical comparison of three boosting algorithms on real data sets with artificial class noise, in: Multiple Classifier Systems, 4th International Workshop, MCS 2003, Guilford, UK, June 11–13, 2003, Proceedings, 2003, pp. 35–44.
[50] Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, J. Song, RBoost: label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners, IEEE Trans. Neural Netw. Learn. Syst. 27 (11) (2016) 2216–2228.
[51] C. Nadeau, Y. Bengio, Inference for the generalization error, Mach. Learn. 52 (3) (2003) 239–281, doi:10.1023/A:1024068626366.
[52] A. Peters, T. Hothorn, ipred: Improved predictors, R Package Version 0.8-8, 2009.
[53] J.R. Quinlan, Bagging, Boosting, and C4.5, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996, pp. 725–730.
[54] G. Rätsch, T. Onoda, K.R. Müller, An improvement of AdaBoost to avoid overfitting, in: Proceedings of the International Conference on Neural Information Processing, Kitakyushu, Japan, 1998, pp. 506–509.
[55] J.J. Rodríguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: a new classifier ensemble method., IEEE Trans. Pattern Anal. Mach.Intell. 28 (10) (2006) 1619–1630.
[56] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (1990) 197–227.
[57] R.E. Schapire, Y. Freund, Boosting: Foundations and Algorithms, The MIT Press, 2012.
[58] C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse, A. Napolitano, Resampling or reweighting: a comparison of boosting implementations, in: 20th International Conference on Tools with Artificial Intelligence, 2008, pp. 445–451.
[59] P.K. Shivaswamy, T. Jebara, Variance penalizing AdaBoost, in: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F.C.N. Pereira, K.Q. Weinberger (Eds.), NIPS, 2011, pp. 1908–1916.
[60] B. Sun, S. Chen, J. Wang, H. Chen, A robust multi-class AdaBoost algorithm for mislabeled noisy data, Knowl. Based Syst. 102 (2016) 87–102.
[61] Y. Sun, J. Li, W.W. Hager, Two new regularized AdaBoost algorithms, in: Proceedings of the 2004 International Conference on Machine Learning and Applications - ICMLA 2004, 16–18 December 2004, Louisville, KY, USA., 2004, pp. 41–48.
[62] Y. Sun, S. Todorovic, J. Li, Reducing the overfitting of adaBoost by controlling its data distribution skewness, Int. J. Pattern Recognit.Artif. Intell. 20 (7) (2006) 1093–1116.
[63] S. Tulyakov, S. Jaeger, D. Govindaraju, V. Doermann, 2008. Machine Learning in Document Analysis and Recognition, Springer Berlin Heidelberg.
[64] R. Younsi, A. Bagnall, Ensembles of random sphere cover classifiers, Pattern Recognit. 49 (2016) 213–225.
[65] Z. Yu, D. Wang, J. You, H.-S. Wong, S. Wu, J. Zhang, G. Han, Progressive subspace ensemble learning, Pattern Recognit. 60 (2016) 692–705.

Maryam Sabzevari received her B.Sc. (2008) degree in Computer Science and M.Sc. (2010) in Artificial Intelligence from Azad University of Mashhad, Iran. Later, she received another M.Sc. (2015) in Neuroinformatics from Universidad Autnoma de Madrid (UAM), Spain. Currently, she is conducting her Ph.D. in Computer Science in Universidad Autnoma de Madrid (Spain), where she is also a Teaching Assistant. Her current research interests include machine learning, pattern recognition, neural networks, ensemble learning and learning methods in the presence of noise.

Gonzalo Martínez-Muñoz received the university degree in physics and the M.Sc. and Ph.D. degree in computer science from the Universidad Autónoma de Madrid (UAM), Madrid, Spain, in 1995, 2001 and 2006, respectively. Currently, he is Assistant Professor of Experimental Algorithmics and Computer Science Applied to the Environment 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 specialised 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 machine learning, pattern recognition, neural networks, decision trees and ensemble learning.

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.