An Analysis of Ensemble Pruning Techniques Based on Ordered Aggregation

Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato,
and Alberto Suárez, *Member, IEEE*

G. Martínez-Muñoz, D. Hernández-Lobato and A. Suárez are with the Computer Science Department, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, 11, Madrid 28049 Spain. Emails: {gonzalo.martinez, daniel.hernandez and alberto.suarez} @uam.es.

Abstract

Several pruning strategies that can be used to reduce the size and increase the accuracy of bagging ensembles are analyzed. These heuristics select subsets of complementary classifiers that, when combined, can perform better than the whole ensemble. The pruning methods investigated are based on modifying the order of aggregation of classifiers in the ensemble. In the original bagging algorithm, the order of aggregation is left unspecified. When this order is random, the generalization error typically decreases as the number of classifiers in the ensemble increases. If an appropriate ordering for the aggregation process is devised, the generalization error reaches a minimum at intermediate numbers of classifiers. This minimum lies below the asymptotic error of bagging. Pruned ensembles are obtained by retaining a fraction of the classifiers in the ordered ensemble. The performance of these pruned ensembles is evaluated in several benchmark classification tasks under different training conditions. The results of this empirical investigation show that ordered aggregation can be used for the efficient generation of pruned ensembles that are competitive, in terms of performance and robustness of classification, with computationally more costly methods that directly select optimal or near-optimal subensembles.

Index Terms

Ensembles of classifiers, bagging, decision trees, ensemble selection, ensemble pruning, ordered aggregation.

I. INTRODUCTION

Pooling the decisions of classifiers that are complementary can improve the classification accuracy of an ensemble with respect to individual learners. Two classifiers are said to be complementary when their errors are uncorrelated. When complementary classifiers are combined in an ensemble, correct decisions are amplified by the aggregation process [1], [2]. A necessary, albeit not sufficient, condition for complementarity is that the classifiers be diverse [3].

In bagging, diversity is achieved using different bootstrap samples of the training data to construct each ensemble member [4]. Bagging does not explicitly encourage the generation of complementary classifiers. The diversity among the ensemble members has its origin in the statistical fluctuations of the random bootstrap sampling process. In general, the error of bagging becomes smaller as the number of classifiers aggregated in the ensemble increases. Eventually, the error asymptotically tends to a constant level at large ensemble sizes. This asymptotic error level is generally considered the best result that bagging can achieve. Bagging is a very robust

learning algorithm under diverse noise conditions [5], [6]. Because of the statistical origin of the error reduction produced by bagging, the amount of overfitting does not generally increase with the number of classifiers that are combined [7].

Another very effective ensemble algorithm is boosting [8], [9]. In boosting, the ensemble grows by incorporating a new classifier that is constructed using the original training data with modified weights: The weights of the examples that are incorrectly (correctly) classified by the most recent classifier in the ensemble are increased (decreased). In this way, the new base learners gradually focus on examples that are harder to classify by previous ensemble members. This sequential mechanism encourages the construction of learners that are complementary. Boosting is one of the most effective methods for constructing ensembles [5], [6], [10]. However, its performance is poor in noisy domains [5], [6], [11] and overfitting is sometimes observed when large numbers of classifiers are combined [12]. In general, bagging is considered safer and more robust than boosting [11].

Despite their remarkable performance, a major drawback of ensembles is that, generally, it is necessary to combine a large number of classifiers to ensure that the error converges to its asymptotic value. This entails large memory requirements and slow speeds of classification. These aspects can be critical in online applications [13], [14]. A possible way to alleviate these shortcomings is the selection of a fraction of the classifiers from the original ensemble. Besides the reduction in complexity, ensemble pruning has other potential benefits. In particular, a subset of complementary classifiers can perform better than the complete ensemble [13], [15]–[21].

The selection of the subset of classifiers that has the best generalization performance is a difficult problem whose solution is computationally expensive. Assuming that the generalization performance can be estimated in terms of some quantity measured on the training set, the problem of finding the optimal subensemble is a combinatorial search whose complexity grows exponentially with the size of the initial pool of classifiers: For an ensemble of T classifiers, the number of non empty subensembles is $2^T - 1$. Therefore, computing the exact solution by exhaustive search is unfeasible for typical ensemble sizes. To overcome this difficulty, it is possible to use approximate algorithms that, with high probability, select near-optimal subensembles. In particular, genetic algorithms (GA) [15], [16] and semidefinite programing (SDP) [22] have been recently proposed to address the problem of ensemble pruning. Despite the fact that the time complexity of these methods is no longer exponential in the size of the initial ensemble,

their computational costs are still rather large.

In this work we analyze the properties and performance of a family of ensemble pruning methods based on ordered aggregation [13], [17]–[21]. Since they are based on reordering, their computational costs tend to be lower than the direct selection methods. These techniques take advantage of correlations between the individual classifiers of the ensemble to identify near-optimal nested subensembles of increasing size. Starting with an optimal subensemble of size $u-1$, a near-optimal subensemble of size u is obtained by incorporating the classifier from the initial pool of classifiers generated by bagging that is expected to improve the performance of the augmented subensemble the most. Even if this assumption does not hold exactly, it seems reasonable to expect that optimal subensembles of sizes $u-1$ and u share most of their classifiers. Using this greedy algorithm, the combinatorial search problem is reduced to reordering the classifiers in the ensemble. The time complexity of this simpler problem is polynomial in the size of the ensemble.

Pruning based on ordered aggregation can be effective in parallel ensemble methods, which are composed of individual classifiers that are trained independently, when the decisions of the classifiers are combined using simple majority voting. In sequential ensembles, such as boosting, the individual classifiers are generated in an order that is specified by the learning algorithm. Modifying this ordering in the aggregation stage is not expected to be useful for the selection near-optimal subsets of classifiers. In fact, pruning does not significantly improve and sometimes slightly deteriorates the classification accuracy of boosting ensembles [13], [22], [23].

The goal of this investigation is to analyze the performance of pruned ensembles generated via ordered aggregation in terms of the characteristics of the classification problems and of alternative configurations of the learning process. In particular, we investigate the following issues: whether the ordered subensembles have a comparable performance to the optimal ones, identified with exhaustive search; what is the influence of the conditions for training, namely, the amount of labeled examples available for training and the use of an independent selection set for ordering; how large the initial pool of classifiers should be, and how the performance of the pruned ordered ensembles depends on the complexity of the base learners. We also address questions regarding the cost of the different pruning methods, the memory requirements for storage, the speed of classification and the accuracy of the selected subensembles. Finally, we asses the robustness of their classification performance in problems contaminated with noise in the class labels.

The organization of the article is as follows: Section II reviews previous work on ensemble pruning. Pruning methods based on altering the order of aggregation in bagging are described in Section III. Section IV presents an empirical analysis of the pruning methods based on ordered aggregation. Their performance is compared with two effective ensemble pruning methods (based on Genetic Algorithms, and on Semi-definite Programming) and with Adaboost. A summary of the results and conclusions of this investigation is given in V.

II. Related Work

In a real-world application, it is difficult to justify the use of a classification system that requires more storage than the data from which it is induced, especially when a simple technique, such as nearest-neighbors, can also perform sufficiently well. This observation spurred Margineantu & Dietterich to investigate if, in fact, all the classifiers generated in a boosting ensemble are essential for its performance [13]. Using pruning heuristics based on measures of diversity and classification accuracy they empirically show that, in the classification tasks investigated, the number of classifiers of a boosting ensemble can be substantially reduced (up to 60-80% in some classification problems) without a significant deterioration of the generalization accuracy of the ensemble. Similar pruning strategies are investigated in [19], [24]. In [19], the initial complete ensemble is *thinned* (using the term proposed by the authors) by a sequential backward selection process that attempts to maximize the accuracy of the ensemble by eliminating classifiers whose contribution to the generalization performance (estimated in terms of measures of accuracy and/or diversity on the training set) is either detrimental or small.

In the investigations carried out by Zhou *et al.* [15], [16], genetic algorithms (GA) are applied to the selection of optimal subensembles. In [15], a GA is used to evolve the voting weights of the individual predictors in an ensemble of neural networks. Once the final chromosome is obtained, only the classifiers whose weight is above the average of the weights in the selected chromosome are included in the pruned ensemble. A similar procedure is applied to ensembles of decision trees using binary chromosomes that directly encode which classifiers should be aggregated in the final ensemble [16]. This approach has been applied to boosting in [23].

In [22], ensemble pruning is formulated as a quadratic integer programming problem defined in terms of a matrix $\mathbf{G}$, whose element G_{ij} represents the number of common errors between classifier i and classifier j. The diagonal term G_{ii} is the number of errors made by classifier i.

Normalization is applied so that the elements on the matrix are on the same scale

$$\tilde{G}_{ii} = \frac{G_{ii}}{N}, \quad \tilde{G}_{ij,i\neq j} = \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ji}}{G_{jj}}\right), \tag{1}$$

where N is the number of training instances. Intuitively $\sum_i \tilde{G}_{ii}$ measures the overall strength of the ensemble classifiers and $\sum_{ij,i\neq j} \tilde{G}_{ij}$ measures their diversity. The subensemble selection problem of size k can now be formulated as a quadratic integer programming problem

$$\underset{\mathbf{z}}{arg\,min}\ \mathbf{z^T} \cdot \tilde{\mathbf{G}} \cdot \mathbf{z}, \quad s.t. \sum_i z_i = k, \quad z_i \in \{0,1\}. \tag{2}$$

The binary variable z_i indicates whether classifier i should be selected. The size of the pruned ensemble, k, needs to be specified beforehand. This problem is a standard binary optimization problem, which is NP-hard. However, its solution can be approximated in polynomial time by applying semi-definite programming (SDP) to a convex relaxation of the original problem [22].

In [25], [26], Tsoumakas et al. devise a pruning method called *selective fusion* that combines the outputs of a subset of classifiers selected from a heterogeneous ensemble using weighted voting. The selection of the optimal subsensemble is approached as a multiple comparisons problem, which is solved by applying statistical tests to detect significant differences in cross-validation estimates of the prediction errors. In a separate work, these authors propose to use reinforcement learning to identify optimal subensembles [27]. More recently, Meynet and Thiran have proposed an information theoretic measure of ensemble performance that can be used for the selection of a subset from an initial pool of classifiers [28].

Ensemble pruning has also been studied in cost-sensitive applications, where the benefits of correct classification and the penalties for errors are example-dependent [29]. A greedy approach based on the total benefit only (i.e. ignoring the possible effects of diversity) allowed pruning rates of up to 90% without deterioration of performance. Furthermore, the authors note that, for a given instance, the combined output of the classifiers in the ensemble often converges to its final value before all ensemble classifiers are queried. Based on this observation, a dynamic instance-based pruning method is proposed: the combination process is stopped when the probability that the current prediction will be modified by the contribution of the remaining classifiers falls below a specified (small) value. Subsets of different sizes are used for classifying different examples.

There are other instance-based (dynamic) techniques for classifier selection based on measures of local accuracy [30]–[33]. The main goal of these methods is to improve the overall performance of the ensemble by selecting and/or giving more weight to the classifiers that perform best

in instances that are similar to the one that is being classified. In general, these methods do not reduce the storage and computational needs for classification. Typically, all ensemble members need to be retained in memory. Furthermore, there is a computational overhead in selecting for each instance the classifiers involved in the final prediction.

An alternative to selecting a subset of classifiers consists in replacing the ensemble by a single surrogate classifier that reproduces the decisions of the original ensemble. *Combined Multiple Models* [34] builds a single classifier using a training set that, besides the original training examples, includes synthetic examples labeled by the ensemble.

A technique that is intermediate between the selection of classifiers and the substitution of the ensemble by a single classifier is described in [14]. First, a new classification task is defined. This task consists in predicting the class label assigned by the ensemble as a whole to each of the examples in the original training set, using the outputs of the individual classifiers as predictor variables. Using this auxiliary training data a CART tree is fully grown and then pruned using cost-complexity pruning [35]. Finally, a selected subensemble is obtained from the original ensemble by eliminating those classifiers whose outputs are not used in the decisions of the CART tree.

In [36], the cost in time of classifying new instances is considered in the selection process. Ensembles are pruned by maximizing a utility function that explicitly takes into account both speed of classification and accuracy. Ref. [37] analyzes different methods for selecting subsets of classifiers from an ensemble in terms of accuracy and diversity.

A different method for ensemble pruning is to replace the ensemble by a set of classifiers using clustering. The objective is to group classifiers by similarity and to retain one representative classifier per cluster [38], [39].

Most of these studies focus on small ensembles. A notable exception is [18], where an extensive library of approximately 2000 models is compiled by inducing classifiers of different types (support vector machines, artificial neural nets, nearest-neighbors, decision trees, bagged decision trees, boosted decision trees, and boosted stumps), and with different parameter settings from the same training data. A subset of models is selected from this library according to different performance metrics: Accuracy, cross entropy, mean precision, ROC area, etc.

III. PRUNING IN ORDERED BAGGING ENSEMBLES

In this section, we introduce a family of pruning methods based on modifying the order in which classifiers are aggregated in a bagging ensemble. The order of aggregation in the original bagging algorithm is unspecified. In practice, classifiers are aggregated in the same random order as they are generated from the different bootstrap samples of the original training data. In randomly ordered ensembles, the classification error generally exhibits a monotonic decrease as a function of the number of elements in the emsemble (upper curve in Fig. 1). For large ensemble sizes these curves approach an asymptotic constant error level, which is usually considered the best result that bagging can achieve. The number of classifiers that should be included in the ensemble can be determined using heuristic rules [40].

Before describing the pruning methods in detail, it is useful to introduce some notation. The input of the learning algorithm consists in a set of labeled instances $\mathcal{Z}_{train} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, N_{train}\}$. Each instance is characterized by a feature vector (also known as the vector of attributes or predictor variables), $\mathbf{x}_i$, and a class label $y_i \in \{1, 2, \ldots, l\}$. The objective of the learning algorithm is to induce from the training data $\mathcal{Z}_{train}$, a hypothesis $h(\mathbf{x})$ that predicts the class label of a new example characterized by the vector of attributes $\mathbf{x}$.

Ensemble methods generate a variety of hypothesis (training phase) that are pooled to produce a final prediction by either weighted or unweighted voting, stacking, or some other combination methodology (classification phase). The result of combining the predictions of the classifiers in

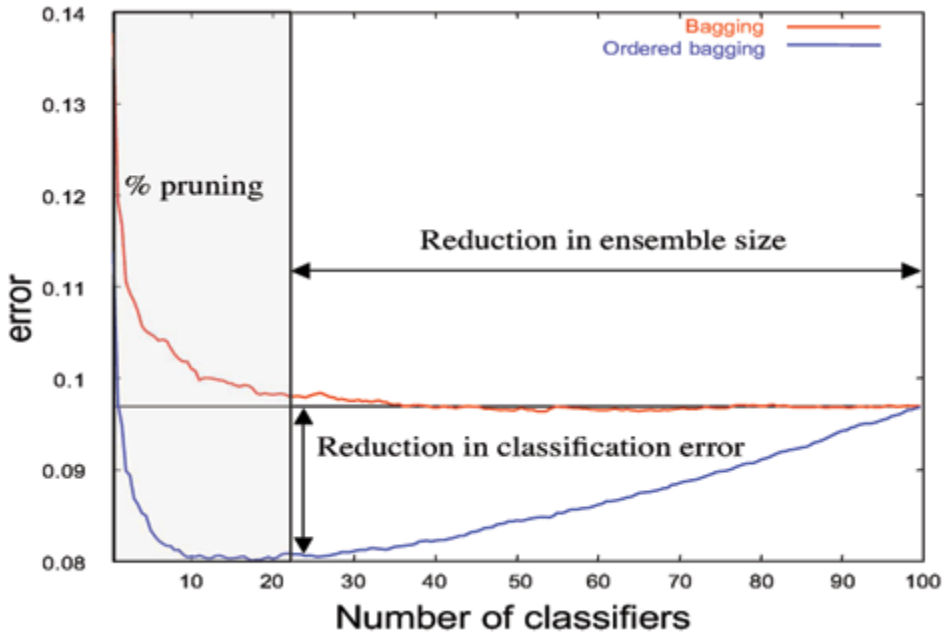

Fig. 1. Error curves that trace the dependence of the classification error as a function of ensemble size for bagging (upper curve) and ordered bagging (lower curve). The shape of these curves is similar for most of the classification problems investigated.

ensemble $E_T \equiv \{h_t(\mathbf{x})\}_{t=1}^{T}$ using equally-weighted voting is

$$H_{E_T}(\mathbf{x}) = arg\,\underset{y}{max} \sum_{t=1}^{T} \mathbb{I}\left(h_t(\mathbf{x}) = y\right), y \in \mathcal{Y}. \tag{3}$$

where $\mathbb{I}(\cdot)$ is an indicator function ($\mathbb{I}(true) = 1$ and $\mathbb{I}(false) = 0$), $h_t(\mathbf{x})$ is the class label predicted by the tth member of the ensemble, and $\mathcal{Y} = \{1, 2, \ldots, l\}$ is the set of possible class labels. In this work, ties are resolved by discarding the votes of the last classifiers included in the ensemble, one at a time, until the tie is broken. The ordering rules make use of a selection set composed of N_{sel} labeled examples $\mathcal{Z}_{sel} = \{(\mathbf{x}_i, y_i),\ i = 1, \ldots, N_{sel}\}$ to guide the order of aggregation. In general, the training set is also used for selection (i.e. $\mathcal{Z}_{sel} = \mathcal{Z}_{train}$).

The pruning strategies proposed are based on modifying the order of aggregation in the bagging ensemble: classifiers that are expected to perform better when combined are aggregated first. From the subensemble S_{u-1} of size $u-1$, the subensemble S_u of size u is constructed by incorporating a single classifier selected from the set $E_T \backslash S_{u-1}$, which contains the classifiers from the original ensemble not included in S_{u-1}. This classifier is identified using a rule that attempts to optimize the performance of the augmented ensemble S_u. The original random order of the pool of classifiers $t = 1, 2, \ldots, T$ is replaced by an ordered sequence $s_1, s_2, \ldots, s_T$, where s_j is the original label (in the randomly ordered bagging ensemble) of the classifier that occupies the jth position in the newly ordered ensemble. The curves that trace the evolution of the error as a function of the number of classifiers included in the ordered ensemble generally exhibit a minimum at intermediate ensemble sizes (lower curve in Fig. 1). This minimum corresponds to subensembles whose misclassification rates are below the error of the complete bagging ensemble. In this manner, approximate solutions to the problem of identifying near-optimal subensembles can be obtained in polynomial time. Finally, depending on the desired amount of pruning, the first τ classifiers in the sequence are selected. If the goal is to improve accuracy, τ should correspond to the minimum in the test set. Determining the location of the minimum in the test error curve using information only from the training set is a difficult task since test and train minima can occur at different subensemble sizes. Nonetheless, the minimum observed in the ensemble test error curves is fairly broad, which means that it is easy to improve the results of bagging by early stopping in the aggregation process in the ordered bagging ensembles. A heuristic that performs well in ensembles of standard (pruned) CART trees consists in stopping the aggregation process after $20 - 40\%$ of the classifiers in the ordered ensemble have been

incorporated [17], [21].

It is known in the literature that neither the accuracy of the base classifiers nor diversity are *by themselves* sufficient to identify effective ensembles [3], [41]. Successful ensemble creation methods need to take into account both accuracy and diversity [28], [42]. Preliminary experiments using ordered aggregation [17] confirm that the properties of individual classifiers are not useful to identify subensembles with good generalization performance, and that it is necessary to take into account the complementarity of the classifiers. In particular, an individual classifier may have a poor performance but its contribution can be important when combined with other classifiers in the ensemble, provided that it correctly classifies examples in which the rest of the ensemble errs. Some rules that can be used to guide the aggregation ordering are: Reduce-Error pruning [13], Kappa Pruning [13], Complementarity Measure [17], Margin Distance Minimization [17], Orientation Ordering [20] and Boosting-Based pruning [21].

1) Reduce-Error pruning: This method was proposed by Margineantu and Dietterich in [13]. The first classifier incorporated to the ensemble is the one with the lowest classification error, as estimated on the selection set $\mathcal{Z}_{sel}$. The remaining classifiers are then sequentially incorporated in the ensemble, one at a time, in such a way that the classification error of the partial subensemble, estimated on the selection set, is as low as possible. In particular, the subensemble S_u, is constructed by incorporating to S_{u-1} the classifier

$$s_u = \underset{k}{arg\,max} \sum_{(\mathbf{x},y)\in\mathcal{Z}_{sel}} \mathbb{I}\left(H_{S_{u-1}\cup h_k}(\mathbf{x}) = y \right), \quad k \in E_T \backslash S_{u-1}, \tag{4}$$

where the index k runs over the classifiers that have not already been selected up to that iteration.

In the original algorithm proposed in [13], backfitting is applied after each step. Backfitting sequentially attempts to replace one of the selected classifiers by another classifier not yet included in the ensemble. A replacement is made if a classifier that reduces the subensemble error is found in the pool of unselected classifiers. If one or more classifiers are substituted then backfitting is applied repeatedly with a limit of 100 iterations. In bagging ensembles, when the training set is used as the selection set, backfitting does not significantly reduce the generalization error of the selected subensembles [20]. Since it dramatically increases the execution time, backfitting is not used in the present investigation. Pruning using the reduce-error heuristic has been applied, with small modifications, by other authors [17]–[19].

2) Kappa pruning: This method attempts to select the subset of most diverse classifiers [13]. The amount of diversity is measured by the κ statistic. This quantity is computed on $\mathcal{Z}_{sel}$ as the fraction of examples from that selection set where the two classifiers agree, normalized to account for the measure of agreement expected by chance [6], [13]. The value $\kappa = 0$ corresponds to classifiers whose agreement equals that expected by chance. The value $\kappa = 1$ (minimum diversity) corresponds to classifiers that agree on every example in $\mathcal{Z}_{sel}$. The classifiers of the ensemble are then ordered by iteratively selecting pairs of classifiers with the highest diversity between them. Finally, the first classifiers are aggregated in a subensemble of a prescribed size.

The generalization accuracy of the pruned subensembles obtained with this method is generally poorer than that of the complete ensemble [22]–[24]. This can be explained by the fact that this technique considers only the pairwise diversity and not the diversity of the unselected classifiers with respect to the selected subensemble as a whole. Here, we propose an improved version of kappa pruning that takes this observation into account: The process initially selects the pair of classifiers that has the lowest value of κ. At step u the algorithm incorporates the classifier with the highest diversity (lowest value of the κ statistic, estimated on the selection set) with respect to the selected subensemble of size $u-1$

$$s_u = \underset{k}{arg\,min}\ \kappa_{\mathcal{Z}_{sel}}\big(h_k, H_{S_{u-1}}\big), \quad k \in E_T \backslash S_{u-1}. \tag{5}$$

In [19], a similar algorithm is proposed. It proceeds from the complete ensemble by iteratively removing the classifier that produces a thinned ensemble with the largest average diversity, as measured by the average value of the κ statistic.

3) Complementarity Measure: This method attempts to incorporate at each iteration the classifier whose performance is most complementary to that of the selected subensemble [17]. As in reduce-error, the first classifier incorporated is the one with the lowest error on the selection set $\mathcal{Z}_{sel}$. Subensemble S_u is obtained from S_{u-1} by incorporating the classifier that has the highest classification accuracy in the set of examples that are misclasified by S_{u-1}

$$s_u \;=\; \underset{k}{arg\,max} \sum_{(\mathbf{x},y)\in\mathcal{Z}_{sel}} \mathbb{I}\Big(y = h_k(\mathbf{x})\ and\ H_{S_{u-1}}(\mathbf{x}) \neq y\Big), \quad k \in E_T \backslash S_{u-1}. \tag{6}$$

The quantity maximized can be thought as the amount by which the classifier considered shifts the decision of the ensemble toward the correct classification. A similar method is proposed

in [19] under the name *concurrency thinning*. In that work, it is implemented using sequential backward selection.

4) Margin Distance Minimization: This method is introduced in [17]. Given a labeled selection set $\mathcal{Z}_{sel}$ of size N_{sel}, the signature vector $\mathbf{c}^{(t)}$ of classifier t is defined as the $N_{sel}-$dimensional vector whose *i*th component is

$$\mathbf{c}_i^{(t)} = 2\,\mathbb{I}\left(h_t(\mathbf{x}_i) = y_i\right) - 1, \quad (\mathbf{x}_i, y_i) \in \mathcal{Z}_{sel}. \tag{7}$$

The quantity $\mathbf{c}_i^{(t)}$ is equal to 1 if the tth classifier correctly classifies the *i*th example in Z_{sel}, and -1 otherwise. The ensemble signature vector, $\mathbf{c}_{ens}$, is defined as the sum of the signature vectors of the classifiers in the ensemble. The *average* ensemble signature vector is

$$\langle \mathbf{c} \rangle = T^{-1} \sum_{t=1}^{T} \mathbf{c}^{(t)} \,. \tag{8}$$

In a binary classification problem, the *i*th component of $\langle \mathbf{c} \rangle$ is the margin of the *i*th example, defined as the difference between the correct and incorrect votes that this example receives, normalized in the interval $[-1, 1]$ [43]. In problems with more than two classes, this quantity is equal to $(1 - 2edge(i))$, where $edge(i)$ is the difference between the votes corresponding to the correct class and the votes for the other (incorrect) classes, normalized in the interval $[0, 1]$ [44].

The *i*th example is correctly classified by the ensemble if the *i*th component of $\langle \mathbf{c} \rangle$ is positive. In consequence, a subensemble whose average signature vector $\langle \mathbf{c} \rangle$ is in the first quadrant of the N_{sel}-dimensional hyperspace (i.e. all the components are positive), correctly classifies all the examples in $\mathcal{Z}_{sel}$. The objective is to select a subensemble whose average signature vector is as close as possible to a reference position placed somewhere in the first quadrant. We arbitrarily select this objective position as a point $\mathbf{o}$ with equal components, namely

$$\mathbf{o}_i = p \quad \text{with} \;\; i = 1, \ldots, N_{sel} \;\; \text{and} \;\; 0 < p < 1. \tag{9}$$

The first classifiers that are incorporated into the ensemble are those that reduce the distance from the vector $\langle \mathbf{c} \rangle$ to the objective point $\mathbf{o}$ the most. In particular, the classifier selected in the *u*th iteration is

$$s_u = \underset{k}{argmin}\; d\left(\mathbf{o}, T^{-1}\left(\mathbf{c}^{(k)} + \sum_{t=1}^{u-1} \mathbf{c}^{(t)}\right)\right), \quad k \in E_T \backslash S_{u-1}, \tag{10}$$

where $d(\mathbf{v}, \mathbf{u})$ is the usual Euclidean distance between points $\mathbf{v}$ and $\mathbf{u}$.

The constant p should be sufficiently small (e.g. $p \sim 0.075$) so that easy examples (those correctly classified by most of the base learners) quickly achieve a value close to p. Subsequently, their influence in the selection of the next classifiers becomes smaller. This allows the algorithm to progressively focus on examples that are more difficult to classify. By contrast, if a value of p close to 1 were used, there would be a similar attraction for all examples throughout the selection process, which would diminish the effectiveness of the method. In this article, we propose a small improvement of this procedure and use a moving objective point $\mathbf{o}$ in (9), by allowing $p(u)$ to vary with the size of the subensemble, u. Exploratory experiments show that a value $p(u) \propto \sqrt{u}$ is appropriate.

5) Orientation Ordering: This method, proposed in [20], uses similar ideas as the previous one. Classifiers are ordered by increasing values of the angles of the signature vectors (8) with a reference vector $\mathbf{c}_{ref}$, which is defined as the projection of the diagonal of the first quadrant onto the hyperplane defined by $\langle \mathbf{c} \rangle$. The vector $\mathbf{c}_{ref}$ is selected to maximize the torque on $\langle \mathbf{c} \rangle$ (which represents the classification given by the whole ensemble) with respect to the direction that corresponds to an ideal classification (the diagonal of the first quadrant)

$$\mathbf{c}_{ref} = \mathbf{o} + \lambda \langle \mathbf{c} \rangle , \tag{11}$$

where $\mathbf{o}$ is a vector oriented along the diagonal of the first quadrant, and λ is a constant such that $\mathbf{c}_{ref}$ is perpendicular to $\langle \mathbf{c} \rangle$ ($\mathbf{c}_{ref} \perp \langle \mathbf{c} \rangle$). In the ordering phase, a stronger pull is applied along the dimensions corresponding to examples that are harder to classify by the complete ensemble. The estimation of $\mathbf{c}_{ref}$ becomes unstable when the vectors that define the projection (i.e. $\langle \mathbf{c} \rangle$ and the diagonal of the first quadrant) are close to each other. This makes the selection of $\mathbf{c}_{ref}$ less reliable and renders the ordering process less efficient. This behavior occurs when the training set is used as the selection set for ensemble construction algorithms that quickly reach zero training error, such as boosting or bagging ensembles of unpruned trees.

For an ensemble composed of T members, this ordering procedure can be carried out using *quicksort*, which has a time complexity $\mathcal{O}(Tlog(T))$. If only the first τ classifiers need to be extracted, *quickselect* can be used. This further reduces the average running time to $\mathcal{O}(T)$. This rule is the most efficient of the heuristics analyzed: The time required to perform the ordering is nearly linear in the number of elements of the original ensemble. The linear complexity of orientation ordering is related to the fact that, in contrast to the other ordering heuristics, it uses

properties of the complete ensemble as a reference in the ordering phase.

6) Boosting-based Ordering: This method, proposed in [21], consists in constructing a sequence of subensembles of increasing size from a pool of classifiers previously generated by bagging. The ensemble grows by selecting at each step the classifier that minimizes a weighted error on the selection set. The voting weights used in the computation of the error are determined following the prescription given by *Adaboost* [8]. The algorithm is very similar to boosting. However, instead of generating a hypothesis from the weighted training data at each iteration, the classifier with the lowest weighted error on $\mathcal{Z}_{sel}$ is selected from the pool of classifiers in the initial bagging ensemble. If no classifier has a weighted training error below 50%, the weights of the examples are reset to $1/N_{sel}$ and the process is continued. In contrast to regular boosting, when the selected classifier has zero error on the selection set, the aggregation proceeds until all the classifiers from the initial pool have been included in the ensemble. Note that bagging rarely generates classifiers with 0 training error and that, if this were the case, they would be selected in the first iterations of this algorithm. The final decision is performed by unweighted majority voting as recommended in [21].

IV. Empirical Analysis

This section is devoted to the analysis of the pruning methods based on ordered aggregation described in the previous section. First, the optimality of the ordering heuristics is investigated. Experiments using exhaustive search on ensembles of 25 classifiers show that the greedy ordering strategies identify subensembles that are very close to being optimal. A second group of experiments analyzes how the performance of the ordering heuristics is affected by factors such as the size of the training data, the use of an independent selection set for ordering, the size of the initial pool of the classifiers, or the complexity of the base learners. The actual performance of the pruning methods in terms of classification accuracy, training time and speed of classification is assessed in experiments on different classification tasks. These include synthetic and real-world problems from different fields of application obtained from the UCI repository [45]. Finally, the robustness of the pruned ensembles is investigated in classification problems with different levels of noise in the class labels. The pruning techniques based on ordered aggregation are compared to effective ensemble pruning techniques based on genetic algorithms [16] or semidefinite programming [22] and to Adaboost [8].

A. Optimality of the Ordering Algorithms

Ordered bagging proceeds by incorporating at each iteration the classifier that is expected to improve the performance of the ensemble the most. This greedy strategy can in fact lead to the selection of subensembles that are suboptimal. In this section, we investigate the effectiveness of these techniques by comparing the composition and the generalization performance of ordered subensembles with subsets of classifiers that exactly minimize the error in the training set. The goals of the experiments prerformed are, on the one hand, to assess how effective the greedy optimization based on ordering is, and, on the other hand, to determine whether there are significant differences in generalization performance between subensembles that are optimal with respect to the training set error and the (suboptimal) ordered subensembles.

All the experiments are performed on the *Waveform* classification problem [35] using the *reduce error* heuristic or *margin distance minimization* with $p = 0.075$. Qualitatively similar results are obtained for other classification problems investigated.

The experiments consist in constructing 100 different bagging ensembles composed of 25 standard (pruned) CART trees. In each of the 100 executions, the classifiers are aggregated either using the random order of generation in bagging, or after being ordered using the *reduce error* heuristic or *margin distance minimization* with $p = 0.075$. The original training data is used as the selection set to compute the metrics that guide the ordering process. Finally, exhaustive search is used to identify subensembles of increasing size that are optimal with respect to the classification error in the training set. In the remainder of this section, these subensembles will be referred to as "optimal subensmebles". It should be understood that their optimality is only with respect to the training set error.

Fig. 2 displays the evolution of the average training (left plot) and testing errors (right plot) of the different ensembles as a function of ensemble size. The curves for the optimal subensemble in this figure do not correspond to a nested sequence of subensembles of increasing size, as in the ordered case. In particular, the optimal subensemble of size u may not include all classifiers from the optimal subensemble of size $u - 1$. The training error curves corresponding to the optimal subensembles are a lower bound to the learning curves of subensembles obtained through ordered aggregation. This need not be the case for the error curves on the testing set. The training error curve for *Reduce-Error* is closer to the optimal subensembles than *Margin distance minimization*

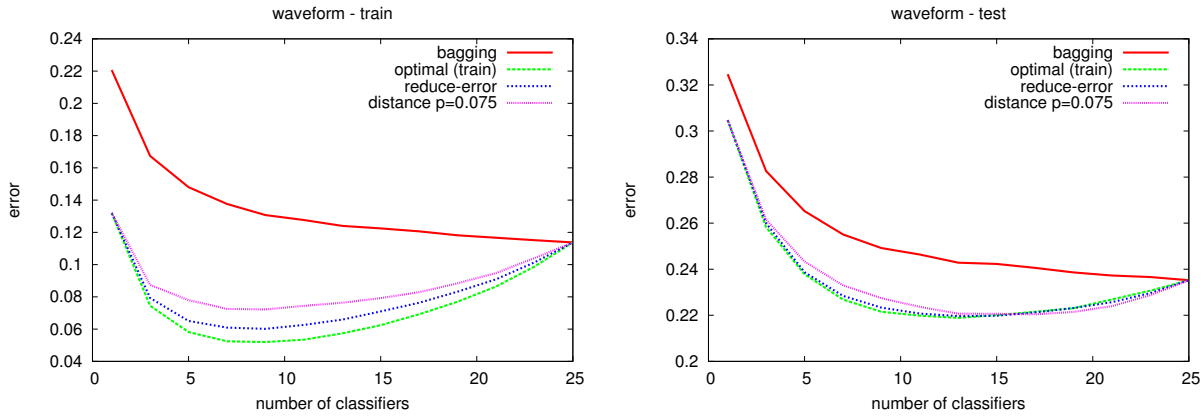

Fig. 2. Average training and test error curves for randomly ordered bagging ensembles (continuous line), subensembles that are optimal with respect to the training set error (dashed line) and ordered ensembles of increasing size using *reduce error* (dotted line) and *margin distance minimization* ($p = 0.075$) (dashed-dotted line). The curves correspond to averages over 100 executions of bagging for the *Waveform* classification problem. In each execution, a bagging ensemble composed of 25 standard (pruned) CART trees is used.

(using $p = 0.075$). This seems reasonable, since *reduce error* directly attempts to optimize the error of the subensembles in the training set. By contrast, in the test set, ordered and optimal subensembles obtain similar results.

The differences between the optimal subensembles of a given size and the corresponding ordered ensembles can be quantified using a coincidence matrix $\mathbf{O}$, whose element O_{ij} takes the value 1 if the classifier selected by the ordering heuristic in the jth position is included in the optimal subensemble of size i and 0 otherwise. If the ordered subensembles were optimal for all sizes, the matrix $\mathbf{O}$ would be a lower triangular matrix (i.e., all ones on and under the diagonal and zeros above the diagonal). Fig. 3 displays the the average occurrence matrix, $\bar{\mathbf{O}}$, for bagging ensembles composed of 25 classifiers. Averages are over the 100 different executions of bagging on the *Waveform* problem. An inverse gray-scale is used to indicate the value of the matrix elements: A black (white) cell in the position (i, j) corresponds to $\bar{O}_{ij} = 1(0)$. Gray cells correspond to $0 < \bar{O}_{ij} < 1$. The matrices are predominantly lower diagonal. There is some dispersion, mainly around the principal diagonal. This confirms the hypothesis that optimal subensembles of sizes u and $u-1$ share most of their elements and that the heuristic ordering rules can be used identify near-optimal solutions of the subensemble selection problem. Experiments

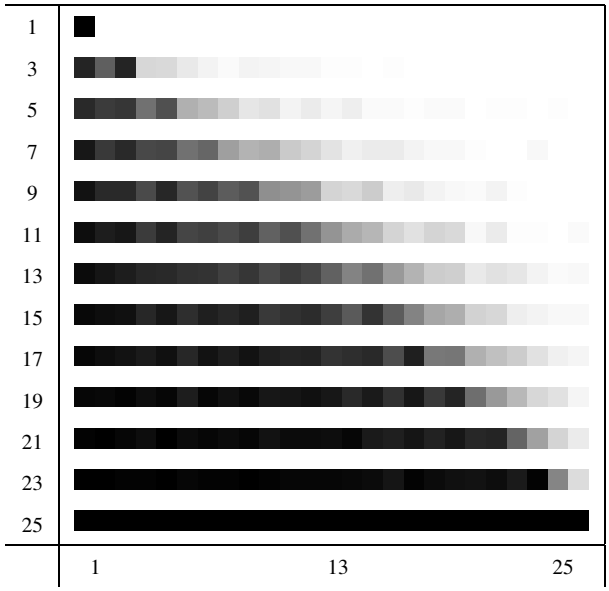

Fig. 3. Coincidence matrix, (**O**), for the optimal and ordered subensembles. Results are for bagging ensembles of 25 members, and are averaged over 100 executions of bagging on the *Waveform* problem.

the *Waveform* problem without averaging over different executions of bagging confirm these conclusions in larger ensembles (31 individual classifiers).

B. Influence of training conditions.

This series of experiments is designed to determine how the effectiveness of the ordering strategies depends on the amount of data available for training and whether, instead of the original training data, an independent selection set should be used to order the classifiers in the ensemble. For every training dataset two kinds of bagging ensembles are generated: Classifiers in ensembles of the first type are built and ordered using the same training set (all of the training data). The second kind of ensembles are generated by building classifiers with $2/3$ of the training data and using the remaining $1/3$ for ordering. We consider ensembles of 100 classifiers for the *Magic04* and for the *Waveform* problems from the UCI repository [45], trained on datasets of different sizes: 500, 1000, 2000, 5000 and 10000. The generalization error of the different subensembles is estimated on independent test sets. For the synthetic problem *Waveform* a set of fixed size (5000) is used for testing. The test set for *Magic04* includes all the examples not used in training. The results reported in Tables I and II correspond to averages over 100 realizations

(random stratified partitions into train and test set for *Magic04*; independent simulations for *Waveform*). The second column in these tables shows the results for the complete bagging ensembles. The third column displays the error rates for complete bagging ensembles that are built using only two thirds of the training data available. These rates correspond to the test errors of complete bagging ensembles that set apart one third of the original training data for ordering. The remaining columns correspond to pruned ensembles with 21% of the classifiers from the original ensemble, obtained by ordering with the reduce-error and margin distance (MDSQ) heuristics. The fourth and sixth columns (labeled *all*) display the results of ensembles that use the same data for training the classifiers and for ordering them. Columns 5 and 7, labeled *1/3 withheld*, correspond to ensembles where one third of the original training data are set apart for ordering.

These results show that, as one should expect, the performance of bagging ensembles improves with larger training sets. More interestingly, pruned ensembles outperform bagging in the cases investigated, even when large training sets are used. However, the improvements in error rate become smaller as more data is available for training. This is due to the fact that the bagging error itself becomes smaller and the margin for improvement diminishes. A second conclusion from these experiments is that using all available data for training the individual classifiers and for ordering seems to be preferable to withholding some data for ordering. In the problems investigated, the benefits of using an independent selection set in the ordering phase (namely, that it provides unbiased estimates of the generalization error curves) do not compensate the loss in accuracy arising from the fact that fewer examples are used to train each individual classifier.

TABLE I

DEPENDENCE OF GENERALIZATION ERROR ON THE TRAINING CONDITIONS FOR *Magic04*.

Training	Bagging		Redure-error (21%)		MDSQ (21%)	
set size	all	2/3	all	1/3 withheld	all	1/3 withheld
500	19.1 $\pm$ 1.4	19.8 $\pm$ 1.7	17.3 $\pm$ 0.9	18.0 $\pm$ 1.1	17.1 $\pm$ 0.9	17.7 $\pm$ 0.9
1000	17.6 $\pm$ 0.9	18.4 $\pm$ 1.3	16.2 $\pm$ 0.6	16.7 $\pm$ 0.7	16.0 $\pm$ 0.5	16.6 $\pm$ 0.7
2000	16.5 $\pm$ 0.6	17.1 $\pm$ 0.8	15.4 $\pm$ 0.4	15.8 $\pm$ 0.5	15.3 $\pm$ 0.4	15.7 $\pm$ 0.5
5000	15.5 $\pm$ 0.4	16.0 $\pm$ 0.5	14.6 $\pm$ 0.3	15.0 $\pm$ 0.4	14.6 $\pm$ 0.3	15.0 $\pm$ 0.4
10000	14.6 $\pm$ 0.3	15.0 $\pm$ 0.4	14.0 $\pm$ 0.3	14.4 $\pm$ 0.3	14.0 $\pm$ 0.3	14.4 $\pm$ 0.3

TABLE II

DEPENDENCE OF GENERALIZATION ERROR ON THE TRAINING CONDITIONS FOR *Waveform*.

Training	Bagging		Redure-error (21%)		MDSQ (21%)	
set size	all	2/3	all	1/3 withheld	all	1/3 withheld
500	21.8 ± 2.2	22.6 ± 2.4	19.8 ± 1.1	20.3 ± 1.3	19.3 ± 0.9	19.9 ± 1.2
1000	20.7 ± 1.7	21.3 ± 1.9	19.0 ± 0.9	19.4 ± 1.1	18.7 ± 0.9	19.2 ± 1.0
2000	19.6 ± 1.0	20.3 ± 1.2	18.5 ± 0.7	18.8 ± 0.8	18.2 ± 0.6	18.6 ± 0.8
5000	18.8 ± 0.8	19.2 ± 0.8	18.0 ± 0.7	18.1 ± 0.6	17.8 ± 0.7	18.0 ± 0.7
10000	18.1 ± 0.7	18.5 ± 0.7	17.5 ± 0.6	17.8 ± 0.6	17.4 ± 0.7	17.7 ± 0.6

TABLE III

AVERAGE ERROR AND AVERAGE NUMBER OF CLASSIFIERS FOR THE MINIMA IN THE TEST ERROR CURVES FOR *Waveform*.

Initial pool size	11	25	51	75	101	151	201	251	501	751	1000
Error	23.3	21.7	20.8	20.4	20.2	20.0	19.9	19.7	19.6	19.5	19.5
# of classifiers	9	17	25	29	37	49	61	65	131	189	261
Percentage (%)	81.8%	68.0%	49.0%	38.7%	36.6%	32.5%	30.3%	25.9%	26.1%	25.2%	26.1%

C. Influence of the Size of the Initial Pool of Classifiers

In this series of experiments, we investigate how ordered aggregation is affected by the size of the initial pool of classifiers. An initial bagging ensemble of 1000 standard (pruned) CART trees is built for the *Waveform* problem. The classifiers are then ordered considering only the first $11, 25, 51, 75, 101, 151, 201, 251, 501, 751$ and 1000 trees of the original ensemble. These steps are repeated for 100 different samples of training (300 instances) and testing data (5000 instances). Fig. 4 shows the average error curves in the training set (left plot) and in the test set (right plot) using reduce-error ordering. Table III displays the value of the error and the numbers of classifiers for the minima in the test error curves. The numbers in the fourth row are the percentages of classifiers from the initial pool that are included in subensembles that correspond to the minima in the test error curves for ordered bagging.

The training error curves (left plots in Fig. 4) show an initial steep decrease of the training error as a function of the number of classifiers in the ensemble. As the size of the ensemble increases, the curves diverge: The error curves of subensembles ordered from a smaller initial

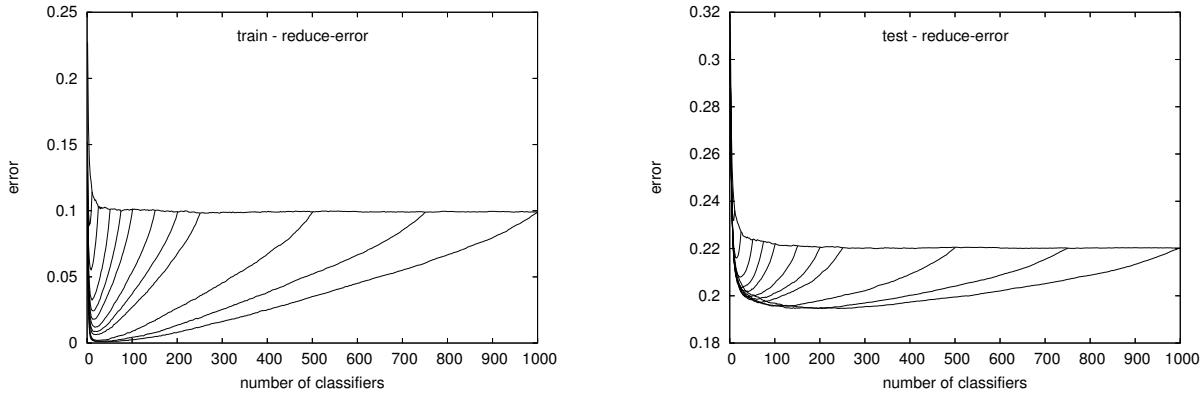

Fig. 4. Training and test errors for *Waveform* for bagging and ordered bagging using the first: 11, 25, 51, 75, 101, 151, 201, 251, 501, 751 and 1000 trees.

pool of classifiers start increasing before those of larger initial pools. The training error curves are nested and do not intersect. For a given ensemble size, the errors of ordered subensembles generated from larger initial pools are generally lower than those from smaller initial pools. This seems reasonable: The quantity used to guide the ordering procedure is estimated on the training dataset. Furthermore, the initial pools are nested; that is, all classifiers in an initial pool of a given size are also included in larger pools. For instance, the classifiers in the initial pool composed of 251 elements are also present in the pools of sizes 501, 751 and 1000.

The test error curves display a behavior similar to the training error curves, except that they intersect at several points. In any case, these curves also tend to be nested. Their minima are generally lower for curves that correspond to larger initial bagging ensembles. However, the improvements in the minimum test error become smaller as the number of classifiers in the initial pool is increased. The number of classifiers needed to achieve the minimum in test error increases with the size of the initial pool. This increase is slower than the increase in the size of the initial selection pool. Qualitatively similar curves are obtained in most of the other classification tasks and for the different ordering heuristics. Deviations from this behavior appear in the test-error curves for *Australian*, *Horse-colic* (only with boosting-based ordering) and *Votes*. These anomalies are small and do not affect the effectiveness of the pruning procedures analyzed except in *Votes*.

In summary, increasing the size of the initial pool of classifiers generally improves the

generalization performance of ordered bagging ensembles. Beyond a certain size, the benefits in error reduction are small. These improvements in performance are achieved at the expense of selecting larger subensembles. Nonetheless, if the objective is to improve the classification performance, larger initial ensembles should be generated. If the goal is improving performance and reducing ensemble size, bagging ensembles of intermediate size should be used. In the problems investigated, using bagging ensembles of 100 elements as the initial pool of classifiers provides a good balance between complexity and accuracy.

D. Dependence on the complexity of the base classifiers.

Extensive experiments have been carried out to analyze the efficiency of the ordering procedure as a function of the complexity of the base classifiers. The architectures considered are, in order of complexity: Decision stumps, CART trees pruned with the standard cross-validation procedure proposed in [35] and fully developed (unpruned) CART trees.

For simple classifiers, such as decision stumps, the minima in the training and test data appear at similar subensemble sizes [46]. Hence, the pruning rule for simple base learners is to select the ordered subensemble whose size corresponds to the minimum error in the training set. When more complex classifiers are used as base learners (e.g. CART trees), the minima in the training error curves generally correspond to smaller subensembles than those in the test error curves. For ensembles of pruned CART trees, in the problems investigated, subensembles with $20-40\%$ of the initial classifiers have the best overall classification accuracy. By contrast, reducing the size of ensembles of unpruned CART trees does not improve and sometimes even deteriorates the generalization performance of the ensemble: Since trees are grown until all instances are correctly classified, the error rate of an individual unpruned tree on the bootstrap training data is, by construction, 0%. For this reason, the metrics used in the selection process are unable to distinguish among the different classifiers.

A representative subset of results from these experiments is given in Table IV. The best generalization performance is obtained with pruned ensembles of pruned CART trees and with complete ensembles of unpruned CART trees. However, the large complexity of the latter models makes them impractical. For decision stumps, it is also possible to select subsets of classifiers that outperform the complete ensemble. Except in the *Heart* problem, the performance of pruned ensembles of decision stumps is clearly inferior to ensembles of CART trees, especially in

TABLE IV

DEPENDENCE ON THE COMPLEXITY OF THE BASE CLASSIFIERS.

	Decision Stumps			**CART trees (pruned)**			**CART trees (unpruned)**		
	Bagging	MDSQ	SDP	Bagging	MDSQ	SDP	Bagging	MDSQ	SDP
Breast	7.0±3.7	5.2±2.9	5.4±2.8	4.8±2.8	4.0±2.6	3.8±2.4	3.7±2.3	3.9±2.5	3.8±2.2
Diabetes	27.8±4.1	26.2±4.2	26.9±4.7	24.9±3.9	24.0±4.1	24.3±4.3	24.7±4.0	25.8±4.4	25.9±4.5
Heart	24.2±10.1	16.8±7.0	17.2±6.7	19.6±7.9	17.7±6.8	17.7±6.9	19.5±7.0	21.0±7.3	20.5±8.0
Waveform	39.2±5.0	28.3±4.6	29.4±6.2	23.0±2.4	19.9±1.2	20.0±1.2	19.7±1.4	20.1±1.0	20.4±1.1

classification tasks with more than 2 classes, where decision stumps are at a clear disadvantage with respect to decision trees. Nonetheless, pruned ensembles of decision stumps have very low memory requirements and can be generated very quickly. Therefore, they can be an attractive alternative when there are stringent memory requirements, or if some of the accuracy in classification can be sacrificed for speed in training and classification.

Given that the objective of this investigation is to design pruning techniques that reduce the memory requirements and increase the speed of classification of ensembles while maintaining or improving their generalization performance, only ensembles of pruned CART trees will be considered further. The exception is complete ensembles of unpruned trees, which, despite their complexity, will be used for comparisons in classification accuracy because of their good overall generalization performance.

E. Classification performance of pruned ensembles

The performance of the different pruning heuristics is evaluated in a series of experiments on 28 datasets from the UCI repository [45]. Each experiment consists in 100 executions for each dataset. For the synthetic datasets (*Led24*, *Ringnorm*, *Twonorm* and *Waveform*) random training and testing samples are generated. In the remaining problems, $10\times$ 10-fold cross-validation is used. Each execution involves the following steps:

1) Generate the training and testing sets by either 10-fold-cv or by random sampling (see Table V).
2) Build a bagging ensemble of 100 standard (pruned) CART trees [35].
3) Order the classifiers in the bagging ensemble using the heuristics described in Section III:

TABLE V

CHARACTERISTICS OF THE DATASETS AND TESTING METHOD

Dataset	Instances	Test	Attrib.	Classes	Dataset	Instances	Test	Attrib.	Classes
Audio	226	10-fold-cv	69	24	New-thyroid	215	10-fold-cv	5	3
Australian	690	10-fold-cv	14	2	Pendigits	10992	10-fold-cv	16	10
Breast W.	699	10-fold-cv	9	2	Ringnorm	300	5000 cases	20	2
Diabetes	768	10-fold-cv	8	2	Satellite	6435	10-fold-cv	36	2
Ecoli	336	10-fold-cv	7	8	Segment	2310	10-fold-cv	19	7
German	1000	10-fold-cv	20	2	Sonar	208	10-fold-cv	60	2
Glass	214	10-fold-cv	9	6	Spam	4601	10-fold-cv	57	2
Heart	270	10-fold-cv	13	2	Tic-tac-toe	958	10-fold-cv	9	2
Horse-Colic	368	10-fold-cv	21	2	Twonorm	300	5000 cases	20	2
Ionosphere	351	10-fold-cv	34	2	Vehicle	846	10-fold-cv	18	4
Labor	57	10-fold-cv	16	2	Votes	435	10-fold-cv	16	2
Led24	200	5000 cases	24	10	Vowel	990	10-fold-cv	10	11
Liver	345	10-fold-cv	6	2	Waveform	300	5000 cases	21	3
Magic04	19020	10-fold-cv	10	2	Wine	178	10-fold-cv	13	3

Reduce-Error (RE), Kappa pruning (Kappa), Complementarity Measure (CC), Margin Distance Minimization using a moving reference point $p(u) = 2\sqrt{2u}/T$ (MDSQ), orientation ordering (OO) and Boosting-based pruning (BB). The training data is also used as the selection set for ordering.

4) Select the first 21 trees of the ordered bagging ensemble for aggregation.

To compare with other pruning methods proposed in the literature, the same bagging ensembles are pruned using a Genetic Algorithm (GA) [15], [16] and a technique based on Semi-definite Programming (SDP) [22]. The results for Adaboost (AB) and complete bagging ensembles of unpruned trees (BagU) are also given for reference. The GA implemented is adapted from [16]. Candidate solutions are represented by binary string chromosomes, $\{b_i\}_{t=1}^{T}$, whose size equals the number of classifiers in the ensemble. The *t*th allele of the chromosome, b_t, is a binary variable that encodes whether the *t*th classifier is included ($b_t = 1$) or not ($b_t = 0$) in the subensemble. The fitness function used to rank individuals (subensembles) considers two factors: The accuracy of the subensemble, estimated in the selection set, and a bias term that favors the selection of

TABLE VI

TEST ERROR RATES (AVERAGE ± STANDARD DEVIATION) FOR ENSEMBLES OF STANDARD (PRUNED) CART TREES.

Dataset	**Bagging**	**RE**	**Kappa**	**CC**	**MDSQ**	**OO**	**BB**	**GA**	**SDP**	**AB**	**BagU**
Audio	26.8±8.0	**21.1±7.3**	27.7±7.7	**21.0±8.0**	**20.5±7.6**	**21.2±7.6**	**20.9±7.6**	**20.8±8.3**	**21.1±7.4**	**17.8±7.8**	**19.9±7.6**
Australian	14.5±3.8	13.7±4.0	**13.7±4.0**	13.8±4.1	**13.6±4.0**	13.6±4.0	**13.6±3.9**	14.4±3.8	**13.5±4.1**	14.0±3.9	**13.4±3.9**
Breast	4.8±2.8	**4.2±2.4**	4.9±2.9	**4.2±2.5**	**4.0±2.6**	**4.1±2.5**	**4.1±2.6**	**4.2±2.5**	**3.8±2.4**	**3.3±1.9**	**3.7±2.3**
Diabetes	24.9±3.9	24.6±4.1	24.2±3.9	**23.9±3.8**	**24.0±4.1**	**24.0±4.1**	24.2±3.9	25.2±5.1	24.3±4.3	26.7±3.9	24.7±4.0
Ecoli	17.7±5.9	**16.2±5.7**	18.4±6.0	**16.0±6.0**	**15.3±5.7**	**15.5±5.6**	**15.4±6.0**	**16.5±6.0**	**15.8±5.7**	**14.0±5.3**	**16.2±5.6**
German	25.6±3.0	**24.6±3.4**	25.9±3.0	**24.8±3.8**	**24.7±3.9**	25.0±3.5	**24.8±3.7**	25.1±3.3	**24.8±3.5**	25.0±3.3	**23.9±3.8**
Glass	29.8±7.1	**24.6±7.7**	31.5±7.9	**24.7±7.6**	**24.4±7.3**	**24.3±7.8**	**25.0±7.6**	**24.7±7.5**	**24.6±7.8**	**21.2±8.2**	**23.2±8.1**
Heart	19.6±7.9	19.2±7.3	**17.7±6.7**	18.8±6.9	**17.7±6.8**	**17.3±6.9**	18.5±7.1	19.2±7.2	**17.7±6.9**	21.1±7.8	19.5±7.0
Horse-colic	17.7±6.2	**15.7±5.6**	16.7±6.0	**16.3±5.7**	**15.7±5.5**	**14.8±5.8**	**16.3±6.4**	**15.8±5.8**	**16.6±6.0**	18.8±6.1	**15.4±5.4**
Ionosphere	9.7±4.6	**7.5±4.4**	**8.0±4.4**	**7.3±4.0**	**7.3±3.9**	**7.7±4.3**	**7.5±4.2**	**7.5±4.3**	**7.2±3.9**	**6.4±3.7**	**8.4±4.9**
Labor	13.4±12.8	11.8±12.2	**9.2±11.7**	13.9±12.7	**6.9±9.8**	**8.2±10.2**	**8.2±10.0**	**10.9±12.0**	**8.4±10.4**	13.4±15.4	**7.7±10.1**
Led24	30.8±3.0	**30.2±2.4**	31.4±3.2	**30.1±2.3**	**29.9±2.2**	**29.8±1.9**	**30.1±2.1**	**30.1±2.6**	**30.2±2.4**	**29.9±2.3**	34.1±2.1
Liver	31.0±6.3	**27.8±6.9**	29.9±6.1	**28.2±6.7**	**28.2±7.0**	**28.9±7.6**	**28.3±6.7**	**28.7±6.9**	**28.2±7.0**	**28.5±7.1**	29.2±7.8
Magic04	14.0±0.7	**13.5±0.7**	**13.8±0.7**	**13.5±0.7**	**13.5±0.7**	**13.5±0.7**	**13.6±0.7**	**13.7±0.7**	**13.5±0.7**	**12.0±0.7**	**12.1±0.7**
New-thyroid	6.9±5.3	**5.8±5.0**	6.1±5.2	**5.6±4.7**	**5.0±4.6**	**5.2±4.7**	**4.9±4.3**	**5.3±4.5**	**5.1±4.6**	5.8±4.6	**5.6±4.5**
Pendigits	2.0±0.3	**1.8±0.3**	2.3±0.3	**1.8±0.3**	**1.7±0.3**	**1.7±0.3**	**1.9±0.4**	**1.8±0.3**	**1.7±0.3**	**0.6±0.2**	**1.7±0.3**
Ringnorm	12.6±2.7	**11.5±1.8**	14.6±2.2	**11.4±1.6**	**9.8±1.5**	**10.0±1.5**	**9.8±1.4**	**10.7±1.7**	**9.7±1.5**	**5.9±0.6**	**10.2±1.8**
Satellite	11.8±1.1	**11.1±1.1**	11.9±1.1	**11.0±1.1**	**11.0±1.0**	**11.1±1.1**	**11.1±1.1**	**11.3±1.0**	**11.0±1.1**	**7.6±1.0**	**8.8±1.0**
Segment	3.0±1.1	**2.6±1.0**	3.9±1.2	**2.6±0.9**	**2.6±1.0**	**2.5±1.0**	**2.6±1.0**	**2.6±1.0**	**2.5±1.0**	**1.5±0.7**	**2.3±0.9**
Sonar	25.1±9.5	**20.8±9.1**	24.3±9.0	**21.8±9.0**	**19.8±9.8**	**21.2±8.8**	**20.0±10.5**	**21.7±9.7**	**19.5±9.6**	**13.3±8.0**	**20.1±8.3**
Spam	7.2±1.4	**6.3±1.6**	**6.9±1.7**	**6.4±1.5**	**6.3±1.5**	**6.4±1.6**	**6.4±1.5**	**6.6±1.5**	**6.3±1.6**	**4.6±1.1**	**5.3±1.3**
Tic-tac-toe	1.5±1.3	1.6±1.4	3.3±1.9	1.6±1.3	1.4±1.2	1.4±1.1	1.5±1.2	1.5±1.3	**1.2±1.1**	**0.8±1.1**	**1.3±1.1**
Twonorm	9.6±3.1	10.0±1.8	12.9±2.4	10.2±1.9	**8.0±1.1**	**7.9±1.1**	**8.2±1.2**	**8.9±2.1**	**8.0±1.1**	**4.8±2.3**	**6.9±1.5**
Vehicle	28.5±3.5	**25.6±4.1**	28.6±3.9	**25.7±4.0**	**25.2±4.2**	**25.2±4.4**	**25.4±3.8**	**25.9±4.1**	**25.3±4.2**	**22.7±3.6**	**24.6±3.6**
Votes	4.4±3.0	4.7±3.1	4.8±3.2	4.7±3.2	4.8±3.2	5.3±3.4	4.7±3.2	4.4±3.0	4.8±3.2	4.6±3.0	4.4±3.1
Vowel	9.2±2.9	9.7±3.2	12.1±3.3	9.8±3.1	9.1±3.1	8.6±2.9	8.7±2.9	9.2±2.8	8.7±2.9	**3.5±2.4**	**8.6±2.8**
Waveform	23.0±2.4	**20.6±1.3**	**22.2±1.7**	**20.4±1.2**	**19.9±1.2**	**20.3±1.3**	**20.3±1.4**	**20.5±1.4**	**20.0±1.2**	**18.0±0.8**	**19.7±1.4**
Wine	4.4±5.0	4.2±5.1	6.7±6.3	4.4±5.1	**3.1±4.0**	4.0±4.7	**3.3±4.2**	3.6±4.7	**3.2±4.2**	4.1±4.8	**3.0±4.0**

large subensembles. This small bias term is introduced to avoid the selection of subensembles that are too small, which, generally, do not have good generalization properties. Its value is always smaller than $1/N_{sel}$ so that (i) a higher fitness is assigned to chromosomes with lower error in the selection set, independently of the number of selected classifiers, and (ii) if two chromosomes (subensembles) have the same accuracy on the selection set, a higher fitness is assigned to the chromosome that encodes the larger subensemble. The configuration of the GA is adjusted in exploratory experiments using the recommendations given in [47]. The size of population evolved is 200. The chromosomes are initialized diagonally, so that all the different

subensembles of size 1 are included in the initial population: The alleles of the *i*th individual are initialized to zero except for the *i*th allele, which is set to 1. A simple bit-flip operator is used for mutation. It is applied with probability $p_m = 0.005$. To avoid positional bias, a uniform crossover operator is applied with probability $p_c = 0.65$. Elitism is used: The best two individuals from the population are included in the next generation. The population is evolved during 200 epochs. In contrast to the other methods investigated, where the pruning rate can be adjusted to obtain near-optimal subensembles of different sizes, GA-pruning produces a single solution.

SDP pruning is executed setting the number of classifiers in the pruned ensemble to $k = 21$ in (2). This value is chosen because (i) it coincides with the size of the subensembles selected by the ordering heuristics, and (ii) pruned ensembles using SDP with $k = 21$ exhibit a good overall performance in the classification problems investigated. Exploratory experiments show that the overall performance of the ensembles pruned using SDP is very similar for a large range values of k in the vicinity of 21.

Table VI shows the average values of the test error for the different datasets and pruning methods. Values that are significantly better than bagging (using a paired t-test with $p-value < 0.01$) are highlighted in boldface. Results where complete bagging ensembles perform significantly better than pruned ensembles (with p-value < 0.01) are underlined. One should be cautious with the interpretation of the confidence levels in the real-world datasets, where cross-validation is used: Their significance may be overestimated in the statistical test, as illustrated in [48]. For the synthetic datasets, given that the experiments are performed using independent samples, the confidence levels are unbiased. Adaboost achieves the lowest classification error in the majority of the classification problems investigated. Among pruned bagging ensembles, the best results correspond to Margin distance minimization (MDSQ) and SDP, followed by orientation ordering (OO) and boosting-based pruning (BB). The overall performance of these pruned ensembles is comparable to the more complex complete bagging ensembles of unpruned trees (BagU). For almost all datasets, most pruning techniques improve the generalization performance of bagging ensembles of standard (pruned) CART trees independently of the number of examples, attributes or classes of the datasets. An exception to this general behavior is *Votes*. In this problem, only GA-pruning achieves an error comparable to bagging. However, the ensemble selected by GA retains most (88%) of the original trees. Kappa pruning, which is based only on diversity, has the poorest overall performance on the datasets investigated.

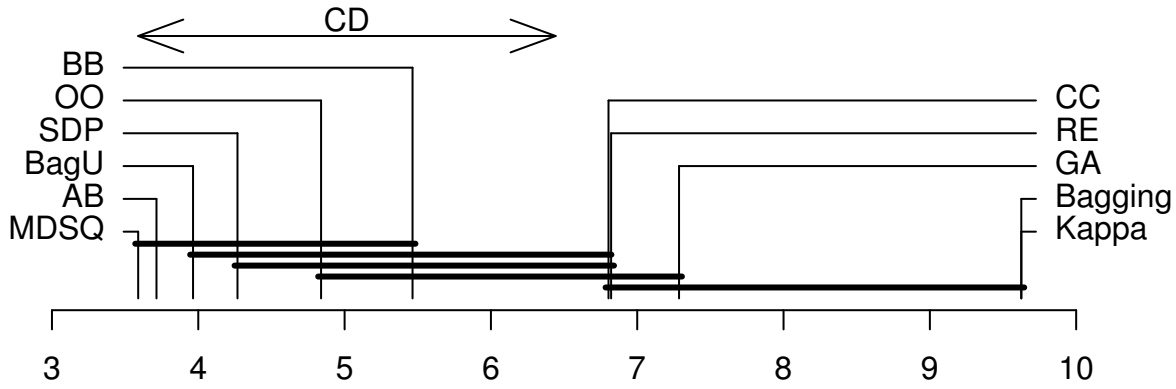

Fig. 5. Comparison of the different methods using the Nemenyi test. Classifiers not significantly different (p-value=0.05) are connected in the diagram.

The performance of the different ensembles in the different data sets is compared using the methodology proposed by Demšar in [49]. Fig. 5 displays the average rank of each ensemble in the problems investigated. In this diagram, methods whose average rank is not significantly different from each order, according to the Nemenyi test (p-value< 0.05), are connected with a horizontal line. The critical difference is shown for reference (CD=2.85 for 11 methods, 28 dataset and p-value< 0.05). The best overall performance corresponds to MDSQ-pruning, followed by Adaboost, bagging composed of unpruned trees and by SDP, OO and BB-pruning. The accuracy of these pruned ensembles is uniformly good in most of the classification problems investigated and is significantly better, in terms of average rank, than standard bagging.

F. Efficiency analysis

This section analyzes the efficiency of the different pruning techniques. Three aspects are of interest: The computational cost of extracting the pruned subensemble from the original pool of classifiers, the amount of memory required to store the pruned ensembles and, finally, their classification speed. Some of the operations can be performed in parallel: the generation of the initial pool of classifiers, and the predictions of the individual classifiers, once the pruned subensemble has been identified. By contrast, the reordering and the selection of the classifiers cannot be parallelized.

1) Complexity of the different pruning techniques: Table VII summarizes the space and time complexities of the different ensemble pruning methods in terms of the size of the initial pool

TABLE VII

COMPLEXITY OF THE DIFFERENT ENSEMBLE PRUNING TECHNIQUES.

Pruning method	Space complexity	Time complexity	
RE	$\mathcal{O}(N_{sel} \cdot T + \|y\|)$		$\mathcal{O}(T^2 \cdot N_{sel} \cdot \|y\|)$
Kappa	$\mathcal{O}(N_{sel} \cdot T + \|y\|^2)$		$\mathcal{O}(T^2 \cdot N_{sel})$
CC	$\mathcal{O}(N_{sel} \cdot T + \|y\|)$		$\mathcal{O}(T^2 \cdot N_{sel})$
MDSQ	$\mathcal{O}(N_{sel} \cdot T)$		$\mathcal{O}(T^2 \cdot N_{sel})$
OO	$\mathcal{O}(N_{sel} \cdot T)$	calculate angles	$\mathcal{O}(T \cdot N_{sel})$
		quicksort	$\mathcal{O}(T \cdot log(T))$
BB	$\mathcal{O}(N_{sel} \cdot T + \|y\|)$		$\mathcal{O}(T^2 \cdot N_{sel})$
GA	$\mathcal{O}(N_{sel} \cdot T)$ + $\mathcal{O}(P \cdot T)$	calculate fitness	$\mathcal{O}(E \cdot P \cdot T \cdot N_{sel} \cdot \|y\|)$
		evolve population	$\mathcal{O}(E \cdot P \cdot T)$
SDP	$\mathcal{O}(T^2)$ + $\mathcal{O}(N_{sel} \cdot T)$ +	calculate $\mathbf{G}$	$\mathcal{O}(T^2 \cdot N_{sel} \cdot \|y\| + T^3)$
	+ Implem. dependent	optimization process	$\mathcal{O}(T^3)$

of classifiers, T, the size of the selection set, N_{sel}, the number of classes of the classification problem, $|y|$, and, for the genetic algorithm, the size of the population, P, and the number of epochs, E. The memory requirements for the different algorithms are estimated assuming that classifiers are queried only once and that the outputs are stored in a matrix of size $N_{sel} \times T$. For large selection datasets it might not be possible to store the whole matrix in memory. In such a case this matrix would need to be stored in a secondary memory device, such as the hard disk. This would reduce the memory requirements to $\mathcal{O}(N_{sel})$ for most ensembles. However, the classification process would be slowed down by the required disk access. In this study, it has not been necessary to resort to secondary memory devices for storage.

To empirically investigate the dependence on T of the different heuristics a series of experiments on the *Pima Indian Diabetes* problem are performed. Table VIII reports the execution times of the different ensembles for initial bagging ensembles of 50, 100, 200, 400, 800 and 1600 CART trees. The values of the remaining parameters are $|y| = 2$, $N_{train} = 468$, $\mathcal{Z}_{sel} = \mathcal{Z}_{train}$. For the Genetic Algorithm, a population of T chromosomes is evolved for a maximum of $E = 200$ epochs. The times reported are averages over 100 executions in a Pentium® D processor at 2.8 GHz. The results displayed in this table show that the best results correspond to ordered

TABLE VIII

AVERAGE EXECUTION TIME (IN S) FOR THE ORDERING HEURISTICS FOR THE PIMA INDIAN DIABETES DATASET

.

# Trees	50	100	200	400	800	1600
RE	0.04	0.09	0.26	0.73	2.43	8.56
Kappa	0.02	0.07	0.20	0.72	2.76	10.76
CC	0.03	0.06	0.15	0.38	1.19	4.15
MDSQ	0.10	0.37	1.41	5.53	21.99	87.51
OO	0.02	0.04	0.06	0.11	0.21	0.42
BB	0.03	0.06	0.13	0.39	1.36	5.03
GA	3.80	7.75	24.44	144.57	765.31	3527.92
SDP	0.18	1.22	6.68	56.01	487.05	4293.91

bagging: Orientation ordering (OO) is the fastest method, with computation times that increase approximately linearly in T. The complexity of the remaining ordering heuristics is quadratic in T. Both SDP and GA are much slower than the ordering heuristics.

2) Memory requirements and speed of classification: Besides improvement in classification performance, the main benefits of ensemble pruning are lower storage requirements and higher classification speed. Storage requirements are linear in the size of the ensemble and in the complexity of the base classifiers. Assuming that the computational costs of retrieving examples from a data repository and of assigning class labels at the leaf nodes are small, classification speed mainly depends on (i) the number of classifiers in the ensemble, and (ii) the complexity of the base classifiers. In particular, the time needed by an ensemble of decision trees to classify an example can be estimated by multiplying the number of trees in the ensemble by the average number of tests that need to be made at internal nodes of a tree. Assuming balanced trees with a total number of nodes N_{nodes}, the average number of tests is $N_{tests} = log_2(N_{nodes}+1)-1$. Given this logarithmic dependence, most of the improvements in classification speed are expected to arise from the smaller number of classifiers in the pruned ensembles. Table IX displays the average number of nodes per tree in a bagging ensemble of standard (pruned) CART trees and in subensembles selected using MDSQ and SDP. The numbers between parentheses are estimates of the classification times by the different ensembles as a percentage of the time employed by a

TABLE IX

AVERAGE NUMBER OF NODES PER TREE AND CLASSIFICATION TIMES.

	Complete	Pruned bagging			Complete	Pruned bagging	
Dataset	**bagging**	**MDSQ**	**SDP**	**Dataset**	**bagging**	**MDSQ**	**SDP**
Audio	32.6	64.0 (25.9%)	60.2 (25.5%)	New-thyroid	9.1	14.0 (26.2%)	13.9 (26.1%)
Australian	6.7	14.3 (31.6%)	14.2 (31.6%)	Pendigits	381.8	468.2 (21.8%)	461.0 (21.8%)
Breast	12.1	25.1 (28.8%)	22.9 (27.8%)	Ringnorm	15.2	22.0 (24.5%)	22.2 (24.6%)
Diabetes	10.7	20.9 (28.5%)	18.4 (27.0%)	Satellite	82.8	131.2 (23.6%)	123.5 (21.3%)
Ecoli	11.7	20.5 (27.0%)	16.4 (24.5%)	Segment	67.0	98.2 (23.3%)	95.1 (23.1%)
German	13.1	24.6 (27.4%)	21.5 (25.9%)	Sonar	8.2	15.8 (29.2%)	15.7 (29.2%)
Glass	18.1	39.8 (28.0%)	37.3 (27.4%)	Spam	56.5	94.5 (24.2%)	88.9 (23.8%)
Heart	10.1	19.4 (28.4%)	17.4 (27.2%)	Tic-tac-toe	72.1	100.4 (22.9%)	99.7 (22.9%)
Horse-colic	4.5	6.7 (27.9%)	6.1 (26.3%)	Twonorm	17.7	28.7 (25.3%)	29.3 (25.6%)
Ionosphere	10.4	18.3 (27.4%)	17.5 (26.8%)	Vehicle	47.7	100.7 (25.9%)	97.5 (25.6%)
Labor	5.4	8.0 (27.2%)	8.1 (27.3%)	Votes	3.9	7.3 (33.3%)	7.1 (32.7%)
Led24	20.5	29.5 (24.1%)	24.8 (22.6%)	Vowel	201.7	214.6 (21.3%)	214.0 (21.3%)
Liver	14.4	34.8 (29.6%)	32.8 (29.0%)	Waveform	14.2	26.8 (27.2%)	25.9 (26.9%)
Magic04	98.9	142.9 (23.0%)	130.6 (22.5%)	Wine	8.4	10.1 (23.3%)	10.7 (24.0%)

complete bagging ensemble of CART trees. The speed of classification is increased by a factor between 3 and 5. An unforeseen result is that the average sizes of the CART trees selected by MDSQ or SDP are larger than in the initial bagging ensemble. Even though it is necessary to store fewer classifiers (21% of the initial ensemble), the classifiers selected tend to be more complex. The net effect is that pruned subensembles need less storage than complete bagging, but the reduction in memory requirements is smaller than expected. The selection of larger classification trees in the pruned ensembles has a negative, but typically small, impact in their classification speed.

G. Robustness of classification.

To investigate the performance of pruned ensembles in noisy classification problems, a series of experiments similar to those conducted by Dietterich in [6] are carried out. In these experiments, classifiers are built using corrupted versions of the original data. For each experiment, the class label of a fixed percentage of examples selected at random is switched. In the series

of experiments performed, the class labels in the training and test sets are modified with a probability of 0.0, 0.05, 0.10 and 0.20. Results are reported for nine problems from the UCI Repository [45] using a bagging ensemble of 100 standard (pruned) CART trees as the initial pool of classifiers. Experimental results for Adaboost with a maximum size of 100 pruned CART trees [9] are also reported for reference. The average error on the test sets for the different ensembles on each problem are displayed in Table X. The values reported correspond to averages and standard deviations over 100 realizations. The error rates that are significantly better than bagging are highlighted in boldface. Statistically significant improvements with respect to Adaboost are underlined. A paired *t-test* (p-value < 0.01) is used to determine the significance of the differences. As known from earlier studies, Adaboost is a powerful classification method and generally achieves lower error rates than bagging or pruned bagging. However, its performance rapidly deteriorates in noisy problems [6]. By contrast, bagging and pruned bagging ensembles are robust with respect to this type of noise. For noise levels in the class labels larger than $\approx 10\%$, bagging and pruned bagging outperform Adaboost in most of the datasets investigated. Even for high noise values, pruned bagging ensembles generally achieve significantly better results than complete bagging. As in the experiments without noise, the best overall results correspond to SDP and MDSQ pruning.

V. Conclusions

Using appropriate heuristics, it is possible to outperform bagging by selecting a subset of *complementary* classifiers from the pool of base learners in the original ensemble. The selection of an optimal subensemble from a given initial bagging ensemble is a difficult combinatorial optimization problem. With the computational resources currently available, only approximate solutions are accessible for ensembles of realistic size. In this work, pruning methods based on modifying the order of aggregation of classifiers in bagging are investigated. A nested sequence of ensembles of increasing size is constructed by incorporating at each iteration the classifier from the initial pool of bagging classifiers that is expected to improve the generalization performance of the ensemble the most. A pruned subensemble is obtained by stopping the ordered aggregation process when a fraction of the classifiers from the original ensemble has been included. The main conclusions for the datasets and the ensemble pruning techniques investigated are:

TABLE X

TEST ERRORS (AVERAGE ± STANDARD DEVIATION) ON NOISY CLASSIFICATION PROBLEMS.

	Problem	Bagging	RE	Kappa	CC	MDSQ	OO	BB	GA	SDP	AB
noise = 0 %	Breast	4.8±2.8	**4.2±2.4**	4.9±2.9	**4.2±2.5**	**4.0±2.6**	**4.1±2.5**	**4.1±2.6**	**4.2±2.5**	**3.8±2.4**	**3.3±1.9**
	German	25.6±3.0	**24.6±3.4**	25.9±3.0	**24.7±3.8**	**24.7±3.9**	25.0±3.5	**24.8±3.7**	25.1±3.3	**24.8±3.5**	25.0±3.3
	Heart	19.6±7.9	**19.2±7.3**	**17.7±6.7**	**18.8±6.9**	**17.7±6.8**	**17.3±6.9**	**18.5±7.1**	**19.2±7.2**	**17.7±6.9**	21.1±7.8
	Ionosphere	9.7±4.6	**7.5±4.4**	**8.0±4.4**	**7.3±4.0**	**7.3±3.9**	**7.7±4.3**	**7.5±4.2**	**7.5±4.3**	**7.2±3.9**	**6.4±3.7**
	Pima	24.9±3.9	**24.6±4.1**	**24.2±3.9**	**23.9±3.8**	**24.0±4.1**	**24.0±4.1**	**24.2±3.9**	25.2±5.1	**24.3±4.3**	26.7±3.9
	Sonar	25.1±9.5	**20.8±9.1**	24.3±9.0	**21.8±9.0**	**19.8±9.8**	**21.2±8.8**	**20.0±10.5**	**21.7±9.7**	**19.5±9.6**	**13.3±8.0**
	Vehicle	28.5±3.5	**25.6±4.1**	28.6±3.9	**25.7±4.0**	**25.2±4.2**	**25.2±4.4**	**25.4±3.8**	**25.9±4.1**	**25.3±4.2**	**22.7±3.6**
	Waveform	23.0±2.4	**20.6±1.3**	**22.2±1.7**	**20.4±1.2**	**19.9±1.2**	**20.3±1.3**	**20.3±1.4**	**20.5±1.4**	**20.0±1.2**	**18.0±0.8**
	Wine	4.4±5.0	4.2±5.1	6.7±6.3	4.4±5.1	**3.1±4.0**	4.0±4.7	**3.3±4.2**	3.6±4.7	**3.2±4.2**	4.1±4.8
noise = 5%	Breast	9.3±3.9	**8.3±3.6**	**9.0±3.8**	**8.2±3.7**	**8.1±3.6**	**8.5±3.8**	**8.4±3.7**	**9.0±3.6**	**8.4±3.7**	9.8±3.6
	German	27.7±3.8	**27.0±4.3**	28.1±3.2	**27.0±4.3**	**27.1±4.1**	**27.6±3.9**	**26.9±4.0**	**27.6±4.1**	**26.9±4.2**	29.0±3.8
	Heart	23.2±8.4	**22.7±7.6**	**21.6±6.7**	**22.0±7.4**	**22.2±7.6**	**22.7±7.8**	**22.2±7.6**	**23.1±8.0**	**22.1±7.5**	25.4±8.4
	Ionosphere	13.1±5.3	**11.8±4.6**	13.0±5.1	**11.5±4.4**	**11.3±4.5**	**11.8±4.8**	**12.2±4.9**	12.2±5.2	**11.5±4.7**	12.1±5.0
	Pima	26.6±4.6	**26.6±4.7**	**26.3±4.5**	26.7±4.4	**26.4±4.5**	**26.4±4.5**	26.7±4.6	27.0±5.0	**26.5±4.8**	30.4±4.8
	Sonar	28.1±10.1	**24.0±11.1**	26.2±10.8	**24.6±10.6**	**23.5±10.4**	**24.4±10.5**	**24.7±10.9**	**25.3±10.5**	**24.0±10.3**	**19.7±9.4**
	Vehicle	33.0±4.2	**29.9±4.4**	32.2±4.3	**29.6±4.4**	**30.1±4.3**	**30.0±4.2**	**30.3±4.3**	**29.8±4.0**	**30.2±4.7**	**27.0±4.1**
	Waveform	26.8±2.4	**24.3±1.3**	**25.8±1.6**	**24.2±1.2**	**23.9±1.3**	**24.2±1.3**	**24.5±1.3**	**24.5±1.4**	**23.9±1.2**	**22.7±1.0**
	Wine	9.9±7.9	9.3±7.0	10.6±7.8	9.1±7.0	**8.1±6.6**	**8.3±6.2**	**8.7±6.5**	8.9±7.2	**8.3±6.5**	**8.4±7.3**
noise = 10%	Breast	14.0±4.0	**13.6±3.9**	**13.8±3.7**	**13.4±3.8**	**13.4±3.8**	**13.7±4.0**	**13.5±3.7**	**13.9±3.8**	**13.3±3.7**	15.8±4.3
	German	31.0±4.1	**30.1±3.9**	31.3±3.9	**29.9±4.1**	**29.7±4.1**	**30.0±4.1**	**30.0±4.0**	**30.7±4.6**	**30.1±4.2**	32.5±5.0
	Heart	28.2±9.6	**28.1±9.9**	**26.5±8.8**	**27.5±9.5**	**27.5±9.6**	**27.6±9.7**	**27.7±9.6**	**28.0±9.0**	**27.3±9.6**	31.2±9.6
	Ionosphere	17.7±6.1	**15.8±5.8**	18.3±6.0	**15.7±6.1**	**15.9±6.2**	**15.7±6.1**	**16.5±6.1**	17.5±6.2	**15.8±6.0**	17.5±6.5
	Pima	30.1±4.3	**29.6±4.7**	**29.7±4.3**	**29.6±4.7**	**29.4±4.6**	**29.6±4.7**	**29.5±4.5**	30.1±4.5	**29.3±4.3**	33.8±5.3
	Sonar	30.1±10.3	28.1±9.5	29.6±8.8	27.7±9.4	**25.9±9.5**	**25.6±9.5**	**27.1±8.8**	28.3±8.9	**26.5±9.8**	**25.2±9.9**
	Vehicle	36.5±4.8	**34.0±4.7**	**35.6±4.3**	**33.4±4.8**	**33.5±4.5**	**33.7±4.6**	**34.0±4.6**	**33.6±4.9**	**33.8±4.6**	**31.5±4.3**
	Waveform	30.1±2.4	**28.1±1.2**	**29.2±1.7**	**27.9±1.2**	**27.6±1.2**	**28.0±1.3**	**28.3±1.4**	**28.6±1.4**	**27.7±1.2**	**27.0±1.1**
	Wine	15.9±9.1	14.7±7.8	14.8±9.1	14.9±8.2	**13.9±7.9**	**13.7±7.8**	**14.2±8.3**	15.7±8.7	14.6±8.3	15.0±9.1
noise = 20%	Breast	23.7±4.6	**23.2±4.7**	**23.3±4.2**	**23.3±4.4**	**23.0±4.4**	**23.3±4.5**	**23.2±4.3**	24.5±4.8	**23.0±4.3**	27.8±5.5
	German	36.2±4.4	**36.1±4.6**	36.4±4.3	**35.9±5.0**	**36.1±4.9**	36.6±4.8	**36.0±4.7**	38.2±5.5	**35.8±4.9**	39.9±4.5
	Heart	33.4±9.1	**32.6±9.5**	**32.9±8.9**	**33.2±10.2**	**33.3±10.0**	33.8±9.8	**33.1±10.2**	34.4±9.8	33.7±10.3	38.4±9.5
	Ionosphere	27.5±7.7	**26.5±7.8**	27.6±7.6	**26.6±8.5**	**26.5±8.0**	**26.3±8.2**	**26.3±8.0**	27.9±8.4	**26.6±7.8**	31.5±7.7
	Pima	35.2±4.9	35.7±5.5	35.8±5.8	35.8±5.6	35.8±5.5	35.9±5.3	35.7±5.5	37.7±5.8	35.7±5.3	40.6±5.4
	Sonar	36.1±10.3	36.5±11.8	38.4±10.8	35.7±10.7	36.4±10.7	36.7±11.4	36.0±10.8	36.5±10.6	35.7±11.3	36.6±11.6
	Vehicle	43.0±5.1	**40.6±5.1**	**42.0±5.3**	**40.5±5.3**	**40.2±5.3**	**40.8±5.3**	**40.9±5.1**	**41.6±5.6**	**41.0±5.0**	**41.2±5.4**
	Waveform	37.6±2.2	**35.8±1.3**	**36.8±1.9**	**35.6±1.2**	**35.3±1.1**	**36.0±1.5**	**36.0±1.3**	37.4±2.2	**35.0±0.6**	**35.6±1.3**
	Wine	26.0±11.4	24.8±10.9	25.9±11.7	25.1±11.1	**23.4±10.5**	24.9±11.3	**24.5±10.9**	26.8±11.8	**24.1±10.9**	25.8±11.3

- Exhaustive search confirms that the greedy ordering heuristics devised can efficiently identify near-optimal subensembles of increasing size.
- As more data is available for training, the errors of both bagging and pruned ensembles decrease. However, the larger the training set is, the lower the margin for improvement becomes. As a result, pruning becomes less effective in increasing classification accuracy.
- In general, it seems preferable to use all available data both for training the individual ensemble classifiers and for ordering, instead of setting apart a selection set, which is used only in the ordering phase.
- In general, the larger the initial pool of classifiers, the larger the pruned subensembles and the better their performance. However, the improvements in accuracy become smaller and eventually saturate as the initial pool of classifiers becomes larger.
- The pruning techniques seem to be less effective in ensembles of learners that are too complex. The size of ensembles of fully developed (unpruned) CART trees cannot be reduced without some deterioration in their performance. By contrast, pruned ensembles of decision stumps and of standard (pruned) CART trees generally outperform the corresponding initial bagging ensembles. The best overall performance is obtained with bagging ensembles of standard (pruned) CART trees with pruning rates of $20 - 40\%$.
- The generalization performance of an ensemble cannot be improved by pruning techniques based on individual properties of the ensemble members (e.g. the classification accuracy of the individual learners). As illustrated by the poor results obtained by kappa-pruning, diversity *by itself* is not sufficient either: It is important to take into account the complementarity of the classifiers.
- The classification accuracy of bagging ensembles pruned with MDSQ, OO and BB is comparable to that of subensembles selected by SDP, which is a very effective ensemble-pruning algorithm.
- Pruning techniques based on ordered aggregation have a lower computational cost than GA or SDP-pruning.
- The average size of the CART trees in the pruned subensembles is larger than in the original bagging ensemble. This means that the reduction in memory requirements is smaller than expected. Classification speed is only slightly diminished by this effect.
- In most of the datasets investigated, the performance of pruned bagging remains inferior to

Adaboost. However, pruned bagging ensembles retain the robustness of bagging in problems with noise in the class labels, where the performance of Adaboost is rather poor.

In summary, for the datasets investigated, bagging ensembles composed of CART trees pruned with either MDSQ, SDP, OO or BB exhibit a good overall performance in terms of accuracy and robustness of classification. Among the pruning techniques analyzed, OO has the lowest computational cost (linear in the size of the initial pool of classifiers, T). The ordering heuristics BB and MDSQ provide similar or better classification performance at a larger computational cost (quadratic in T). Ensembles pruned using SDP are also very accurate. However, they are costly to construct (cubic in T).

REFERENCES

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, 1990.

[2] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. The MIT Press, 1995, pp. 231–238.

[3] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, no. 2, pp. 181–207, May 2003.

[4] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[5] J. R. Quinlan, "Bagging, boosting, and C4.5," in *Proc. of the 13th National Conference on Artificial Intelligence*. AAAI Press and the MIT Press, 1996, pp. 725–730.

[6] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. of the 2nd European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[9] ——, "Experiments with a new boosting algorithm," in *Proc. of the 13th International Conf. on Machine Learning*. Morgan Kaufmann, 1996, pp. 148–156.

[10] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1-2, pp. 105–139, 1999.

[11] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. of the 23rd International Conference on Machine Learning*. New York, NY, USA: ACM Press, 2006, pp. 161–168.

[12] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

[13] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proc. of the 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 211–218.

[14] A. L. Prodromidis and S. J. Stolfo, "Cost complexity-based pruning of ensemble classifiers," *Knowledge and Information Systems*, vol. 3, no. 4, pp. 449–469, 2001.

[15] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[16] Z.-H. Zhou and W. Tang, "Selective ensemble of decision trees," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, ser. Lecture Notes in Artificial Intelligence, Q. Liu, Y. Yao, and A. Skowron, Eds., vol. 2639. Springer, 2003, pp. 476–483.

[17] G. Martínez-Muñoz and A. Suárez, "Aggregation ordering in bagging," in *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*. Acta Press, 2004, pp. 258– 263.

[18] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proc. of the 21st International Conference on Machine Learning*. New York, NY, USA: ACM Press, 2004, p. 18.

[19] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005.

[20] G. Martínez-Muñoz and A. Suárez, "Pruning in ordered bagging ensembles," in *Proc. of the 23rd International Conference on Machine Learning*, 2006, pp. 609–616.

[21] ——, "Using boosting to prune bagging ensembles," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 156–165, 2007.

[22] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, pp. 1315–1338, 2006.

[23] D. Hernández-Lobato, J. M. Hernández-Lobato, R. Ruiz-Torrubiano, and Á. Valle, "Pruning adaptive boosting ensembles by means of a genetic algorithm," in *Proc. of the 7th International Conference on Intelligent Data Engineering and Automated Learning*, ser. Lecture Notes in Computer Science, E. Corchado, H. Yin, V. J. Botti, and C. Fyfe, Eds., vol. 4224. Springer, 2006, pp. 322–329.

[24] C. Tamon and J. Xiang, "On the boosting pruning problem," in *Proc. of the 11th European Conference on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, R. L. de Mátaras and E. Plaza, Eds., vol. 1810. Springer, 2000, pp. 404–412.

[25] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Effective voting of heterogeneous classifiers," in *Proc. of the 11th European Conference on Machine Learning*, ser. Lecture Notes in Artificial Intelligence, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds., vol. 3201. Springer, 2004, pp. 465–476.

[26] G. Tsoumakas, L. Angelis, and I. Vlahavas, "Selective fusion of heterogeneous classifiers," *Intelligent Data Analysis*, vol. 9, pp. 511–525, 2005.

[27] I. Partalas, G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Ensemble pruning using reinforcement learning," in *Proc. of the 4th Helenic Conference on Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, Eds., vol. 3955. Springer, 2006, pp. 301–310.

[28] J. Meynet and J.-P. Thiran, "Information theoretic combination of classifiers with application to adaboost," in *Proc. of the 7th International Workshop on Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, M. Haindl, J. Kittler, and F. Roli, Eds., vol. 4472. Springer, 2007, pp. 171–179.

[29] W. Fan, F. Chu, H. Wang, and P. S. Yu, "Pruning and dynamic scheduling of cost-sensitive ensembles," in *Proc. of the 18th National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, 2002, pp. 146–151.

[30] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 16, no. 1, pp. 66–75, 1994.

[31] K. Woods, W. P. Kegelmeyer, and K. W. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[32] A. Tsymbal and S. Puuronen, "Bagging and boosting with dynamic integration of classifiers," in *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, ser. Lecture Notes in Computer Science, D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, Eds., vol. 1910. Springer, 2000, pp. 116–125.

[33] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition*, vol. 34, no. 9, pp. 1879–1881, 2001.

[34] P. Domingos, "Knowledge acquisition from examples via multiple models," in *Proc. of the 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 98–106.

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

[36] C. Demir and E. Alpaydin, "Cost-conscious classifier ensembles," *Pattern Recognition Letters*, vol. 26, no. 14, pp. 2206–2214, 2005.

[37] A. M. Canuto, M. C. Abreu, L. de Melo Oliveira, J. C. Xavier Jr., and A. de M. Santos, "Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles," *Pattern Recognition Letters*, vol. 28, pp. 472–468, 2007.

[38] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern Recognition Letters*, vol. 22, no. 1, pp. 25–33, 2001.

[39] B. Bakker and T. Heskes, "Clustering ensembles of neural network models," *Neural Networks*, vol. 16, no. 2, pp. 261–269, 2003.

[40] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Transanctions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 173–180, 2007.

[41] C. E. Brodley and T. Lane, "Creating and exploiting coverage and diversity," in *Proc. of the AAAI-96 Workshop on Integrating Multiple Learned Models*, 1996, pp. 8–14.

[42] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, no. 1, pp. 83–98, 2005.

[43] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 12, no. 5, pp. 1651–1686, 1998.

[44] L. Breiman, "Arcing the edge," University of California, Berkeley, CA, Tech. Rep., 1997.

[45] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[46] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "Selection of decision stumps in bagging ensembles," in *Proc. of the 17th International Conference on Artificial Neural Networks*, ser. Lecture Notes in Computer Science, J. M. de Sá, L. A. Alexandre, W. Duch, and D. P. Mandic, Eds., vol. 4668. Springer, 2007, pp. 319–328.

[47] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Berlin: Springer-Verlag, 2003.

[48] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Machine Learning*, vol. 52, no. 3, pp. 239–281, 2003.

[49] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.