A Double Pruning Scheme for Boosting Ensembles

Víctor Soto, Sergio García-Moratilla, Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez, *Fellow, IEEE*

***Abstract*—Ensemble learning consists of generating a collection of classifiers whose predictions are then combined to yield a single unified decision. Ensembles of complementary classifiers provide accurate and robust predictions, which are often better than the predictions of the individual classifiers in the ensemble. Nevertheless, ensembles also have some drawbacks: typically, all classifiers are queried to compute the final ensemble prediction. Therefore, all the classifiers need to be accessible to address potential queries. This entails larger storage requirements and slower predictions than a single classifier. Ensemble pruning techniques are useful to alleviate these drawbacks. Static pruning techniques reduce the ensemble size by selecting a sub-ensemble of classifiers from the original ensemble. In dynamic pruning, the querying process is halted when the partial ensemble prediction is sufficient to reach a stable final decision with a reasonable amount of confidence. In this paper, we present the results of a comprehensive analysis of static and dynamic pruning techniques applied to Adaboost ensembles. These ensemble pruning techniques are evaluated on a wide range of classification problems. From this analysis, one concludes that the combination of static and dynamic pruning techniques provides a notable reduction in the memory requirements and an improvement in the classification time without a significant loss of prediction accuracy.**

***Index Terms*—Adaboost, double pruning, ensemble pruning, instance-based pruning, semi-definite programming.**

I. Introduction

ENSEMBLE learning is a branch of supervised machine learning concerned with the generation and integration of multiple classifiers to obtain a single more accurate predictor [1]–[7]. The final decision of the ensemble for a new data instance is computed by combining the individual predictions of the ensemble classifiers. There is extensive empirical evidence in the literature showing that this learning scheme is an effective strategy that can improve the generalization performance and the robustness of the decisions made [4], [8]–[10]. Furthermore, ensemble methods have been reported to obtain state-of-the-art performance in a wide range of learning tasks such as recommendation, weather forecast, real-time human pose recognition, robust real-time face recognition, gene function prediction, reverse engineering biological networks or credit card fraud detection [11]–[17]. Notwithstanding, the major drawback of ensemble learning algorithms is the elevated cost in terms of memory requirements and computational time. Specifically, the classifiers of the ensemble need to be stored in memory so that they can be readily queried to output the combined ensemble decision. In our experience, oftentimes the memory required for storing the ensemble is several times larger than the memory required for storing the training data. For example, in a large dataset like *Magic04* [18] a 201 tree Adaboost ensemble occupies in disk around 3.0 MiB and the training set around 1.5 MiB. For *Breast* [18] the ensemble is $\approx$140 KiB and the data it was trained on $\approx$14 KiB. These values are obtained using our implementation for the on-disk storage of the ensembles. This implementation only saves the necessary data of the tree nodes. These differences are much more pronounced in other implementations like WEKA [19]. Additionally, since all the ensemble classifiers must be queried to obtain the final ensemble decision, the classification time of ensemble methods is significantly larger than that of a single predictor. To alleviate these shortcomings, several ensemble pruning techniques have been proposed [20]–[23]. These can be divided in two different groups: static and dynamic. The goal of static (off-line) pruning techniques is to select a subset of classifiers of a fixed size whose combined decisions have similar or better accuracy than the complete ensemble. Dynamic (on-line) pruning techniques reduce the number of classifiers needed to obtain the final ensemble decision for each specific test instance during classification. Since only a small number of classifiers need to be queried, these techniques effectively speed-up the classification process. However, unlike static techniques, dynamic techniques require that all the ensemble members be readily available.

In this paper, we investigate the application of an effective pruning scheme for Adaboost ensembles [24]–[26]. This scheme consists in, first, reducing the set of classifiers of the ensemble by applying a static pruning technique. As a result of this type of pruning both the prediction performance and the time needed for classification are improved. Then, dynamic pruning is applied during the classification process to further accelerate the prediction of the class labels associated to the new instances. This double pruning scheme has been used previously in the context of bagging ensembles composed of CART trees [27]. Nevertheless, the application to Adaboost ensembles is challenging for several reasons. First, Adaboost is a sequential method in which the ensemble grows by generating a new classifier using a training set in which

Manuscript received January 11, 2013; revised November 14, 2013 and March 7, 2014; accepted March 8, 2014. Date of publication April 25, 2014; date of current version November 13, 2014. This work was supported by the Spanish Dirección General de Investigación under Project TIN2010-21575-998 C02-02. This paper was recommended by Associate Editor E. Santos.

V. Soto, G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez are with Escuela Politécnica Superior, Universidad Autónoma de Madrid, Cantoblanco 28049, Spain (e-mail: victor.soto@estudiante.uam.es; gonzalo.martinez@uam.es; daniel.hernandez@uam.es; alberto.suarez@uam.es).

S. García-Moratilla is with Risklab-Madrid, Universidad Autónoma de Madrid, Cantoblanco 28049, Spain (e-mail: sergio.garcia@qrr.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2313638

2168-2267 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

the importance of the examples that are misclassified by the previous ensemble members is emphasized. As a result of this learning strategy Adaboost ensembles generally achieve zero error in the training set after combining only a few classifiers [28]. This poses some difficulties for static pruning techniques, which generally attempt to select sub-ensembles with low training error. Another difficulty is that Adaboost introduces strong dependencies among the generated classifiers. The presence of these dependencies invalidates the theoretical analysis of dynamic pruning methods [22], [23], which rely on the independence of the predictions of the individual classifiers for a given test instance. This independence assumption is only fulfilled by parallel randomized ensembles [29]. In spite of these difficulties, experiments carried out on several classification problems from the UCI repository [18] show that the combined pruning scheme is effective also for Adaboost ensembles. Specifically, in the problems investigated, static pruning techniques, besides significantly reducing the number of classifiers that have to be queried in Adaboost ensembles to obtain the final decision, also lead to small but noticeable improvements on the prediction performance with respect to the complete ensemble. A subsequent application of dynamic pruning techniques on the selected sub-ensemble further reduces the average number of classifiers that need to be queried to yield a final decision. This further speed-up increment of the classification process does not lead to a significant increase in the prediction error.

In summary, the goal of this paper is three-fold.

1) To perform a comprehensive analysis of static pruning techniques applied to Adaboost.
2) To determine whether dynamic pruning techniques can be used on Adaboost ensembles in spite of their not fulfilling the assumption of independence among the predictions of ensemble classifiers.
3) To assess the effectiveness of applying dynamic pruning to Adaboost ensembles that have been pruned with static pruning methods.

The result is a double pruning method that combines the advantages of static and dynamic techniques. Namely, better generalization performance, reduced storage requirements (only the classifiers in the pruned ensemble need to be readily available in memory) and improved classification time.

The rest of the paper is organized as follows: Section II reviews Adaboost, the ensemble algorithm considered in this paper. Several static and dynamic pruning techniques are described in Section III. This section also introduces the double pruning technique that is then applied to Adaboost ensembles. Section IV presents the results of the experiments carried out to assess the effectiveness of these pruning techniques when they are applied to Adaboost ensembles. In particular, these experiments show that double pruning reduces the memory requirements and significantly improves the speed of classification of Adaboost ensembles. Finally, the conclusions of the investigation are summarized in Section V.

II. Adaboost

Boosting refers to a family of learning algorithms that were introduced to generate more expressive models (a strong learner) by combining simple predictors whose individual performance is only slightly better than random guessing (weak learners) [24], [25], [30]–[35]. In this paper, we focus on Adaboost [24], [25], which is a powerful ensemble construction method that is representative of these types of algorithms [31], [33], [36]–[38]. Adaboost induces a sequence of predictors by repeatedly applying the same learning algorithm to weighted versions of the initial training data. Initially, the instances are weighted uniformly. At each step a classifier is trained on the weighted training dataset. After the tth classifier is built, the weights of the instances are updated based on its performance. The weights of the instances that are incorrectly classified by the last predictor are increased while the weights of the instances correctly classified are decreased. In this way, subsequent classifiers focus on the instances that are more difficult to classify. The final combined decision is obtained as a weighted combination of the predictions of the individual classifiers of the ensemble. The voting weight β_t corresponding to the tth classifier of the ensemble is computed in terms of the prediction performance ϵ_t of that classifier on the corresponding weighted training set used for induction.

It is possible to show that Adaboost iteratively minimizes a convex exponential loss function that is an upper bound on the mis-classification error [39]. Thus, it is eventually guaranteed to provide zero training error for a sufficiently large number of ensemble classifiers T. Nevertheless, there is empirical evidence showing that including extra classifiers in the ensemble even when the training error reaches zero can be beneficial and can further reduce the generalization error of the ensemble [28]. In the literature, Adaboost has been shown to perform very well in a wide range of classification tasks [31], [40], [41]. However, it is very sensitive to noise in the class labels and it can be severely affected by over-fitting [4], [21], [41]. Since each new predictor that is included in the ensemble corrects the prediction errors of the previous classifier in the sequence, the performance improvements of Adaboost are often related to a decrease in the bias component of the generalization error [30], [40], [42]. Additional experiments show that the first iterations of Adaboost mainly reduce bias, while the later ones reduce variance [43]. Finally, Adaboost has been also shown to generate both accurate and diverse classifiers [9]. This also explains its good prediction performance. In particular, the combination of accurate and diverse classifiers in the ensemble typically leads to better prediction results [1], [8], [44], [45].

III. Ensemble Pruning

The goal of this section is to describe the static and dynamic ensemble pruning techniques that will be applied to classification ensembles generated by Adaboost in the section on experiments. We have selected a representative sub-set of ensemble pruning techniques that have shown good results when applied to classification ensembles. There are other pruning methods besides those described here. For a comprehensive taxonomy and review of ensemble pruning methods, the reader is referred to [46].

A. Static Pruning Techniques

This approach to ensemble pruning consists in selecting from the original ensemble a subset of representative classifiers

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

whose combined performance is equivalent or better than the complete ensemble. More precisely, the problem is to identify a small sub-ensemble with good generalization performance based on estimates computed on the training set or on a separate validation set. This is a computationally expensive problem. In particular, even if it were possible to have accurate estimates of the generalization error of the candidate sub-ensembles, finding the optimal one involves comparing all the possible 2^T-1 non-empty sub-ensembles that can be extracted from the original ensemble.

Several techniques have been proposed to provide approximate solutions to this problem. We introduce first an efficient ensemble pruning method that uses semi-definite programming (SDP) to find a near-optimal sub-ensemble of a fixed size k. This ensemble pruning method is described in detail by Zhang *et al.* in [20]. SDP-pruning considers an ensemble composed of T classifiers and a validation dataset $\mathcal{Z}$ of n labeled instances. Furthermore, Zhang *et al.* introduced a matrix $\mathbf{G}$, whose element G_{ij} is the number of common errors between classifier i and classifier j, on $\mathcal{Z}$, where $i,j=1,2,\ldots,T$ and $i\neq j$. The value of each diagonal term G_{ii} is the number of errors made by classifier i on $\mathcal{Z}$. The matrix is normalized so that its elements are on the same scale (values between 0 and 1)

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

The sum of the diagonal terms, $\sum_{i=1}^{T}\tilde{G}_{ii}$, measures the performance of the ensemble of classifiers in the dataset $\mathcal{Z}$. The sum of the out-of-diagonal terms, $\sum_{ij,i\neq j}\tilde{G}_{ij}$, measures the diversity of the ensemble. Therefore, $\tilde{g}=\sum_{ij}\tilde{G}_{ij}$ is a measure that considers both the individual performance of the ensemble classifiers and also their diversity. The smaller its value, the better the expected performance of the corresponding ensemble. In the literature, there are many measures of the ensemble diversity [10]. The $\tilde{g}$ measure considered by Zhang *et al.* integrates both the performance and diversity of the classifiers and is very convenient for quadratic programming.

SDP-pruning reformulates the sub-ensemble selection problem of size k as a quadratic integer programming problem

$$\underset{\mathbf{s}}{\operatorname{argmin}}\quad \mathbf{s^T}\cdot\tilde{\mathbf{G}}\cdot\mathbf{s},\quad s.t.\sum_{i=1}^{T}s_i=k,\quad s_i\in\{0,1\}. \tag{2}$$

The binary variable $s_i\in\{0,1\}$ indicates whether classifier i is selected ($s_i=1$) or not ($s_i=0$). The size of the pruned ensemble, k, is specified beforehand. Equation (2) is a standard problem for binary optimization, which is NP-hard. In [20] the solution is approximated by applying SDP to a convex relaxation of the original problem whose solution can be found in polynomial time. Thus, the solution of SDP-pruning needs not be equal to the optimal solution of the sub-ensemble selection problem. Nevertheless, SDP has been shown to perform well in practice [20], [21].

Another important family of static pruning techniques is ordered aggregation. The techniques in this family use a greedy strategy based on modifying the aggregation order of the classifiers in the ensemble. The greedy search starts with an initial empty sub-ensemble and iteratively adds in each step the classifier that optimizes a measure that is expected to be correlated with the generalization capacity. Specifically, from the sub-ensemble $\mathcal{S}_{u-1}$ of size $u-1$, the sub-ensemble of size $\mathcal{S}_u$ is constructed by incorporating a single classifier s_u from the remaining pool of classifiers, i.e., $E_T\setminus S_{u-1}$, where E_T denotes the complete ensemble. The result is an ordering of the predictors which can be used to obtain a pruned sub-ensemble by selecting the first k classifiers in the ordered sequence. The search can be stopped once k classifiers have been selected in this way. A comprehensive analysis of order aggregation techniques is found in [21].

The idea of ordered aggregation can also be used to find an approximate solution to (2). In this paper, we propose an ordered aggregation technique that incorporates the classifier from the original pool of classifiers that is expected to minimize the target value of (2) the most. We refer to this method as Greedy-G.

Another ordered aggregation algorithm that optimizes a measure different from the one employed in (2) is *margin distance minimization* (MDM). This technique was introduced in [47] and has been exhaustively analyzed in [21]. This metric iteratively incorporates into the partially aggregated sub-ensemble of size u, $\mathcal{S}_u$, the classifier j that minimizes a classification distance between the optimum margin and the actual margin of the instances.

Another static ensemble pruning technique is the diversity regularized ensemble pruning (DREP) proposed in [48]. DREP is another particular case of ordered aggregation that aims at optimizing the estimated sub-ensemble prediction performance on the validation set $\mathcal{Z}$. However, to provide sub-ensembles of better generalization properties the optimization process is regularized by the sub-ensemble diversity.

Partalas *et al.* [49] introduce a different measure that can be combined with an ordered aggregation strategy to search for optimal sub-ensembles. The measure proposed is referred to as uncertainty weighted accuracy (UWA). It is based on defining the following events that concern the decision of a classifier h_i and a partially aggregated sub-ensemble $\mathcal{S}_{u-1}$ of size $u-1$:

$$\begin{aligned}
&e_{tf}(h_i,\mathcal{S}_{u-1},\mathbf{x}_l,y_l){:}\,h_i(\mathbf{x}_l)=y_l\wedge s_{u-1}(\mathbf{x}_l)\neq y_l\\
&e_{ft}(h_i,\mathcal{S}_{u-1},\mathbf{x}_l,y_l){:}\,h_i(\mathbf{x}_l)\neq y_l\wedge s_{u-1}(\mathbf{x}_l)=y_l\\
&e_{ff}(h_i,\mathcal{S}_{u-1},\mathbf{x}_l,y_l){:}\,h_i(\mathbf{x}_l)=y_l\wedge s_{u-1}(\mathbf{x}_l)=y_l\\
&e_{tt}(h_i,\mathcal{S}_{u-1},\mathbf{x}_l,y_l){:}\,h_i(\mathbf{x}_l)\neq y_l\wedge s_{u-1}(\mathbf{x}_l)\neq y_l
\end{aligned}$$

where $s_{u-1}(\cdot)$ denotes the prediction of the sub-ensemble $\mathcal{S}_{u-1}$. Furthermore, let NT_l be the fraction of classifiers in the current sub-ensemble $\mathcal{S}_{u-1}$ that classify the instance $(\mathbf{x}_l,y_l)\in\mathcal{Z}$ correctly. Similarly, let $\mathrm{NF}_l=1-\mathrm{NT}_l$ be the fraction of classifiers in $\mathcal{S}_{u-1}$ that classify $(\mathbf{x}_l,y_l)\in\mathcal{Z}$ incorrectly. The UWA measure for classifier i and sub-ensemble $\mathcal{S}_{u-1}$ is

$$\begin{aligned}
&\mathrm{UWA}(h_i,\mathcal{S}_{u-1})\\
&\quad=\sum_{l=1}^{n}\Big\{\mathbb{I}(e_{tf}(h_i,\mathcal{S}_{u-1}))\mathrm{NT}_l-\mathbb{I}(e_{ft}(h_i,\mathcal{S}_{u-1}))\mathrm{NF}_l\\
&\qquad+\mathbb{I}(e_{tt}(h_i,\mathcal{S}_{u-1}))\mathrm{NF}_l-\mathbb{I}(e_{ff}(h_i,\mathcal{S}_{u-1}))\mathrm{NT}_l\Big\}.
\end{aligned} \tag{3}$$

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

In summary, examples for which the ensemble's decision is highly uncertain influence the metric stronger, while examples where most of the ensemble's members agree do not influence the metric a lot. The pruning method of Partalas *et al.* simply incorporates into the sub-ensemble of size $u-1$, $\mathcal{S}_{u-1}$, the classifier from the remaining pool of classifiers, $E_T \setminus S_{u-1}$, maximizing (3).

The last static ensemble pruning technique that we consider is the one described in [50]. The method proposed, ensemble pruning via individual contribution ordering (EPIC), works by computing the individual contribution of each classifier h_i to the complete ensemble E_T. This individual contribution, IC_i, is defined as

$$\mathrm{IC}_i = \sum_{l=1}^{n} \mathrm{IC}_i^{(l)} \tag{4}$$

where $\mathrm{IC}_i^{(l)}$ is the individual contribution of classifier h_i on the l-th data point in the pruning dataset $\mathcal{Z}$, of size n. $\mathrm{IC}_i^{(l)}$ is then defined as follows:

$$\mathrm{IC}_i^{(l)} = \begin{cases} 2v_{\max}^{(l)} - v_{h_i(\mathbf{x}_l)}^{(l)}, & \text{if } h_i(\mathbf{x}_l) = y_l \text{ and } h_i(\mathbf{x}_l) \text{ does not equal the ensemble prediction,} \\ v_{\mathrm{sec}}^{(l)} & \text{if } h_i(\mathbf{x}_l) = y_l \text{ and } h_i(\mathbf{x}_l) \text{ equals the ensemble prediction,} \\ -(v_{h_i(\mathbf{x}_l)}^{(l)} + (v_{\mathrm{correct}}^{(l)} - v_{\max}^{(l)})), & \text{otherwise} \end{cases} \tag{5}$$

where $v_{\max}^{(l)}$ is the number of votes in the majority class prediction of the ensemble for instance $(\mathbf{x}_l, y_l)$; $v_{h_i(\mathbf{x}_l)}^{(l)}$ is the number of votes in the class $h_i(\mathbf{x}_l)$ in the ensemble prediction for instance $(\mathbf{x}_l, y_l)$; $v_{\mathrm{sec}}^{(l)}$ is the number of votes in the second majority class in the ensemble prediction for instance $(\mathbf{x}_l, y_l)$; and $v_{\mathrm{correct}}^{(l)}$ is the number of votes for class label y_l in the ensemble prediction for instance $(\mathbf{x}_l, y_l)$. $2v_{\max}^{(l)} - v_{h_i(\mathbf{x}_l)}^{(l)}$ estimates the positive contribution of the classifier to the ensemble, i.e., the value of the predictions in the non-majority class. $v_{\mathrm{sec}}^{(l)}$ is also an estimate of the *degree of positive contribution*. Intuitively, if the majority of classifiers predict correctly the instance, the contribution of h_i is not very valuable because without its prediction the ensemble would still be correct. Finally, $v_{h_i(\mathbf{x}_l)}^{(l)} + (v_{\mathrm{correct}}^{(l)} - v_{\max}^{(l)})$ measures the degree of negative contribution of the classifier to the ensemble, which should be minimized. The EPIC method simply sorts the classifiers in the ensemble in decreasing order of IC_i to then select only the first p classifiers of the ordered sequence.

B. Dynamic Pruning Techniques

Dynamic pruning techniques focus on reducing the time needed by the ensemble to output a decision. These techniques dynamically determine, for each test instance to be classified, the number classifiers that need to be queried. Assuming that the queries are sequential, the process is halted when there is enough evidence that the partial ensemble prediction will not be changed by the remaining (unknown) classifier predictions. For example, in binary classification problems when assuming majority voting one can output the ensemble prediction when more than half of the ensemble classifiers agree on their predictions. This rule yields the same prediction as the complete ensemble without querying all the classifiers. Nevertheless, it is possible to achieve higher dynamical pruning rates if some uncertainty is allowed in the estimation of the final ensemble prediction. Specifically, the querying process can be halted when the probability that the predictions of the current sub-ensemble of size t and the complete ensemble coincide is above a specified threshold $p^{\mathrm{final}}(\mathbf{x}; t) \geq \alpha$, where the threshold α is smaller than, but close to 1 [22], [23]. Dynamic methods significantly improve the classification speed, but do not reduce the storage requirements of the ensemble because all the classifiers need to be available during the classification process to resolve potential queries [22], [23], [51]–[53].

Statistical instance-based pruning (SIBP) is an example of a dynamic ensemble pruning technique [22]. Assume that the class predictions of t classifiers from the original ensemble of size T are known. Let $\mathbf{t}(\mathbf{x}) = \{t_1(\mathbf{x}), \ldots, t_l(\mathbf{x})\}$ with $\sum_{i=1}^{l} t_i(\mathbf{x}) = t$ be a vector whose components summarize the observed votes for the different l class labels, (see $t_3(\mathbf{x})$ is the number of votes observed for the third class label for instance $\mathbf{x}$). Let $p(\mathbf{t}(\mathbf{x}); T)$ denote the probability that the prediction of the complete ensemble, which is composed of T classifiers, coincides with $k_{\mathbf{t}(\mathbf{x})}$, the majority class from the vector of observed votes $\mathbf{t}(\mathbf{x})$. A closed form expression for the probability $p(\mathbf{t}(\mathbf{x}); T)$ can be derived assuming that: (i) the predictions of the classifiers for a given test instance are independent when conditioned to the training data; (ii) the prior distribution of votes for the different classes is uniform; and (iii) majority voting is used for the combination of the individual predictions. From these assumptions and making use of Bayes' theorem, the following expression can be derived:

$$p(\mathbf{t}(\mathbf{x}); T) = \frac{(T-t)!}{(t+l)_{T-t}} \sum_{\mathbf{T} \in \mathcal{T}_{\mathbf{t}(\mathbf{x})}} \frac{\prod_{i=1}^{l} (t_i(\mathbf{x}) + 1)_{T_i - t_i(\mathbf{x})}}{\prod_{i=1}^{l} (T_i - t_i(\mathbf{x}))!} \tag{6}$$

where $\mathbf{T} = \{T_1, \ldots, T_l\}$ is a vector whose components summarize the potential (future) votes for each different class label for the complete ensemble; and $\mathcal{T}_{\mathbf{t}(\mathbf{x})}$ is the set of vectors $\mathbf{T}$ compatible with the observed tally of votes $\mathbf{t}(\mathbf{x})$. That is, each $\mathbf{T} \in \mathcal{T}_{\mathbf{t}(\mathbf{x})}$ satisfies the following restrictions.

1) $\sum_{i=1}^{l} T_i = T$, i.e., the final sum of votes has to be equal to the ensemble size, T.
2) $T_i \geq t_i(\mathbf{x})\ \forall i$, i.e., the potential votes $\mathbf{T}$ have to be compatible with the votes already observed in $\mathbf{t}(\mathbf{x})$.
3) Let $k_{\mathbf{t}(\mathbf{x})}$ the majority class from the vector of votes $\mathbf{t}(\mathbf{x})$. Then $T_{k_{\mathbf{t}(\mathbf{x})}} > T_j\ \forall j \neq k_{\mathbf{t}(\mathbf{x})}$, i.e., the observed majority class in $\mathbf{t}(\mathbf{x})$ is preserved.

If $p(\mathbf{t}(\mathbf{x}); T) = 1$, the prediction of the complete ensemble necessarily coincides with the prediction of the sub-ensemble of size t. However, if it is acceptable that, with a small probability $1-\alpha$, the prediction of the partially polled ensemble and the prediction of the complete ensemble disagree, the voting process can be stopped when $p(\mathbf{t}(\mathbf{x}); T) \geq \alpha$. The final classification would be given as the combined decision of the partially polled classifiers only. These predictions are expected to agree with the prediction of the complete ensemble with

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

probability at least α. Furthermore, if a lookup table of critical values is precalculated for specific values of T and α, the method described does not imply any computation overhead in the classification phase for both binary [22] and multiclass problems [54].

Finally, in [53] it is considered an approximation to the method described in this section that allows to compute the look-up table more efficiently. The method proposed performs similarly to SIBP for very large ensembles ($T > 500$). However, for normal size ensembles ($T = [100 - 200]$) SIBP tends to select a smaller number of classifiers [53]. There are also other dynamic methods in the literature, see [55] and [56], which aim at improving the accuracy of the ensemble by selecting for each test instance to be predicted a specific subset of classifiers. The disadvantage is that they incur in additional computational overheads during the classification phase. For these reasons, we restrict the present study to SIBP.

C. Application of Static and Dynamic Pruning to Adaboost Ensembles

The application of the static and dynamic pruning techniques described in Section III-A and B to Adaboost ensembles is not guaranteed to be effective. In particular, there are several assumptions made by these techniques that are not satisfied by the ensembles generated by Adaboost. For example, the objective defined in (2) does not consider the possibility of using weights in the voting process, as Adaboost does. Furthermore, each individual classifier in Adaboost ensembles is built using a training set in which the weight of the individual training instances depends on the errors of the previously generated classifiers. Thus, the predictions of the ensemble classifiers for a given test instance are dependent, which is contrary to the assumption of independence made by SIBP. This means that the dependencies among classifiers, introduced by Adaboost are ignored, leading to biased estimates of $p(\mathbf{t}(\mathbf{x}); T)$ in (6).

To make the static and dynamic pruning techniques described in the previous sections applicable to Adaboost two modifications to the original algorithm need to be made: first, instead of weighted voting, majority voting is used to combine the predictions of the classifiers, as in [20]. Empirically, we have observed that the accuracy of the predictions obtained by simple majority is very similar to weighted voting in Adaboost. Second, we randomly permute the order in which the classifiers are aggregated. In this way, the sequential dependencies among their predictions are effectively removed. Once these modifications have been made, both static and dynamic pruning techniques can be safely applied to Adaboost ensembles.

The double pruning algorithm considered in this paper works as follows: first, the Adaboost ensemble is pruned using one of the static techniques described in Section III-A. We present results for each of the different pruning methods described. In this process, the training set is used as the validation set to discriminate among the sub-ensembles. Then, the resulting pruned ensemble is dynamically pruned using SIBP. We have also performed experiments to determine the effects of these two pruning strategies when they are applied separately. In the case of Adaboost dynamically pruned using SIBP, we report results for both the original order of the classifiers produced by Adaboost and when the order of these classifiers is randomly permuted.

IV. Experiments

In this section, we carry out experiments to assess the effectiveness of the pruning strategies described in the previous section when applied to Adaboost ensembles. The first subsection describes the experimental protocol that has been followed in the experiments. Next, we evaluate the utility of the static pruning techniques described in Section III-A to obtain good approximate solutions to the problem of identifying sub-ensembles with good generalization properties. Furthermore, some of these techniques are compared with an exhaustive search strategy which finds optimal sub-ensembles defined in terms of the objective in (2). We also investigate here the combination of both pruning techniques, i.e., static and dynamic, with the aim of further reducing the number of classifiers queried to obtain the final decision of the ensemble. Finally, we analyze the computational cost of static pruning techniques and the gains obtained in the speed of classification when double pruning is applied.

A. Experiment Setup

In this subsection, we describe the general experimental setup carried out on 22 classification problems extracted from the UCI repository [18] to investigate the utility of combining static with dynamic pruning techniques toward reducing the number of classifiers queried by the ensemble. Table I shows the characteristics of the classification problems used. Most of the problems included in our selection are binary classification problems. However, we also considered some multiclass classification problems, see *Waveform*, *Wine* and *New-thyroid* or *Satellite*. Finally, three of the problems considered are synthetic, see *Twonorm*, *Ringnorm*, and *Waveform*.

The same protocol is used in all our experiments. Namely, the results reported are averages over 10 independent realizations of 10-fold cross validation. The details of the protocol are as follows.

1) For each problem and each train and test partition, Adaboost ensembles of 201 pruned CART trees [57] are generated. The standard settings for the generation of the CART trees are used. That is, the splits are obtained using the Gini criterion and the 1-SE pruning rule is applied. Majority voting is used to output the ensemble decision. We specifically employ an odd number of predictors to avoid ties in the voting process of binary problems. In nonbinary problems ties are broken by recursively removing the last vote until a single class gets more votes.
2) We report the performance of the complete ensembles on the test set.
3) We dynamically prune the complete ensemble at classification time using SIBP and $\alpha = 99\%$. The corresponding prediction performance on the test set and

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

TABLE I
CHARACTERISTICS OF THE DATASETS USED IN THE EXPERIMENTS

Dataset	Instances	Attributes	Classes
Australian	690	14	2
Breast	699	10	2
Diabetes	768	8	2
German	1,000	20	2
Heart	270	13	2
Hepatitis	155	19	2
Horse-colic	368	27	2
Ionosphere	351	34	2
Labor	57	16	2
Liver	345	7	2
Magic04	19020	10	2
New-thyroid	215	5	3
Ringnorm	-	20	2
Satellite	6435	36	6
Sonar	208	60	2
Spam	4601	57	2
Tic-tac-toe	958	9	2
Twonorm	-	20	2
Vehicle	846	18	4
Votes	435	16	2
Waveform	-	21	3
Wine	178	13	3

the average number classifiers used for prediction are recorded. We report results for both the ordering of the original ensemble and for a random permutation of the ensemble of classifiers.

4) Then, we extract from each ensemble a sub-ensemble of 101 trees using different static pruning techniques described in Section III-C: SDP-pruning, ordered aggregation based on the G criteria (Greedy-G), UWA [49], individual contribution pruning [50], diversity regularized pruning [48], and MDM [21]. The training set is used as the validation set $\mathcal{Z}$. Then, we report the corresponding prediction performance on the test set for all the static pruning methods considered. The prediction performance of an early-stopping pruning strategy which only considers the first 101 trees of the Adaboost ensemble is also reported. Note that 101 trees in the pruned ensembles corresponds to a pruning rate of 50%. Such a pruning rate provides good results in our experiments in general. However, the results reported are not very sensitive to this parameter and preliminary experiments indicate that any pruning rate in the vicinity of 50% performs similarly.
5) Finally, we apply again SIBP to the best off-line pruned sub-ensembles and we record the corresponding number of trees used for prediction and the associated classification error. Note that, as described in Section III-C, in all these ensembles we employ majority voting for output combination instead of the weighted voting scheme of Adaboost.

B. Static Pruning Techniques

Table II displays the average error for all the static pruning strategies. The table shows in order the results for: adaboost ensembles composed of 201 classifiers (Adaboost-201), Adaboost composed of 101 trees (Early-Stopping), UWA, EPIC, DReg40, MDM, Greedy-G, and SDP. In the case of diversity regularized pruning we evaluated different values for ρ, i.e., 5, 10, 20, 40, and 60%, and observed that the best overall test error is obtained for $\rho = 40\%$. For each dataset in Table II, the best result is highlighted in boldface and results better than Adaboost-201 are underlined. The last row shows the average rank of each method on the 22 datasets considered.

To analyze the overall performance of the static pruning techniques, the statistical framework described in [58] was applied. Under this framework, a Friedman test is evaluated on the average ranks of the different methods on the classification problems considered. This test rejects the null hypothesis that, in the classification problems investigated there are no significant differences in performance among the methods compared (p-value below 5%). Then, a Nemenyi test is applied to determine whether the differences in average rank between pairs of methods are statistically significant. Fig. 2 shows these average ranks and the results of the Nemenyi test for the methods and datasets shown in Table II. In the figure, two methods are linked with a horizontal solid black line if their difference in average rank is not statistically significant (p-value above 5%). The critical distance ($CD = 2.24$) above which the differences in rank are considered significant is displayed at the top of the figure for reference.

The results shown in Table II and Fig. 2 indicate that only two pruning strategies, SDP and Greedy-G, achieve an average rank better than the complete Adaboost ensemble (not statistical difference). The EPIC pruning method achieves an average rank that is similar to Early-Stopping, but not better than the full Adaboost ensemble. DReg40, UWA, and MDM obtain results that are worse than those of Early-Stopping using the same number of classifiers, however, the differences are statistically significant only for DReg40.

DReg40, UWA, and MDM have shown clear improvements when applied to bagging ensembles [21], [48], [49]. However, in the context of pruning Adaboost ensembles, and for the datasets analyzed, their performance is rather poor and below one of the simplest pruning methods which is early-stopping. This behavior can be explained by the fact that Adaboost ensembles reach zero error in the training set after generating a few classifiers. The consequence is that pruning strategies based on error related measures will reach zero error after a very low number of iterations. After that point, these measures will fail to identify adequate candidate classifiers to be added to the sub-ensemble. The reason for this is that most of the remaining classifiers will not change the training set error of the resulting sub-ensemble, which is already optimal, i.e., equal to zero. A similar effect is described in [21] when applying different pruning strategies to bagging ensembles composed of un-pruned decision trees. These trees have a prediction error on the training set that is close to zero. Leaving out a validation set for pruning set would improve the selection performance of these methods at the expense of worsening the accuracy of Adaboost.

Fig. 1 shows the average test error with respect to the number of classifiers for the different static pruning techniques and for Adaboost for four representative datasets, i.e.,

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

TABLE II
TEST ERROR IN % OF ADABOOST ENSEMBLES AND ADABOOST PRUNED USING DIFFERENT STATIC PRUNING TECHNIQUES

Problem	Adaboost-201	ES-101	UWA	EPIC	DReg-40	MDM	Greedy-G	SDP
Australian	**13.0±3.7**	13.1±3.7	13.5±4.2	13.8±3.8	19.2±4.7	14.9±4.8	13.1±4.0	13.1±4.0
Breast	3.5±2.1	3.5±2.1	3.6±2.2	**3.2±2.0**	3.9±2.2	3.6±2.0	**3.2±2.1**	**3.2±2.1**
Diabetes	25.4±4.2	25.2±4.2	25.1±4.2	**24.8±3.6**	28.3±3.9	27.8±4.1	25.0±4.0	24.9±4.0
German	24.4±3.3	24.8±3.4	24.8±3.5	24.3±2.9	26.5±3.9	25.5±3.9	**24.0±3.3**	**24.0±3.3**
Heart	19.9±7.6	19.8±7.1	**19.3±7.0**	20.0±6.7	24.1±8.5	22.6±8.2	19.9±7.0	20.0±7.1
Hepatitis	**15.8±8.1**	16.7±7.3	**15.8±7.3**	18.7±7.0	24.0±8.9	16.2±7.6	16.3±7.5	16.3±7.5
Horse-colic	19.5±5.5	19.7±6.1	21.5±6.3	17.6±5.5	33.0±7.7	20.4±6.4	17.3±5.5	**17.1±5.5**
Ionosphere	**6.2±3.7**	6.4±3.8	6.8±3.9	6.3±3.6	8.7±4.4	7.0±3.9	**6.2±3.4**	**6.2±3.4**
Labor	6.5±9.5	**6.4±9.4**	7.5±10.2	9.8±11.7	9.3±10.3	7.5±10.1	6.7±9.7	6.7±9.7
Liver	**27.1±7.0**	27.8±7.0	27.5±6.9	28.2±6.1	30.0±7.8	30.4±6.8	27.5±6.8	27.6±6.9
Magic04	12.0±0.7	12.2±0.7	12.3±0.7	12.2±0.7	13.1±0.8	12.4±0.7	**11.8±0.6**	**11.8±0.7**
New-thyroid	4.7±4.1	4.6±3.9	4.7±4.0	**4.3±4.1**	**4.3±3.8**	4.6±4.0	4.7±4.1	4.7±4.1
Ringnorm	6.2±0.7	6.7±0.8	13.2±3.0	5.8±0.8	14.1±1.8	7.6±0.8	**5.7±0.8**	**5.7±0.8**
Satellite	7.6±0.9	7.7±0.9	8.0±1.0	**7.5±0.9**	8.0±1.0	7.9±0.9	**7.5±0.9**	**7.5±0.9**
Sonar	**13.0±8.2**	14.5±9.2	16.8±9.4	14.9±9.1	16.6±9.1	13.8±8.1	13.7±8.4	13.7±8.4
Spam	5.3±1.0	5.4±1.1	5.8±1.4	6.0±1.1	6.0±1.1	5.0±1.1	**4.8±1.0**	**4.8±1.1**
Tic-tac-toe	0.7±0.9	0.8±0.9	0.8±1.0	**0.6±0.9**	0.9±1.0	0.9±1.1	**0.6±0.8**	**0.6±0.8**
Twonorm	**4.1±0.3**	4.4±0.4	5.6±0.7	4.5±0.5	5.4±0.6	4.6±0.4	4.4±0.4	4.4±0.4
Vehicle	25.7±10.2	25.3±8.6	25.1±8.7	24.4±8.8	25.9±8.9	24.5±8.7	**24.0±8.8**	24.1±8.7
Votes	**4.3±2.8**	**4.3±2.9**	4.4±3.1	**4.3±3.1**	7.1±3.5	5.9±3.2	4.5±2.9	4.5±2.9
Waveform	**17.4±0.8**	17.7±0.8	18.3±1.1	17.8±0.9	18.3±0.9	17.8±0.8	17.7±0.8	17.7±0.8
Wine	3.1±4.0	3.3±4.2	3.0±3.9	3.4±4.1	3.2±4.1	**2.9±4.2**	3.3±4.1	3.2±4.1
avg.ranks	3.1	4.4	5.3	4.4	7.2	5.4	2.9	3.1

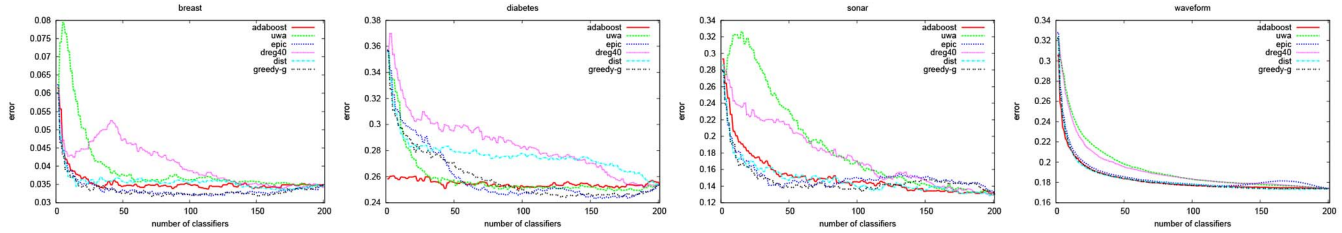


Fig. 1. Average generalization error with respect to the number of classifiers for Adaboost, and the ensembles pruned with UWA, EPIC, DReg40, MDM, and Greedy-G for *Breast*, *Diabetes*, *Sonar*, and *Waveform* datasets.

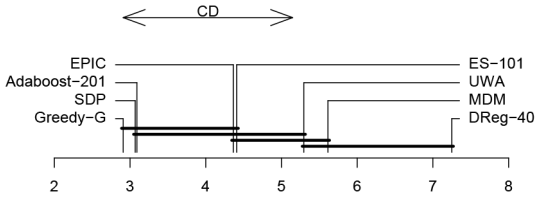


Fig. 2. Graphical representation of the results using a Nemenyi test on the average ranks of the different static pruning techniques for Adaboost in the classification problems investigated. Methods for which the differences in average rank are not statistically significant are linked with a solid black line. The critical distance (*CD* = 2.24) above which the differences in rank are considered significant is displayed at the top of the figure.

Breast, *Diabetes*, *Sonar*, and *Waveform*. In this figure it can be observed the rather poor performance of RReg40, UWA and MDM.We observe that the improvements are not remarkable in general. In the case of Greedy-G, there are some datasets where the average error is below the Adaboost for almost all ensemble sizes (see *Breast*). However, in some other datasets the improvements are observed only after around half of the classifiers have been included (see *Diabetes*). In *Sonar* the improvements are obtained for the first half of the ensemble. Finally, for *Waveform* the pruned sub-ensembles and the early-stopping sub-ensemble perform similarly for all ensemble sizes. Thus, we believe that these results justify the selection of 50% as the pruning rate in the static pruning techniques analyzed.

When we focus on the best performing pruning methods, i.e., SDP-pruning and Greedy-G, and we compare their performance with the performance of the complete ensemble, we observe typically small error rate improvements, albeit some exceptions (see the first and last two columns of Table II). For example, in *Sonar* or *Votes* the error rates of the resulting sub-ensembles are higher. By contrast, in *Horse-colic*, *Liver Australian* or *Diabetes* the error rates of the sub-ensembles are lower. When we compare SDP-pruning with Greedy-G there is no significant evidence to suggest a performance difference. Finally, we observe that both pruning techniques typically generate sub-ensembles with better prediction accuracy than the corresponding sub-ensembles obtained by the early-stopping strategy. Hence, in the rest of the paper we will exclusively consider these two pruning methods: SDP-pruning and Greedy-G.

1) Exhaustive Search Comparison: In this subsection, we further evaluate the best static pruning algorithms of the previous section in terms of the accuracy of the resulting approximate solutions. For this, we carry out an experiment comparing the performance of the solutions found by SDP-pruning and Greedy-G with the performance of the solutions

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

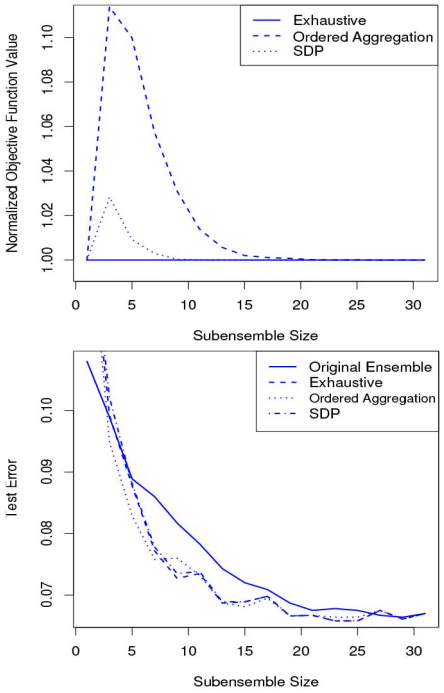


Fig. 3. Top: Average value of the objective function defined in (2) that is obtained by each pruning strategy, for each different sub-ensemble. The value of this function is normalized by the optimal value, i.e., the value obtained by the exhaustive search procedure. Bottom: Average test error of each sub-ensemble obtained by each different pruning strategy. The test error of the original ensemble is also displayed as a reference.

found by an exhaustive search procedure. The classification problem *Ionosphere*, extracted from the UCI repository [18], is considered for this purpose. The cost of the exhaustive search procedure grows exponentially with the ensemble size. Thus, we restrict the experiment to classification ensembles of 31 classifiers only and one 10-fold cross validation procedure. Then, different sub-ensembles of size $k = 1, 2, \ldots, 31$ are found by minimizing the objective function defined in (2), where the training set is used as the validation set $\mathcal{Z}$. We employ for this purpose the exhaustive search procedure and the approximate pruning strategies SDP-pruning and Greedy-G. For each one of the different sub-ensembles found, we evaluate the associated minimum in the objective function of (2) and the test error on the partition left out for validation. This experimental process is repeated 10 times and the estimates are averaged.

Fig. 3 (top) shows the average value of the objective function in (2) for each sub-ensemble size and each different static pruning strategy. The value of this function is normalized by the optimal value, i.e., the value that is obtained by the exhaustive search procedure. The picture confirms that the pruning methods actually select sub-ensembles that are very close to the optimal ones found by exhaustive search. The best optimization procedure is SDP-pruning. This technique obtains values that are at worst only 3% above the optimal ones. Greedy-G selects sub-ensembles whose associated objective function is at worst 10% above the optimal value. Fig. 3 (bottom) displays the average test error associated with the different sub-ensembles obtained by each pruning strategy. The test error of the original ensemble is also displayed as a reference. The picture shows that the errors of all the pruning strategies are very similar. For sub-ensembles of small size ($k < 5$) all the strategies over-fit the training data and select sub-ensembles with worse generalization performance than the sub-ensembles obtained by just early-stopping of the aggregation process of classifiers into the original ensemble. However, for sub-ensembles of intermediate size ($k > 5$), the opposite effect is observed. The test errors of the sub-ensembles provided by the different pruning strategies show a steeper descent. These errors are lower than the ones of the sub-ensembles obtained by early-stopping, although the error reduction is above the generalization error of the complete ensemble. The experiments of the previous subsection show that, if larger ensembles are available, it is possible to obtain sub-ensembles with better generalization properties than the complete ensemble.

C. Dynamic Pruning Techniques

Table III displays the results of the dynamic pruning strategies applied to the best performing static pruning methods: SDP and Greedy-G. The best performing method for each dataset has been highlighted in bold face. Moreover, when a method performs better than the complete Adaboost ensemble it has been underlined. Adaboost and the pruned ensembles results are repeated in this table for easier comparison. The first column of the table shows the average prediction error obtained by the full Adaboost ensembles (Adaboost-201). The following two columns (SIBP-Seq and SIBP-Perm) display the prediction error that is obtained when the dynamic pruning technique SIBP is applied, by processing the classifiers using either their original order in the ensemble or a random permutation, respectively. The next column (Early-Stopping) shows the prediction error of the early-stopping pruning strategy which queries only the first 101 classifiers in the Adaboost ensemble. The next two columns (Complete and SIBP-Seq) show the prediction error of statically pruned ensembles using Greedy-G. In particular, the first column (Complete) shows the prediction error that is obtained when only Greedy-G is applied. The second column (SIBP-Seq) shows the prediction error that is obtained when SIBP is applied in the classification phase to the ensemble pruned using Greedy-G. The following two columns show the same information when SDP-pruning is employed instead of Greedy-G.

Table IV displays, for each problem, the average number of trees queried by each pruning strategy. In the case of non dynamically pruned ensembles, we employ SIBP with $\alpha = 100\%$. That is, that if the number of votes for the majority class is larger than the votes for the second most voted class plus the number of remaining classifiers, then the voting process is stopped, since the final ensemble decision will not change. This procedure queries less predictors than the total number of predictors contained in the corresponding ensemble (i.e., 201 predictors or 101 predictors) without modifying the prediction of any instance.

Finally, Table V shows the average disagreement rates between the test predictions of Adaboost ensembles not

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

TABLE III
TEST ERROR IN % OF THE RESULTING ADABOOST ENSEMBLES (COMPLETE AND PRUNED, EITHER STATISTICALLY, DYNAMICALLY OR USING BOTH TECHNIQUES)

	Adaboost-201			Adaboost-101	Adaboost-101-GreedyG		Adaboost-101-SDP	
Problem	**Complete**	**SIBP-Seq**	**SIBP-Perm**	**Early-Stopping**	**Complete**	**SIBP-Seq**	**Complete**	**SIBP-Seq**
Australian	**13.0±3.7**	**13.0±3.7**	13.4±3.9	13.1±3.7	13.1±4.0	**13.0±4.0**	13.1±4.0	13.1±3.9
Breast	3.5±2.1	3.5±2.1	3.5±2.1	3.5±2.1	**3.2±2.1**	**3.2±2.1**	**3.2±2.1**	**3.2±2.1**
Diabetes	25.4±4.2	25.4±4.2	25.9±4.2	25.2±4.2	25.0±4.0	25.2±4.1	24.9±4.0	**24.8±4.0**
German	24.4±3.3	24.3±3.4	24.6±3.5	24.8±3.4	24.0±3.3	24.1±3.2	24.0±3.3	**23.9±3.5**
Heart	19.9±7.6	19.9±7.6	20.0±7.5	**19.8±7.1**	19.9±7.0	20.4±7.1	20.0±7.1	19.9±7.1
Hepatitis	**15.8±8.1**	15.9±8.1	16.1±8.6	16.7±7.3	16.3±7.5	16.6±7.7	16.3±7.5	16.4±7.5
Horse-colic	19.5±5.5	19.1±5.6	20.0±5.4	19.7±6.1	17.3±5.5	17.2±5.5	17.1±5.5	**16.9±5.4**
Ionosphere	**6.2±3.7**	**6.2±3.6**	6.3±3.6	6.4±3.8	**6.2±3.4**	**6.2±3.4**	**6.2±3.4**	**6.2±3.4**
Labor	6.5±9.5	6.5±9.5	6.8±9.6	**6.4±9.4**	6.7±9.7	7.6±10.7	6.7±9.7	6.7±9.7
Liver	**27.1±7.0**	**27.1±7.0**	27.8±6.8	27.8±7.0	27.5±6.8	28.5±6.9	27.6±6.9	27.5±6.9
Magic04	12.0±0.7	12.0±0.7	12.3±0.7	12.2±0.7	**11.8±0.7**	**11.8±0.7**	**11.8±0.7**	**11.8±0.7**
New-thyroid	4.7±4.1	4.7±4.2	**4.5±4.1**	4.6±4.0	4.7±4.1	4.7±4.1	4.7±4.1	4.7±4.1
Ringnorm	6.3±0.7	6.3±0.7	6.7±0.7	6.7±0.8	**5.7±0.7**	5.8±0.8	**5.7±0.7**	5.8±0.8
Satellite	7.6±0.9	7.6±0.9	7.6±1.0	7.7±1.0	**7.5±0.9**	**7.5±0.9**	**7.5±0.9**	**7.5±0.9**
Sonar	**13.0±8.2**	13.3±8.5	14.0±8.2	14.5±9.2	13.7±8.4	14.0±8.3	13.7±8.4	13.8±8.4
Spam	5.3±1.0	4.8±1.1	5.6±1.0	5.4±1.1	4.8±1.0	**4.6±1.1**	4.8±1.1	**4.6±1.1**
Tic-tac-toe	0.7±0.9	0.8±1.0	0.7±0.9	0.8±0.9	**0.6±0.8**	0.7±0.9	**0.6±0.8**	0.7±0.9
Twonorm	**4.1±0.4**	4.2±0.4	4.3±0.4	4.4±0.5	4.4±0.4	4.5±0.4	4.4±0.4	4.5±0.4
Vehicle	25.7±10.2	23.7±5.2	25.7±10.2	25.3±8.6	24.1±8.7	**22.5±3.9**	24.1±8.7	22.8±4.3
Votes	**4.3±2.8**	**4.3±2.9**	**4.3±2.8**	**4.3±2.9**	4.5±2.9	4.6±2.9	4.5±2.9	4.5±2.8
Waveform	**17.5±0.9**	**17.5±0.9**	17.7±0.9	17.8±0.9	17.7±0.8	17.8±0.8	17.7±0.8	17.8±0.8
Wine	3.2±4.1	**3.1±4.2**	3.2±4.1	3.3±4.2	3.2±4.1	3.2±4.1	3.2±4.1	3.2±4.1

TABLE IV
NUMBER OF TREES USED FOR CLASSIFICATION IN EACH ADABOOST ENSEMBLE (COMPLETE AND PRUNED, EITHER STATISTICALLY, DYNAMICALLY, OR USING BOTH TECHNIQUES)

	Adaboost-201			Adaboost-101-GreedyG		Adaboost-101-SDP	
Problem	**Complete**	**SIBP-Seq**	**SIBP-Perm**	**Complete**	**SIBP-Seq**	**Complete**	**SIBP-Seq**
Australian	152.4±3.2	44.3±6.9	47.1±6.5	71.5±1.5	24.2±3.0	71.7±1.5	26.0±3.3
Breast	122.7±2.6	14.9±3.2	17.0±3.1	56.6±1.0	9.8±1.5	57.2±1.1	10.2±1.5
Diabetes	167.3±3.2	73.5±10.4	73.8±8.6	77.2±1.4	35.7±3.4	77.3±1.6	35.7±3.6
German	166.4±2.3	71.0±8.0	72.0±6.3	79.4±1.2	38.3±3.1	79.0±1.2	37.5±3.0
Heart	158.7±4.1	56.5±11.7	59.3±10.2	74.4±2.2	29.8±4.9	74.2±2.2	30.1±5.0
Hepatitis	150.3±5.7	42.5±13.4	44.3±11.0	68.9±3.2	22.9±7.0	69.6±3.1	24.0±6.2
Horse-colic	165.8±4.7	61.2±12.7	71.1±12.8	75.1±2.2	29.2±5.5	75.6±2.1	30.4±4.7
Ionosphere	139.7±3.1	24.4±6.5	29.1±5.7	62.2±1.7	13.2±2.8	63.6±1.7	15.0±3.1
Labor	138.7±9.7	32.2±18.5	31.7±16.6	63.7±5.4	15.8±9.7	65.2±5.2	19.2±9.3
Liver	173.0±3.3	90.8±12.0	84.9±10.4	81.1±2.0	40.7±5.4	82.0±2.0	44.9±5.9
Magic04	152.6±1.9	39.6±2.7	48.3±2.6	70.6±0.3	23.1±0.6	70.2±0.4	22.9±0.9
New-thyroid	124.6±4.2	19.1±7.2	18.8±5.8	57.1±2.1	10.6±3.1	58.5±2.1	12.7±3.2
Ringnorm	152.2±1.2	46.1±2.4	47.2±1.7	71.4±0.7	24.9±1.2	72.2±0.6	26.8±1.1
Satellite	131.1±1.0	25.5±1.5	27.7±1.5	64.3±0.5	18.7±0.7	64.5±0.5	18.5±0.7
Sonar	160.2±4.5	61.5±13.9	60.7±11.8	76.3±2.4	32.3±5.9	76.9±2.3	35.4±5.8
Spam	144.5±3.0	21.6±2.8	40.4±3.5	65.2±0.8	14.0±0.9	65.6±0.9	14.4±1.0
Tic-tac-toe	133.2±1.5	16.3±2.4	20.1±2.4	61.7±0.7	11.4±1.1	62.9±0.8	12.5±1.2
Twonorm	143.3±0.8	35.6±1.6	35.5±1.0	68.8±0.3	22.5±0.5	69.3±0.3	23.5±0.7
Vehicle	153.1±8.0	59.3±8.9	57.6±12.6	73.2±2.5	31.6±3.3	73.6±2.5	33.6±3.4
Votes	130.8±3.3	18.7±5.2	22.3±4.6	58.2±1.3	10.3±2.1	59.4±1.4	12.1±2.1
Waveform	158.7±1.9	61.5±4.1	60.7±3.0	75.4±0.9	32.5±1.4	76.0±0.9	34.3±1.5
Wine	126.8±4.2	18.1±6.7	19.9±5.9	57.8±1.9	9.6±2.2	58.8±1.9	12.0±3.1

dynamically pruned and the corresponding Adaboost ensembles dynamically pruned using SIBP with $\alpha = 99\%$. The disagreement rate measures the percentage of examples that get a different class label prediction when SIBP in applied with respect to when SIBP is not applied. Recall that these rates should theoretically remain below 1% for $\alpha = 99\%$.

The figures displayed in Table IV (see the first two columns) show that when SIBP is applied to the Adaboost ensemble with the original ordering of the classifiers (SIBP-Seq), there is an ample reduction in the number of queries (a factor reduction between 2 and 8), and consequently, also the time needed for classification. Furthermore, Table III indicates that the corresponding average error rate is often similar and sometimes even better than the average error rate of the complete ensemble (see the first two columns of Table III). When the ordering of the classifiers in the ensemble is not the original one, but a random permutation (SIBP-Perm), the dynamic strategy produces slightly worse prediction results than those of the complete ensemble (see the first and third columns of Table III). Moreover, the disagreement rates obtained by the permuted ensemble typically exceed the imposed bound of 1% (see second column of Table V). On the contrary, the disagreement rates of the ensemble which follows the original ensemble order are generally below the value of 1%.

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

TABLE V
DISAGREEMENT RATES BETWEEN NOT DYNAMICALLY PRUNED ADABOOST ENSEMBLES AND DYNAMICALLY PRUNED WITH $\alpha = 99\%$

	Adaboost-201		Adaboost-101 GreedyG	Adaboost-101 SDP
Problem	**SIBP-Seq**	**SIBP-Perm**	**SIBP-Seq**	**SIBP-Seq**
Australian	0.0±0.2	1.1±1.3	0.4±0.7	0.2±0.5
Breast	0.0±0.0	0.2±0.6	0.0±0.2	0.0±0.2
Diabetes	0.2±0.5	2.1±1.7	0.9±1.1	0.3±0.7
German	0.3±0.6	2.1±1.5	1.0±0.9	0.5±0.7
Heart	0.0±0.0	1.6±2.2	0.5±1.4	0.2±0.8
Hepatitis	0.1±0.9	1.0±2.6	0.3±1.4	0.2±1.1
Horse-colic	0.9±1.5	1.7±2.3	0.2±0.9	0.4±1.0
Ionosphere	0.0±0.3	0.3±0.8	0.0±0.3	0.0±0.3
Labor	0.0±0.0	0.3±2.3	0.9±3.8	0.0±0.0
Liver	0.1±0.6	2.6±2.6	1.4±2.0	0.2±0.7
Magic04	0.2±0.1	1.0±0.2	0.5±0.1	0.2±0.1
New-thyroid	0.0±0.5	0.4±1.3	0.0±0.5	0.0±0.0
Ringnorm	0.2±0.1	0.9±0.1	0.4±0.1	0.2±0.1
Satellite	0.1±0.1	0.2±0.2	0.2±0.2	0.1±0.2
Sonar	0.3±1.3	1.5±2.5	0.4±1.3	0.4±1.4
Spam	1.0±0.5	0.7±0.5	0.5±0.4	0.6±0.4
Tic-tac-toe	0.0±0.2	0.0±0.2	0.0±0.2	0.1±0.2
Twonorm	0.2±0.1	0.6±0.1	0.3±0.1	0.2±0.1
Vehicle	3.0±9.0	0.8±1.0	2.1±7.5	1.7±6.6
Votes	0.0±0.3	0.1±0.5	0.1±0.4	0.0±0.2
Waveform	0.1±0.1	1.0±0.2	0.5±0.2	0.2±0.1
Wine	0.1±0.6	0.2±1.0	0.0±0.0	0.0±0.0

These results show that when SIBP-Seq is applied to Adaboost ensembles, the prediction performance is similar to the original ensemble performance. However, the time needed for classification is significantly reduced. In the problems analyzed, only 12%–60% of the original trees need to be queried on average.

The fact that randomly permuting the classifiers in SIBP leads to a deterioration of the prediction performance and to an increase of the disagreement rates is an unexpected result. To shed light on this matter, we carry out extra experiments. In particular, Fig. 4 displays the Adaboost ensemble error as a function of the classifiers queried, when following the original ensemble order and when following a random permutation of the classifiers. We only provide results for *Diabetes*. Nevertheless, similar results are observed in the other classification problems investigated. Namely, when the order of the classifiers is altered, the error curve is generally above the one corresponding to the original Adaboost ensemble. They only coincide for $T = 201$, when both ensembles are composed of the same classifiers. This suggests a difference between the distribution of votes of a classifier retrieved randomly from the Adaboost ensemble and a classifier retrieved randomly from the first classifiers generated by Adaboost. The actual differences between these probability distributions are displayed in Fig. 5 for *Diabetes*. These plots show the probability distribution of predicting class label 1, p_1, by an ensemble member. The plot in the left column shows the probability distribution when all classifiers of the ensemble are considered. The plot in the middle shows the same distribution but when only the 20 first classifiers of the randomly permuted ensemble are considered. Finally, the plot on the right shows the same distribution when the first 20 classifiers of the original Adaboost sequence are considered. We observe that the distributions on the left and the middle are very similar. This is an expected result since the plots in the middle (20 randomly picked classifiers) correspond to a random subsampling of the classifiers

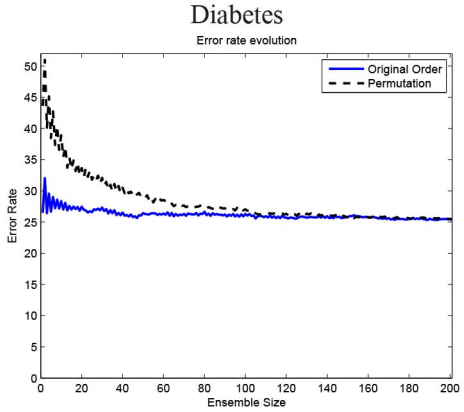


Fig. 4. Test error rate for Adaboost ensembles as a function of the number of classifiers queried. We provide results when following the original order provided by Adaboost and when following a random permutation of the ensemble classifiers.

corresponding to the plots on the left (full ensemble). An important observation is that these probability distributions are not uniform. Recall that SIBP assumes that the prior probability distribution of votes is uniform. The plots show that the vote distributions associated with randomly ordered ensembles are significantly more peaked at $p_1 = 1/2$ than the distributions obtained when considering only the first classifiers in the original Adaboost sequence. Thus, using a uniform prior distribution for random ordered Adaboost ensembles results in unrealistic and optimistic stopping rules. By contrast, when the original sequence of classifiers generated by Adaboost is considered, the first classifiers in the sequence have a more uniform voting distribution which provides better results, both in terms of the expected disagreement rates and error values.

D. Results for Double Pruning

Finally, we test the performance of double pruning when applied to Adaboost ensembles. Double pruning consists of first summarizing the original Adaboost ensemble using SDP-pruning and Greedy-G, as explained earlier, to then apply the on-line instance-based pruning algorithm. Table III shows that when SIBP is applied to statically pruned Adaboost ensembles (either by SDP-pruning or Greedy-G) the resulting prediction performance is equivalent to the one of the sub-ensembles which do not consider SIBP. Furthermore, the double pruning strategy provides sub-ensembles that query fewer classifiers. Specifically, between $\approx$ 14% (for *Votes* and *New-Thyroid* dataset) and $\approx$ 33% (for *Liver* dataset) of the classifiers queried by the Adaboost ensemble when it is dynamically pruned using $\alpha = 100\%$ (see respectively the fifth and seventh columns of Table IV). The percentage of queried classifiers when using double pruning with respect to the original 201 classifiers is below 10% for some datasets and no bigger than 30%. This confirms the utility of the on-line pruning phase to further reduce the average number of classifiers queried during classification. We also observe that typically double pruning queries a lower number of trees when Greedy-G is used instead of SDP-pruning. Finally, we remark that the application of this double pruning scheme does not lead to an increase of the time

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

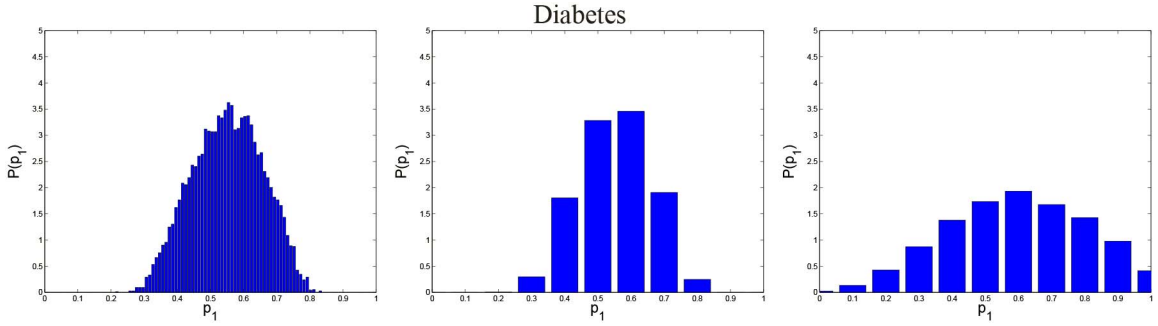


Fig. 5. Probability distributions for p_1, i.e., the probability mass that an ensemble classifier predicts class label 1, for *Diabetes*. The left column shows the prior distribution associated with all the classifiers in the ensemble. The center column displays the same distributions for the randomly ordered ensembles using the first 20 classifiers. The right column shows the distributions for the first 20 classifiers of the original sequence of Adaboost.

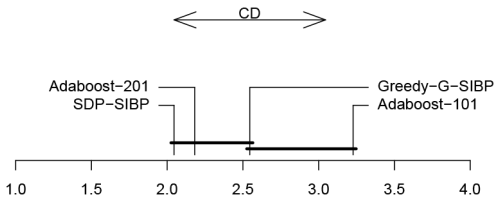


Fig. 6. Graphical representation of the results using a Nemenyi test on the average ranks of the different ensemble methods in the classification problems investigated.

TABLE VI
THEORETICAL TIME COMPLEXITY AND AVERAGE EXECUTION TIME IN SECONDS FOR THE DIFFERENT PRUNING STRATEGIES FOR THE DIABETES DATASET AS A FUNCTION OF T, THE ENSEMBLE SIZE

Pruning Method	Theoretical Complexity	Average execution time for T= 50	100	200	400	800	1600
Greedy-G	$\mathcal{O}(T^2)$	0.022	0.076	0.252	0.93	3.67	15.07
SDP	$\mathcal{O}(T^3)$	0.113	0.704	3.567	27.48	277.20	2840

needed for classification. In particular, the use of static techniques is off-line and needs to be done only once to obtain the corresponding sub-ensemble from the original Adaboost ensemble. Similarly, the computational cost of SIBP is negligible since all the required values to efficiently implement this dynamic pruning technique can be precomputed as described in [22].

The overall performance of double pruning is analyzed using the statistical framework described in [58]. Fig. 6 shows these average ranks and the results of the Nemenyi test for the double pruning strategies that use static SDP-pruning (SDP-SIBP) and Greedy-G (Greedy-G-SIBP), the complete Adaboost ensemble (Ada201), and with early-stopping (Ada101). The plot shows that there is not enough statistical evidence to indicate a performance difference between double pruning based on either SDP-pruning or Greedy-G and the complete Adaboost ensemble (Ada201), even though double pruning uses only a small fraction of the original ensemble classifiers. By constrast, early-stopping performs significantly worse than the complete Adaboost ensemble. When we compare double pruning based on SDP-pruning with double pruning based on Greedy-G we observed that the last method performs slightly worse and despite of the fact that Greedy-G performs slightly better than SDP when only static pruning is considered (see average ranks in Table II). This is due to the slight deterioration of double pruning applied to Greedy-G in a few datasets. However, the differences are not statistically significant. Finally, the differences in performance of double pruning based on SDP-pruning with respect to early-stopping (Ada101) are statistically significant. Double pruning based on Greedy-G performs better than early-stopping (Ada101), but the differences are not statistically significant. These are very good results considering the fact that double pruning uses significantly less classifiers for prediction than early-stopping (Ada101). In summary, these experiments illustrate the benefits of classifier selection to improve the prediction speed in Adaboost ensembles without deteriorating the prediction accuracy.

E. Time Analysis

In this last subsection, the computational costs of the different pruning strategies are further analyzed for both the training and the testing phases. The training phase involves the computational complexity only of the static pruning techniques. The test phase involves the time gains obtained for the speed of classification. This is measured in terms of the number of instances that the ensembles can classify per second in both full ensemble and when static and dynamic pruning techniques are applied.

Table VI displays the theoretical complexity of the static pruning algorithms, SDP-pruning, and Greedy-G, in the train phase in terms of the ensemble size, T. Recall that to the costs of these algorithms we have to add the cost of computing the matrix $\mathbf{G}$, which is $\mathcal{O}(T^2 \cdot N)$, where N is the number of training instances, since the training set is typically used as the validation set $\mathcal{Z}$. We note that Greedy-G has a quadratic cost on T, while the cost of the SDP-pruning approach is cubic in T. We further investigate the dependence on T of the execution time of the different pruning strategies by carrying out a series of experiments on the *Diabetes* dataset. Table VI displays the average execution time of each algorithm for initial ensembles of 50, 100, 200, 400, 800, and 1600 classifiers. The times reported are averages over 10 realizations on a QuadCore Q6660 processor. The results displayed in this table show that the best results correspond to Greedy-G.

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
THROUGHPUT (IN CLASSIFIED INSTANCES PER SECOND) FOR THE FULL ADABOOST ENSEMBLE AND THE DOUBLE PRUNED ENSEMBLE USING THE GREEDY-G

Dataset	Adaboost-201	Greedy-GSIBP	Increment Factor
Australian	22149	109802	5
Breast	24617	269884	11
Diabetes	20304	59076	2.9
German	18728	59189	3.2
Heart	25295	98181	3.9
Hepatitis	27385	137900	5
Horse-colic	26802	110411	4.1
Ionosphere	23111	205744	8.9
Labor	37025	205776	5.6
Liver	21849	59257	2.7
Magic04	19520	34062	1.7
New-thyroid	28828	252347	8.8
Ringnorm	20166	104384	5.2
Sonar	27034	108333	4
Spam	11286	114795	10.2
Tic-tac-toe	15357	284063	18.5
Twonorm	25504	155279	6.1
Vehicle	13168	68881	5.2
Votes	39164	351657	9
Waveform	24736	94393	3.8
Wine	30526	290849	9.5

Table VII shows, for each dataset investigated, the average number of instances classified per second (throughput) for the full Adaboost ensemble and the double pruned ensemble with Greedy-G-SIBP. The average improvement factor obtained in terms of the number of instances classified per second is also displayed. The figures have been computed using a Q6660 @ 2.40 GHz processor. We note that improvement factor is remarkable with respect to the full Adaboost ensemble. In particular, the throughput increment varies from ≈ 2 times, in the case of *Magic-04*, up to 18 times, in the case of *Tic-tac-Toe*. This speed increment is expected to apply both when the ensemble prediction is computed sequentially or in a parallel environment. For example, when the predictions need to be computed in parallel, the double pruned ensemble can be distributed across different machines that process classification tasks in parallel. These improvements are specially useful when important amounts of data needs to be processed rapidly such as in online applications or classification in data-streams.

V. CONCLUSION

In this paper, we have proposed to combine static and dynamic (instance-based) pruning strategies to improve the classification speed and to reduce storage requirements of Adaboost ensembles without significantly decreasing their prediction performance. The goal of static ensemble pruning algorithms (see SDP-pruning, and Greedy-G) is to select a sub-ensemble whose predictive accuracy is equivalent or better than the original complete ensembles. Besides potential improvements in the generalization performance of the ensemble, static pruning reduces both storage requirements and the computational cost of the prediction process. SIBP is a dynamic strategy whose goal is to reduce the number of classifiers that need to be queried for a particular instance so that the partial ensemble prediction coincides with the prediction of the complete ensemble. SIBP does not reduce the storage requirements. The complete ensemble needs to be accessible at all times in case that, for a particular instance, all the classifiers are queried to reach a final prediction. However, SIBP speeds up the prediction process further. This is a significant advantage, especially in online applications.

The results of the empirical investigation carried out indicate that pruning Adaboost ensembles using static techniques is a difficult task. In particular, we have observed that many state-of-the-art ensemble pruning techniques have difficulties when pruning these type of ensembles. The reason for this is that most of these techniques use an estimate of the training error of the sub-ensemble to guide the process of classifier selection. Adaboost ensembles quickly achieve very low errors on the training set. Thus, this search criterion is not very useful to prune this type of ensembles. The only observed effective criterion that can be used to prune an Adaboost ensemble with static pruning techniques is the G measure introduced in [20]. This measure can be minimized either via SDP-pruning or via a greedy search (Greedy-G) to identify sub-ensembles of a specified size whose generalization capacity is close to being optimal. The near-optimal sub-ensembles identified by SDP-pruning are closer to the optimal ones than those selected by Greedy-G. However, in most of the problems investigated both static pruning techniques lead to similar improvements in the prediction accuracy with respect to the original ensemble. Nonetheless, these improvements are not statistically significant. When we consider the computational costs of the pruning phase, Greedy-G should be preferred to SDP-pruning. Specifically, the cost of SDP-pruning is cubic in the size of the original ensemble while the cost of Greedy-G is only quadratic.

Another interesting result of our investigation is that SIBP can be successfully applied to Adaboost ensembles despite the fact the classifiers in the ensemble are generated sequentially. Specifically, the gradual emphasis on examples that are difficult to classify by previous classifiers in the ensemble introduces dependencies among the individual classifiers. Because of these dependencies the statistical analysis of SIBP presented in [22] is not directly applicable to Adaboost ensembles. Nonetheless, the dependencies among the classifiers introduce a bias that turns out to be beneficial: the estimated confidence in the stability of the class prediction is generally lower than the actual confidence. Since the querying process is halted when the estimated confidence is above a certain threshold, SIBP tends to be conservative and rarely leads to over-pruning. Even with this conservative bias, SIBP reduces the number of classifiers queried in Adaboost ensembles up to 12%–60% of the original classifiers (a speed-up factor of classification between 2 and 8), depending on the problem considered.

A final contribution of this paper is the design of a double pruning strategy for Adaboost ensembles: dynamic pruning (in particular, SIBP) is applied to sub-ensembles that have been identified with static pruning techniques (either SDP-pruning or Greedy-G). The result is a double pruning algorithm, which provides better or similar predictive accuracy and reduces memory requirements and prediction times with respect to the original Adaboost ensemble. More precisely, only the

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

classifiers that are selected by the static pruning strategy need to be stored in memory. Most of these are only seldom queried by the dynamic pruning strategy SIBP. The overall result is a significant speed-up of the prediction process with little or no deterioration (sometimes even improvement) of the predictive accuracy with respect to the original ensemble. In particular, the speed-up is between 2 and 18 times with respect to the original Adaboost ensemble.

REFERENCES

[1] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[2] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[3] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th ICML*, 1996, pp. 148–156.

[4] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[5] G. Martínez-Muñoz and A. Suárez, "Switching class labels to generate classification ensembles," *Pattern Recognit.*, vol. 38, no. 10, pp. 1483–1494, 2005.

[6] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006.

[7] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[8] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. MCS*, Cagliari, Italy, 2000, pp. 1–15.

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

[10] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. New York, NY, USA: Wiley, 2004.

[11] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd ICML*, New York, NY, USA, 2006, pp. 161–168.

[12] Y. Koren and R. M. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook.* New York, NY, USA: Springer, 2011, pp. 145–186.

[13] T. Gneiting and A. E. Raftery, "Weather forecasting with ensemble methods," *Science*, vol. 310, no. 5746, pp. 248–249, Oct. 2005.

[14] J. Shotton *et al.,* "Real-time human pose recognition in parts from single depth images," in *Proc. 24th IEEE Conf. CVPR*, Providence, RI, USA, Jun. 2011, pp. 1297–1304.

[15] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[16] D. Marbach, C. Mattiussi, and D. Floreano, "Combining multiple results of a reverse engineering algorithm: Application to the DREAM five gene network challenge," *Ann. N.Y. Acad. Sci.*, vol. 1158, no. 1, pp. 102–113, Mar. 2009.

[17] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decis. Support Syst.*, vol. 50, pp. 602–613, Feb. 2011.

[18] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository* [Online]. Available: http://www.ics.uci.edu/ mlearn/ MLRepository.html

[19] M. Hall *et al.,* "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[20] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *J. Mach. Learn. Res.*, vol. 7, pp. 1315–1338, Jul. 2006.

[21] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 245–259, Feb. 2009.

[22] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, "Statistical instance-based pruning in ensembles of independent classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 364–369, Feb. 2009.

[23] D. Hernández-Lobato, G. Martńez-Muñoz, and A. Suárez, "Inference on the prediction of ensembles of infinite size," *Pattern Recognit.*, vol. 44, no. 7, pp. 1426–1434, 2011.

[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. 2nd ECCL*, 1995, pp. 23–37.

[25] Y. Freund and R. Schapire, "A short introduction to boosting," *Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.

[26] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Cambridge, MA, USA: MIT Press, 2012.

[27] V. Soto, G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "A double pruning algorithm for classification ensembles," in *Proc. MCS*, Cairo, Egypt, 2010, pp. 104–113.

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

[29] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, "On the independence of the individual predictions in parallel randomized ensembles," in *Proc. 20th ESANN*, Bruges, Belgium, Apr. 2012, pp. 233–238.

[30] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, pp. 197–227, Jul. 1990.

[31] L. Breiman, "Arcing classifiers," *Ann. Statist.*, vol. 26, no. 3, pp. 801–849, 1998.

[32] H. Drucker, "Improving regressors using boosting techniques," in *Proc. 14th ICML*, San Mateo, CA, USA, 1997.

[33] R. Avnimelech and N. Intrator, "Boosting regression estimators," *Neural Comput.*, vol. 11, no. 2, pp. 499–520, 1999 [Online]. Available: http://citeseer.ist.psu.edu/avnimelech99boosting.html

[34] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[35] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," in *Advanced Lectures on Machine Learning*, S. Mendelson and A. J. Smola, Eds. Berlin, Germany: Springer, 2003, pp. 118–183.

[36] J. J. Rodríguez and J. Maudes, "Boosting recombined weak classifiers," *Pattern Recognit. Lett.*, vol. 29, no. 8, pp. 1049–1059, 2008.

[37] O. Pujol, "Boosted geometry-based ensembles," in *Proc. MCS*, Cairo, Egypt, 2010, pp. 195–204.

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

[39] T. Hastie, R. Tibshirani, and J. H. Friedman. (Aug. 2001). *The Elements of Statistical Learning*. Springer [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0387952845

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

[41] J. R. Quinlan, "Bagging, boosting, and C4.5," in *Proc. 13th AAAI*, 1996, pp. 725–730.

[42] L. Breiman, "Bias, variance, and arcing classifiers," Statist. Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. 460, 1996.

[43] P. Domingos, "A unified bias-variance decomposition and its applications," in *Proc. 17th ICML*, San Francisco, CA, USA, 2000, pp. 231–238.

[44] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proc. Int. Conf. Neural Netw.*, Washington, DC, USA, 1996, pp. 90–95.

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

[46] G. Tsoumakas, I. Partalas, and I. Vlahavas, "An ensemble pruning primer," in *Applications of Supervised and Unsupervised Ensemble Methods*, O. Okun and G. Valentini, Eds. Berlin, Germany: Springer, 2009, pp. 1–13 [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03999-7_1

[47] G. Martínez-Muñoz and A. Suárez, "Aggregation ordering in bagging," in *Proc. IASTED*, Calgary, AB, Canada, 2004, pp. 258–263.

[48] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Machine Learning and Knowledge Discovery in Databases*, P. Flach, T. Bie, and N. Cristianini, Eds. Berlin, Germany: Springer, 2012, pp. 330–345.

[49] I. Partalas, G. Tsoumakas, and I. Vlahavas, "An ensemble uncertainty aware measure for directed hill climbing ensemble pruning," *Mach. Learn.*, vol. 81, no. 3, pp. 257–282, 2010.

[50] Z. Lu, X. Wu, X. Zhu, and J. Bongard, "Ensemble pruning via individual contribution ordering," in *Proc. KDD*, Washington, DC, USA, 2010, pp. 871–880.

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.

[51] W. Fan, F. Chu, H. Wang, and P. S. Yu, "Pruning and dynamic scheduling of cost-sensitive ensembles," in *Proc. 18th Nat. Conf. Artif. Intell.*, Menlo Park, CA, USA, 2002, pp. 146–151.
[52] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. KDD*, New York, NY, USA, 2003, pp. 226–235.
[53] J. Basilico, M. Munson, T. Kolda, K. Dixon, and W. Kegelmeyer, "COMET: A recipe for learning and using large ensembles on massive data," in *Proc. IEEE 11th ICDM*, Washington, DC, USA, Dec. 2011, pp. 41–50.
[54] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "Statistical instance-based ensemble pruning for multi-class problems," in *Proc. 19th ICANN*, Limassol, Cyprus, 2009, pp. 90–99.
[55] F. Markatopoulou, G. Tsoumakas, and L. Vlahavas, "Instance-based ensemble pruning via multi-label classification," in *Proc. 22nd IEEE ICTAI*, Arras, France, Oct. 2010, pp. 401–408.
[56] A. Hung-Ren Ko, R. Sabourin, and A. de Souza Britto, "K-nearest oracle for dynamic ensemble selection," in *Proc. 9th ICDAR*, Parana, Brazil, Sep. 2007, pp. 422–426.
[57] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.
[58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

Víctor Soto received the B.Sc. degree in computer engineering and mathematics, and the M.Sc. degree in computer science from the Universidad Autiónoma de Madrid (UAM), Cantoblanco, Spain, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree from the Computer Science Department, Columbia University, New York, NY, USA.

From 2009 to 2011, he researched on double pruning techniques for ensemble classifiers as a part of the M.Sc. thesis. At the same time, he interned at Telefonica Research, Madrid, Spain, where he focused on applying machine learning techniques for cell-phone user modeling. His current research interests include speech processing and keyword search for low-resource languages.

Mr. Soto received the Comunidad de Madrid Award for Outstanding Students, the UAM Graduate Student Grant, and the "la Caixa" Fellowship for postgraduate studies in the USA.



Sergio García-Moratilla received the university and the M.Sc. degrees in computer science from the Universidad Autónoma de Madrid, Cantoblanco, Spain.

He is currently with the industry at Quantitative Risk Research, a technological company specializing in risk measurement and computational finance.



Gonzalo Martínez-Muñoz received the university degree in physics, and the Ph.D. degree in computer science from the Universidad Autónoma de Madrid (UAM), Cantoblanco, Spain, in 1995 and 2006, respectively.

From 1996 to 2002, he was with the industry. Until 2008, he was an Interim Assistant Professor at the Computer Science Department of the UAM. During 2008/2009, he was a Fulbright Post-Doctoral Researcher at Oregon State University, Corvallis, OR, USA, in Prof. T. G. Dietterich's group. He is currently a Professor with the Computer Science Department at UAM. His current research interests include machine learning, computer vision, pattern recognition, neural networks, decision trees, and ensemble learning.



Daniel Hernández-Lobato received the B.Sc. degree in computer science from the Universidad Autónoma de Madrid (UAM), Cantoblanco, Spain, in 2004, and the M.Sc. and Ph.D. degrees, in 2007 and 2009, respectively.

Currently, he is a Lecturer of Computer Science with Escuela Politécnica Superior, UAM, since 2011, where he researches model-based machine learning and pattern recognition following a Bayesian perspective. His current research interests include ensemble methods, Gaussian processes, and sparse linear models, with particular emphasis on deterministic methods for fast approximate Bayesian inference. He has also been a Post-Doctoral with the Université Catholique de Lovain, Louvain-La-Neuve, Belgium, where he has collaborated with the Machine Learning Group.

Dr. Hernández-Lobato received a special prize for his dissertation upon graduation.



Alberto Suárez (F'06) received the degree of Licenciado in chemistry from the Universidad Autónoma de Madrid (UAM), Cantoblanco, Spain, and the Ph.D. degree in physical chemistry from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1988 and 1993, respectively.

Currently, he is a Professor with the Computer Science Department, UAM, after holding the post-doctoral positions at Stanford University, Stanford, CA, USA, the Université Libre de Bruxelles, Brussel, Belgium, and the Katholieke Universiteit Leuven, Leuven, Belgium. He has researched relaxation theory in condensed media, stochastic and thermodynamic theories of non-equilibrium systems, lattice-gas automata, and decision-tree induction. His current research interests include machine learning, computational finance, and information processing in the presence of noise.

Authorized licensed use limited to: Reykjavik University. Downloaded on May 22,2024 at 12:57:19 UTC from IEEE Xplore. Restrictions apply.