Using Boosting to Prune Bagging Ensembles

Gonzalo Martínez-Muñoz * and Alberto Suárez

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

Abstract

Boosting is used to determine the order in which classifiers are aggregated in a bagging ensemble. Early stopping in the aggregation of the classifiers in the ordered bagging ensemble allows the identification of subensembles that require less memory for storage, classify faster and can improve the generalization accuracy of the original bagging ensemble. In all the classification problems investigated pruned ensembles with 20 % of the original classifiers show statistically significant improvements over bagging. In problems where boosting is superior to bagging, these improvements are not sufficient to reach the accuracy of the corresponding boosting ensembles. However, ensemble pruning preserves the performance of bagging in noisy classification tasks, where boosting often has larger generalization errors. Therefore, pruned bagging should generally be preferred to complete bagging and, if no information about the level of noise is available, it is a robust alternative to AdaBoost.

Key words: Machine learning, Decision Trees, Bagging, Boosting, Ensembles, Ensemble pruning

1 Introduction

Numerous experimental studies show that pooling the decisions of classifiers in an ensemble usually improves the generalization performance of weak learners (Breiman (1996a, 1998, 2001); Dietterich (2000); Freund and Schapire (1995); Martínez-Muñoz and Suárez (2005); Webb (2000)). Important shortcomings of ensemble methods are the loss of speed in classification with respect to the base classifier and the growth in storage needs with increasing numbers of inducers. In order to remedy these drawbacks, one can try to retain only

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

those classifiers that are essential to solve the classification task at hand and eliminate those whose contribution is redundant. Ensemble pruning reduces the storage needs, speeds up the classification process and has the potential of improving the classification accuracy of the original ensembles.

Several strategies have been proposed to reduce the number of units in classifier ensembles. In (Domingos (1997)) the ensemble is replaced by a single classifier trained to emulate the behavior of the combined classifiers. Other techniques select a subset of classifiers from the full ensemble (Zhou et al. (2002); Zhou and Tang (2003); Martínez-Muñoz and Suárez (2004a)). The problem of selecting the best combination of classifiers from an ensemble has been shown to be NP-complete (Tamon and Xiang (2000)). In (Zhou et al. (2002); Zhou and Tang (2003)) the problem of finding a globally optimal subset of classifiers is solved approximately by means of a genetic algorithm. Using a different strategy, Prodromidis and Stolfo (2001) construct a tree based on the outputs of the individual classifiers. This tree is then pruned and the classifiers whose outputs are no longer considered in the pruned tree are removed from the ensemble. Demir and Alpaydin (2005) introduce a utility factor that takes into account the cost of classifying new instances in order select the optimal subset of classifiers. In the work of Giacinto and Roli (2001) and Bakker and Heskes (2003), clustering techniques are applied to identify groups of classifiers in the ensemble that give similar classifications. A pruned ensemble is then generated by retaining a single representative per cluster. Margineantu and Dietterich (1997) propose some interesting heuristics based on measures of diversity and performance for the selection of a small subset of representative classifiers in AdaBoost ensembles without a severe deterioration of the classification performance. Some of the heuristics introduced by Margineantu and Dietterich (1997) are used in (Martínez-Muñoz and Suárez (2004a)), where ensembles of increasing size are built by incorporating at each step from a pool of bagging classifiers the one that maximizes a quantity strongly correlated with the generalization error of the ensemble. The final subensemble is selected by stopping aggregation at a prescribed ensemble size. In (Banfield et al. (2005)) the original complete ensemble is pruned (or *thinned*, using the term proposed by the authors) by sequential backward selection: classifiers that do not improve the classification performance are progressively removed from the ensemble. The metrics used to identify the redundant classifiers are based on ensemble accuracy and ensemble diversity.

Most of the pruning strategies introduced in the literature, and, in particular, all methods that are used to reduce the size of boosting ensembles, construct smaller ensembles at the expense of a limited loss in classification accuracy. In this work we design a procedure to construct pruned bagging ensembles that outperform full bagging in all classification tasks investigated, and boosting in noisy classification tasks. The ensemble pruning method designed uses the weighted training error defined in boosting to determine the order in which classifiers are aggregated in an initially randomly ordered bagging ensemble. The algorithm proceeds iteratively by updating the training example weights as in boosting: the weights of training examples correctly (incorrectly) classified by the last classifier incorporated into the ensemble are decreased (increased) according the AdaBoost prescription (Freund and Schapire (1995)). The classifier that minimizes the weighted training error is then incorporated into the ensemble. Early stopping in the aggregation process allows to select subensembles that outperform bagging and retain bagging's resilience to noise in the class labels of the examples.

Section 2 introduces boosting-based ordered bagging and two heuristics that determine when to stop aggregating classifiers in the ordered ensemble. The performance of the pruned ensembles on several UCI (Blake and Merz (1998)) and synthetic datasets is investigated in Section 3. Finally, Section 4 summarizes the results of the present investigation.

2 Boosting-based ordered Bagging

Consider a collection of N labeled examples $L = \{(\boldsymbol{x}_i, y_i),\ i = 1, \dots N\}$. Each example in L is composed of a vector of attributes, $\boldsymbol{x}$, and a class label y, which takes discrete values in a finite space $\phi \equiv \{1, 2, \dots, Y\}$. In a classification problem, the objective is to learn a map from the attribute space to the discrete space of class labels ϕ by induction from a set of labeled examples (the training set). The goal is to construct classifiers that perform well in previously unseen data, using information only from the training set.

Bagging (Breiman (1996a)) is an ensemble method in which different classifiers of the same type are induced by a bootstrap method. Each classifier in a bagging ensemble is constructed using the same learning algorithm on different bootstrap samples of the original training data. A given bootstrap sample is generated by performing N_{train} extractions with replacement from the original training set. The final decision of the ensemble is obtained by a voting procedure that combines the individual decisions of the inducers that compose the ensemble with equal weights:

$$C^*(\boldsymbol{x}) = \underset{y}{argmax} \sum_{t:C_t(\boldsymbol{x})=y}^{T} 1 \tag{1}$$

where T is the number of classifiers of the ensemble and C_t is the $t^{th.}$ classifier.

Boosting (Freund and Schapire (1995)) is a sequential algorithm in which each new inducer is built taking into account the performance of the previously generated classifiers. Figure 1 displays the pseudocode for the AdaBoost algorithm using reweighting. In this ensemble method, classifier t is induced by using a fixed learning algorithm with different sets of weights ($\boldsymbol{w} = \{w_1, w_2, \ldots, w_{N_{train}}\}$) for the examples in the training set. The first classifier in a boosting ensemble is built by setting all the weights of the examples to $1/N_{train}$ (i.e. all examples have initially the same importance). At each iteration a classifier is built with training example weights updated so that misclassified training examples become more relevant. In this way the subsequent classifiers focus on examples that are difficult to classify. Thus, classifiers $\{C_{t+1}, t = 1, 2, \ldots, T-1\}$ are induced using L_{train} and $\boldsymbol{w}$, where the weights are modified by incrementing (decreasing) the weights of the examples incorrectly (correctly) classified by C_t. This procedure is repeated until T classifiers are generated or until a classifier achieves zero error or an accuracy below 50%, in which cases the weight updating rule fails and the boosting algorithm stops. The final decision of the ensemble is obtained by weighted voting of its members

$$C^*(\boldsymbol{x}) = \underset{y}{argmax} \sum_{t:C_t(\boldsymbol{x})=y}^{T} log(1/\beta_t) \tag{2}$$

where $\beta_t = \epsilon_t/(1-\epsilon_t)$ and ϵ_t is the weighted training error of the $t^{th.}$ classifier.

To prevent early stopping when a classifier has $\epsilon > 0.5$ or $\epsilon = 0$, one common variant of AdaBoost is to replace the original training set by a bootstrap sample with all weights set to $1/N_{train}$, and then to continue the boosting process (Bauer and Kohavi (1999)). If $\epsilon = 0$, the classifier is incorporated into the ensemble with $\beta = 10^{-10}$ instead of $\beta = 0$ which would assign an infinite weight to the vote of that classifier (Webb (2000)). These modifications allow the boosting process to always generate the specified number of inducers and, in general, increase the accuracy of AdaBoost (Bauer and Kohavi (1999); Webb (2000)). Under these conditions, comparisons with other ensemble methods that always produce the desired number of classifiers (such as Bagging) are fairer.

In this work, we propose to modify the aggregation order in a bagging ensemble. To guide the aggregation process the weighting scheme proposed in AdaBoost to compute the training error is used. If the aggregation process is halted before all classifiers generated are incorporated into the ensemble, one can obtain pruned subensembles, which, while being smaller, show better classification accuracy than the full bagging ensembles.

Figure 2 presents the pseudocode of the ensemble aggregation ordering algorithm. The first step is to generate a pool of classifiers. In our implementation a bagging ensemble of size T is constructed from the training set (instructions 1-5). Instead of bagging, other parallel ensemble building methods can in principle be used (Breiman (2001); Dietterich (2000); Martínez-Muñoz and

Input:

training set L composed of N examples

number of classifiers T

Output:

$C^*(\boldsymbol{x}) = \underset{y}{argmax} \sum_{t:C_t(\boldsymbol{x})=y} log(1/\beta_t)$

```
set all instance weights w to 1/N
for t=1 to T {
   C_t = TrainClassifier(L, w)
   ε_t = WeightedError(C_t, L, w)
   if (ε_t > 0.5) {
       discard C_t
       break
   }
   β_t = ε_t/(1 − ε_t)
   for j=1 to N {
       if (C_t(x_j) ≠ y_j) then w_j = w_j/2ε_t
       else w_j = w_j/2(1 − ε_t)
   }
}
return C
```

Fig. 1. Pseudocode for the AdaBoost algorithm

Suárez (2004b)). The second phase (instructions 6-20) is similar to boosting. However, instead of generating an inducer from the weighted training dataset at each iteration, the classifier with the lowest weighted training error is selected from the pool of classifiers generated by bagging (step 8). To avoid early stopping, if no classifier has a weighted training error below 50%, the weights of the examples are reset to $1/N_{train}$ and the boosting process continues. In contrast to regular boosting, when the selected classifier has zero training error the process continues. Note that bagging rarely generates zero training error classifiers and that, if they exist, these classifiers would be selected in the first iterations of the algorithm. The process continues until the desired number of classifiers is reached (parameter U in Fig. 2). The final decision is

Input:

training set L composed of N examples

number of classifiers T

number of classifiers to select U

Output:

Ensemble D^* of U classifiers:

$D^*(\boldsymbol{x}) = \underset{y}{argmax} \sum_{u:D_u(\boldsymbol{x})=y} 1$

1. for t=1 to T {
2. L_{bt} = BootstrapSample(L)
3. C_t = TrainClassifier(L_{bt})
4.   Add C_t to pool C
5. }
6. set all instance weights $\boldsymbol{w}$ to $1/N$
7. for u=1 to U {

    //Gets the learner with lowest weighted error

8. D_u = SelectBest(C, L, $\boldsymbol{w}$)
9. ϵ_u = WeightedError(D_u, L, $\boldsymbol{w}$)
10.   if ($\epsilon_u > 0.5$) {
11.   reset all instance weights $\mathbf{w}$ to $1/N$
12.   goto 8
13.   }
14.   Add D_u to pool D
15.   Extract D_u from pool C
16.   for j=1 to N {
17.   if ($D_u(\boldsymbol{x}_j) \neq y_j$) then $w_j = w_j/2\epsilon_u$
18.   else $w_j = w_j/2(1-\epsilon_u)$
19.   }
20. }
21. return D

Fig. 2. Pseudocode for the Boosting-based ordered bagging algorithm

performed by either unweighted (as in bagging) or weighted (as in boosting) voting among the selected classifiers. If weighted voting is used, the weights for each classifier are given by the prescription proposed in AdaBoost.

In contrast to the usual versions of bagging or boosting, where the training error typically decreases as the number of classifiers in the subensemble is increased, the ensembles ordered by the procedure described above have training error curves that exhibit a minimum for intermediate numbers of classifiers (see Figs. 3 to 5). More interestingly, the test error curves also have a minimum for subensembles of intermediate size. In all but the smallest subensembles the test error lies below the asymptotic bagging error, making it easy to select a subensemble that outperforms the original bagging ensemble. In all experiments performed, the minimum in the test error curve corresponds to larger subensembles than the minimum in the train error curve. This poses some difficulty in the selection of the subensemble that has the best generalization error. For small ensembles (around 20 trees), Zhou and Tang (2003) report that the best results are obtained for pruning values that are $\approx 60\%$ (i.e. reducing the ensemble to ≈ 8 trees). In larger ensembles, the optimum amount of ensemble pruning seems to be between 70%-85% (Martínez-Muñoz and Suárez (2004a)). That is, for ensembles of 200 trees 30-60 classifiers should be selected. Note that the minima in the test error curves are fairly broad, which means that the improvements in classification accuracy are not very sensitive to the particular pruning heuristic employed. In our studies two simple rules for the choice of subensemble are proposed. A first rule is to use a fixed pruning rate that selects the subensemble containing only 20% of the original bagging ensemble classifiers. A second proposal is to stop aggregation at the first boosting stopping point, i.e., when the weighted training error reaches 0.5.

Pruned ensembles built by the procedure described in this work take advantage of complementary features of boosting and bagging. In particular, boosting ensembles grow by incorporating classifiers trained to focus on examples that are misclassified by the current ensemble. If these examples are the ones that are relevant to define the actual classification boundary, boosting builds ensembles which generally have a better classification performance than bagging. By contrast, if the examples which are progressively given more weight by boosting are outliers or have been labeled incorrectly, then the distortion introduced by boosting in the original problem may lead to a significant deterioration in the classification performance of the ensemble (Rätsch et al. (2001); Dietterich (2000)). In bagging new classifiers are generated without modifying the original distribution of training examples. The source of variation among classifiers is the fluctuations in the bootstrap samples. Bagging is therefore weaker than boosting, since it does not actively focus on examples that are difficult to classify, but also more resilient to noise in the class labels of the training examples (Dietterich (2000)). The scheme proposed by boosting to update the weights of the training examples is used in our algorithm to select which classifier from the complete ensemble generated by bagging is aggregated next. In this manner, the algorithm follows the strategy of boosting, but it always incorporates classifiers that solve the original classification problem, not modified versions of it. In this regard, the experiments carried out by Dietterich (2000) to compare the effects of noise in the classification performance of bagging, boosting and randomization ensembles are illustrative. In these experiments the original training data is modified by flipping the class label of a number of randomly selected examples. In datasets where the class labels of 20% of the training examples are modified, the corrupted examples reach ≈50% of the total weight after only a few iterations of boosting. This percentage is maintained subsequently in the rest of the learning process. As a consequence of this spuriously high sensitivity to the corrupted examples the generalization performance of the boosted ensemble in this type of noisy classification problems is rather poor. The probability of obtaining such a weight distribution from a uniform bootstrap sampling can be calculated using the binomial distribution and is very small. In summary, the ordering procedure can be seen as a regularized version boosting, where the level of overfitting to noise is limited because the algorithm avoids assigning too much weight to a few examples, which may be incorrectly labeled. Typically, these regularized versions of boosting perform better than ordinary boosting in noisy datasets (Rätsch et al. (2001)).

3 Experiments

To assess the usefulness of the ordering algorithm and the pruning rules proposed in the previous section, experiments in 18 datasets have been carried out. These include 16 real-world classification problems from the UCI repository (Blake and Merz (1998)) and two synthetic sets (*Waveform* and *Twonorm*) described in (Breiman et al. (1984); Breiman (1996b)). The characteristics of these sets are presented in Table 1. This selection includes datasets with different characteristics and from a variety of fields.

For each dataset included 100 executions were carried out involving the following steps:

(1) Generate a stratified random partition between training and testing sets. The sizes of these partitions are given in Table 1. For the synthetic sets a random sampling was performed instead.
(2) Execute the algorithm defined in Fig. 2 with $T = 200$, using CART as the base learner (Breiman et al. (1984)). The classification errors of boosting ensembles with $T = 200$ built using the variant of AdaBoost presented in (Webb (2000)) are reported and will be used as a benchmark.

Table 1
Datasets used in the experiments

Dataset	**Train**	**Test**	**Atts.**	**Classes**
Audio	140	86	69	24
Australian	500	190	14	2
Breast W.	500	199	9	2
Diabetes	468	300	8	2
German	600	400	20	2
Heart	170	100	13	2
Horse-Colic	244	124	21	2
Ionosphere	234	117	34	2
Labor	37	20	16	2
New-thyroid	140	75	5	3
Segment	210	2100	19	7
Sonar	138	70	60	2
Tic-tac-toe	600	358	9	2
Twonorm	300	5000	20	2
Vehicle	564	282	18	4
Vowel	600	390	10	11
Waveform	300	5000	21	3
Wine	100	78	13	3

(3) The generalization capability of the different ensembles considered is estimated using the testing set. To avoid ties in binary classification problems, we report results only for subensembles with an odd number of classifiers.
(4) Boosting assigns progressively higher weights to examples that are more difficult to classify. For this reason, classifiers generated later in the boosting process solve problems that are increasingly more different from the original one. In order to compensate for this effect, classifiers generated later in boosting are given a lower voting weight in the final ensemble classification. To investigate whether this correction is necessary in ordered bagging ensembles, we report the classification accuracy using both weighted and unweighted voting. Voting weights for the combined ensemble are calculated according to the prescription given in AdaBoost.

Figures 3 to 5 display the dependence of the classification error rate on the number of classifiers included in the ensembles, estimated on both the training

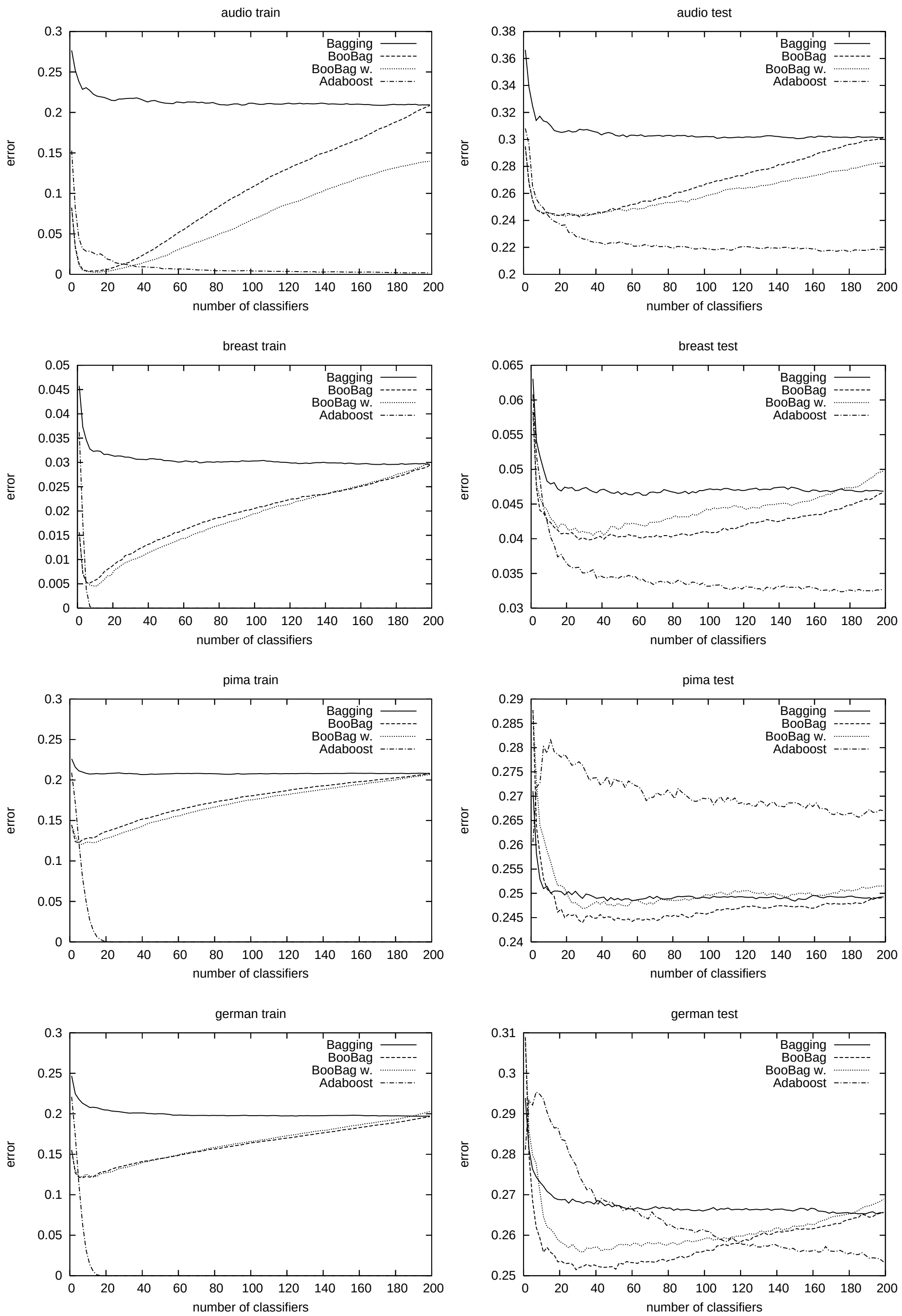

Fig. 3. Train and test error curves for Bagging (*solid line*), ordered bagging with unweighted (*long trait line*) and weighted (*short trait line*) voting and AdaBoost (*dotted line*)

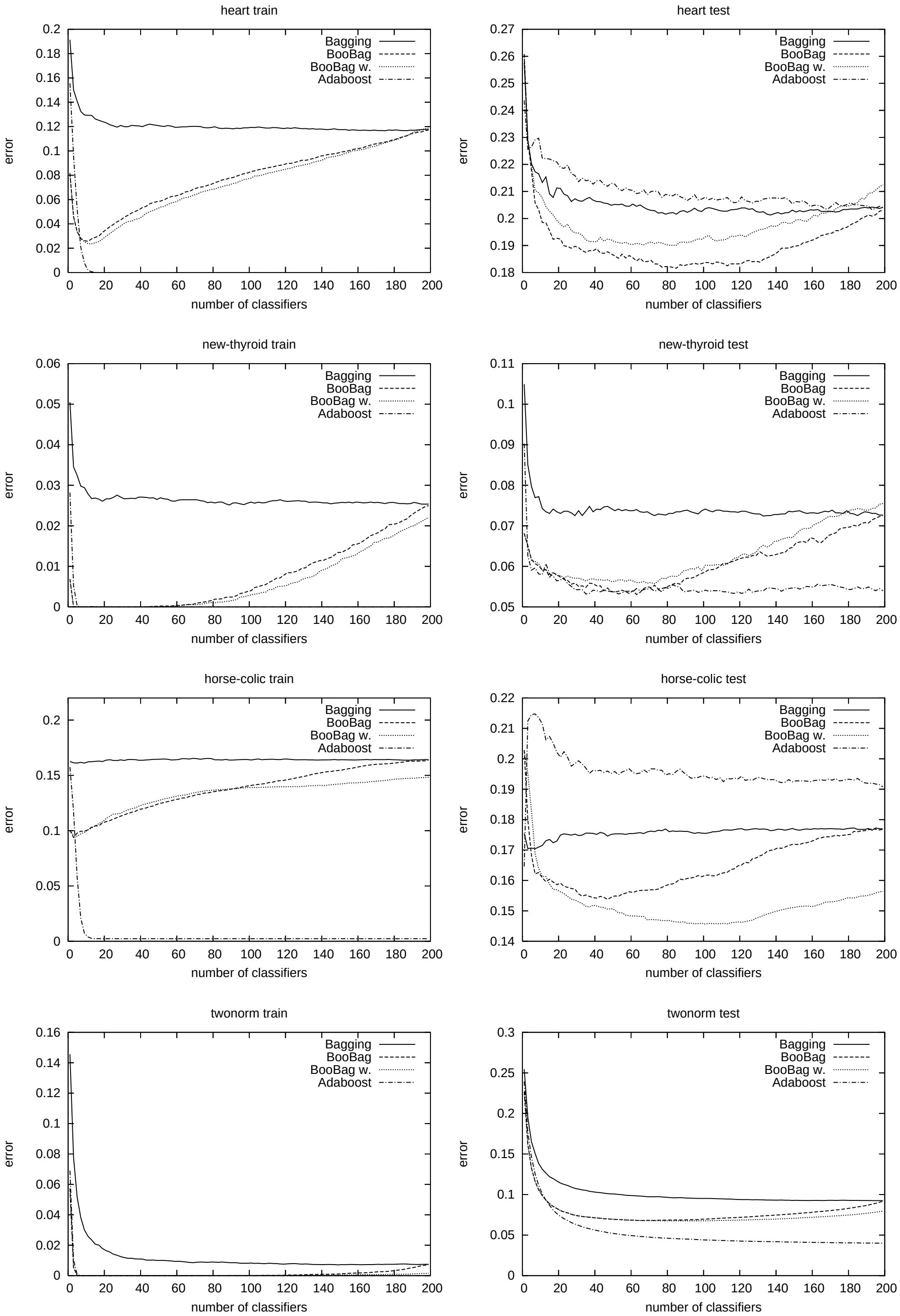

Fig. 4. Train and test error curves for Bagging (*solid line*), ordered bagging with unweighted (*long trait line*) and weighted (*short trait line*) voting and AdaBoost (*dotted line*)

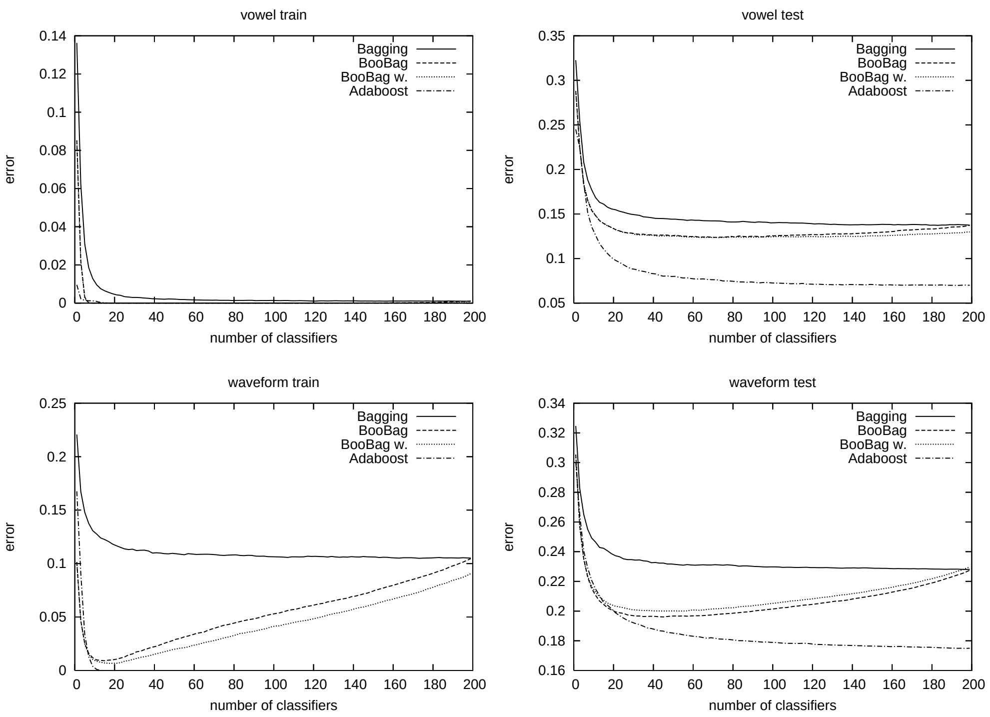

Fig. 5. Train and test error curves for Bagging (*solid line*), ordered bagging with unweighted (*long trait line*) and weighted (*short trait line*) voting and AdaBoost (*dotted line*)

and the test sets. Error curves are given for 10 datasets of the 18 datasets investigated. The plots displayed are representative of all cases investigated. As anticipated, in bagging with random ordering the error decreases monotonically as the number of classifiers in the ensemble grows. This decrease levels off at an asymptotically constant error rate. In contrast to this monotonic behavior, error curves in ordered bagging ensembles exhibit a typical shape where the error initially decreases with the number of classifiers, reaches a minimum, and eventually rises. In the case of unweighted voting, the final error coincides (as it should) with the error level of full bagging. This characteristic shape is reproduced in both the train and test error curves. However, minima in the error curve for the training set appear earlier and are narrower than those in the test set. In datasets where boosting improves the results of bagging ensembles, the test error curves for ordered bagging and boosting run very close for the first few iterations. In noisy sets, such as *Pima Indian Diabetes, Horse-colic*, while AdaBoost increases the test error, ordered bagging still exhibits a broad minimum in the test error curve.

To prune the bagging ensemble two different rules are used

(1) From the ordered bagging ensemble, a subensemble composed of a fixed number of classifiers is selected. In particular, the first 20% classifiers (exactly 41 classifiers) are kept.

Table 2
Errors for the different ensembles (200 trees). The average best result for each problem is highlighted in bold type. The second best results are underlined.

Dataset	Full ensembles			Pruned ensembles				
	Boosting (200 trees)	Bagging (200 trees)		Fixed size (41 trees)		First boosting stop		
		unw.	w.	unw.	w.	unw.	w.	#trees
Audio	**21.8**	30.2	28.3	24.6	24.5	<u>24.1</u>	24.2	29.6
Australian	**13.7**	14.5	14.4	<u>13.8</u>	14.1	14.7	16.3	5.1
Breast W.	**3.3**	4.7	5.0	<u>4.0</u>	4.1	4.2	4.3	15.0
Diabetes	26.6	24.9	25.2	**24.5**	<u>24.8</u>	25.3	26.2	7.7
German	<u>25.4</u>	26.6	26.9	**25.2**	25.7	26.2	27.8	7.6
Heart	20.5	20.4	21.2	**18.9**	<u>19.1</u>	19.6	20.6	18.1
Horse-colic	19.1	17.7	15.7	<u>15.4</u>	**15.2**	16.7	18.9	5.0
Ionosphere	**6.6**	9.3	9.5	<u>7.5</u>	7.8	7.7	7.9	17.1
Labor	**9.7**	14.4	16.1	10.5	10.7	10.6	<u>9.8</u>	132.0
New-thyroid	**5.4**	7.3	7.5	<u>5.5</u>	5.7	**5.4**	<u>5.5</u>	87.4
Segment	**6.2**	9.7	9.6	<u>7.7</u>	<u>7.7</u>	<u>7.7</u>	<u>7.7</u>	43.8
Sonar	**14.7**	24.7	23.8	20.4	20.1	20.5	<u>19.9</u>	65.1
Tic-tac-toe	**0.9**	2.7	2.5	2.1	2.1	2.1	<u>2.0</u>	67.6
Twonorm	**4.0**	9.3	8.0	7.1	7.1	7.1	<u>6.7</u>	123.6
Vehicle	**23.5**	29.6	29.2	26.3	<u>26.2</u>	26.4	26.5	23.8
Vowel	**7.0**	13.7	13.0	12.6	12.6	12.8	<u>12.5</u>	133.3
Waveform	**17.5**	22.8	23.0	<u>19.6</u>	20.0	19.7	20.2	33.7
Wine	**4.0**	6.5	5.7	4.5	4.6	4.6	<u>4.4</u>	133.6

(2) From the ordered bagging ensemble, aggregate classifiers until the first classifier with a weighted training error above 0.5 is found (first boosting stop).

The results of the experiments are summarized in Table 2. For each dataset the averaged generalization accuracies over the 100 executions are shown for AdaBoost (first column), full bagging with both unweighted and weighted voting (second and third columns), pruned subensembles of fixed size (41 trees) with both unweighted and weighted voting (fourth and fifth columns) and pruned ensembles using the first boosting stopping rule with unweighted

Table 3
Errors for the different ensembles (100 trees). The average best result for each problem is highlighted in bold type. The second best results are underlined.

Dataset	Full ensembles: Boosting (100 trees)	Full ensembles: Bagging (100 trees)		Pruned ensembles: Fixed size (21 trees)		Pruned ensembles: First boosting stop		
		unw.	w.	unw.	w.	unw.	w.	#trees
Audio	**21.9**	30.2	28.2	24.9	24.9	24.4	24.5	16.4
Australian	14.2	14.5	14.6	**14.0**	14.3	14.6	16.5	4.8
Breast W.	**3.3**	4.7	4.9	4.2	4.3	4.4	4.5	9.5
Diabetes	27.0	24.9	25.1	**24.5**	24.9	25.0	26.1	6.8
German	26.1	26.6	26.8	**25.4**	25.9	26.3	27.6	6.7
Heart	20.8	20.3	21.0	**19.3**	19.9	20.0	20.9	12.3
Horse-colic	19.4	17.5	16.2	15.7	**15.5**	16.9	19.0	4.3
Ionosphere	**6.5**	9.4	9.5	7.8	7.8	8.1	7.8	10.6
Labor	10.1	14.6	13.9	10.4	10.6	10.4	**9.8**	66.4
New-thyroid	**5.4**	7.5	7.6	**5.4**	**5.4**	5.5	**5.4**	44.0
Segment	**6.3**	9.8	9.7	8.1	8.0	8.0	8.0	24.5
Sonar	**16.0**	24.6	24.1	21.1	20.8	21.6	20.8	36.0
Tic-tac-toe	**1.1**	2.7	2.6	2.4	2.4	2.3	2.3	36.1
Twonorm	**4.4**	9.5	8.5	8.2	8.3	7.6	7.3	64.9
Vehicle	**23.7**	29.5	29.1	27.0	27.0	26.9	27.2	16.1
Vowel	**7.3**	14.0	13.4	13.6	13.5	13.2	12.8	68.6
Waveform	**17.9**	23.0	23.2	20.3	20.9	20.3	20.9	21.9
Wine	**4.2**	6.6	5.9	4.6	4.6	4.8	4.4	66.5

and weighted voting (sixth and seventh columns). The last column displays the average number of trees of the first boosting stopping point for each dataset. For each dataset the best result is highlighted in bold face and the second best is underlined.

In most datasets, boosting ensembles have a better generalization performance than either bagging or boosting-based pruned bagging. However, boosting is less robust and its generalization performance is poor in noisy problems (*Diabetes, Horse-colic*). Pruning the ordered ensembles generally improves the classification performance. In particular, subensembles containing 41 classifiers selected from the ordered bagging ensemble obtain consistently better generalization accuracies than the corresponding complete bagging ensembles. For all datasets, the differences between unweighted 41 trees and full bagging, considered individually, are statistically significant at a 99.5% confidence level using the paired two tailed Student's t-test. This is also true for the weighted 41 trees subensemble except for the *Pima Indian Diabetes* dataset. The improvements are especially large in *Wine* (30.8% generalization error reduction), *Labor Negotiations* (27.1%), *New Thyroid* (24.7%) and *Twonorm* (23.7 %). Using the classifier weights given by boosting in the voting procedure does not seem to have a marked effect: the error rates exhibit minor differences (both positive and negative) between weighted and unweighted voting schemes in the investigated datasets.

Subensembles generated by using the first boosting stop rule have a lower testing error than full bagging in most of the classification problems investigated. The exceptions are *Australian Credit, Pima Indian Diabetes, German Credit* and *Horse Colic*. Nonetheless, for these four sets the error differences are not statistically significant at a 99.5% confidence level, while the reduction in size is substantial: the selected subensembles retain an average of 5.1, 7.7, 7.6 and 5.0, trees, respectively, which is below 5% of the initial pool of 200 trees. For the remaining datasets the first stopping point occurs for larger subensembles, ranging from 15.0 trees of *Breast W.* to 133.6 trees of *Wine*, and the improvements of classification performance are also larger. Note that in the datasets where less than 5% of trees are selected, higher error reductions are obtained for larger ensembles (compare for instance with the classification errors of subensembles with 41 trees, using unweighted voting). This indicates that this stopping rule stops prematurely for some datasets. If the votes of the classifiers in the subensembles are combined using the weights proposed by the boosting algorithm, the results obtained show small variations whose sign depends on the particular classification problem (see Table 2). The differences between unweighted first boosting stop and full bagging are statistically significant at a 99.5% confidence level for 14 datasets (all except *Australian*, *Pima Indian Diabetes*, *German Credit* and *Heart*). For the weighted first boosting stop subensembles the statistically significant differences are favorable in 12 datasets, unfavorable in 4 sets and not significant in 2.

To evaluate how the number of classifiers in the original bagging ensemble affects the performance of the pruned ensembles we carry out a second batch of experiments. For these experiments the randomly ordered ensembles were re-evaluated using the first 100 trees. The ordering algorithm is then applied to this smaller pool of classifiers. Table 3 shows the average errors for 100 trees and 100 executions for AdaBoost (first column), full bagging with both unweighted and weighted voting (second and third columns), pruned subensembles of fixed size (21 trees) with both unweighted and weighted voting (fourth and fifth columns) and pruned ensembles using the first boosting stopping rule with unweighted and weighted voting (sixth and seventh columns). The number of classifiers in the pruned subensemble selected from the ordered ensembles of size 100 are shown in the last column. In the datasets investigated, a bagging ensemble with 100 trees seems to be large enough to achieve the best possible classification performance of bagging. Slight improvements are observed for some sets (*New-thyroid*, *Twonorm*, *Vowel*, *Waveform*) when using the larger ensemble but also small error increases (*Heart* and *Horse-colic*). For pruned ensembles, comparing the results in Tables 2 and 3 show that there are small improvements in classification accuracy for the larger ensembles for most of the investigated datasets, at the expense of using pruned ensembles with approximately twice as many classifiers.

3.1 Effects of noise in the performance of the pruned subensembles.

In order to investigate the performance of the pruned ensembles in noisy datasets, we carry out a series of experiments similar to those conducted by (Dietterich (2000)). In these experiments classifiers are built using as training data corrupted versions of the initial training set. For each experiment, a fixed percentage of training examples is selected at random and their class labels switched. The percentage of examples whose class label is modified increases for each experiment in the series. Experiments on the *Waveform* problem dataset are carried out using the same 100 samples of training and test data generated in the previous set of experiments. Class labels of training and test examples are modified with a probability 0.0, 0.05, 0.1 and 0.2, respectively. This procedure generates noisy training and test datasets containing on average 0%, 5%, 10% and 20% of examples labeled differently from the original *Waveform* problem.

The average error on the test set for standard boosting, bagging and boosting-based pruned bagging on the modified noisy versions of the *Waveform* problem are shown in table 4. As noted in previous experiments (Dietterich (2000)), the performance of AdaBoost strongly deteriorates with increasing levels of noise in class labels. Bagging and pruned bagging subensembles composed of 20 % of the original trees are fairly resilient to this type of noise. In contrast, the performance of pruned subensembles that use the first boosting stop to halt aggregation deteriorates significantly (although not as much as boosting) as the level of noise increases. The amount of pruning determined by the first boosting stop rule is excessive in these problems and therefore the resulting ensembles are too small. Nonetheless, this can be acceptable if we are interested in obtaining very small ensembles at the expense of a limited loss in classification accuracy.

According to these results, pruned bagging ensembles with 20 % of the original

Table 4
Average test error in *Waveform* with modified class labels. The best average result is highlighted in bold type. The second best averaged results are underlined.

Noise	Full ensembles AdaBoost (200 trees)	Full ensembles Bagging (200 trees)		Pruned ensembles Fixed size (41 trees)		First boosting stop		
		unw.	w.	unw.	w.	unw.	w.	#trees
0%	**17.5**	22.8	23.0	<u>19.6</u>	20.0	19.7	20.2	33.7
5%	<u>24.2</u>	26.5	27.2	**23.9**	24.6	25.1	26.6	11.7
10%	31.3	30.2	31.2	**27.6**	<u>28.6</u>	29.1	31.3	9.8
20%	42.3	37.5	38.6	**35.6**	<u>36.4</u>	37.6	40.2	8.8

trees consistently improve the generalization error of the complete bagging ensemble in both noiseless and noisy datasets. Although they are inferior to boosting in the noiseless experiment, pruned ensembles outperform boosting in classification tasks even in classification problems with relatively low levels of noise in the class labels (5 % in our experiments on the *Waveform* problem).

4 Conclusions

This article presents a novel algorithm for pruning classifier ensembles that uses the reweighting scheme for the training examples proposed in AdaBoost to modify the (originally random) aggregation ordering of bagging. For the investigated datasets, if we plot the error rate versus the number of classifiers included in the ordered ensemble, we observe that both training and test error curves exhibit a minimum for partially aggregated subensembles. This minimum corresponds to pruned ensembles whose misclassification rates are below the classification error of the full bagging ensemble. These observations suggest that selecting a subset of classifiers increases the classification speed, lowers the memory requirements and can improve the classification performance of the original ensemble.

The minimum observed in the ensemble test error curves is fairly broad, which implies that it is easy to improve the results of bagging by early stopping in the aggregation process in the ordered bagging ensemble. We propose two heuristics to halt aggregation. The first ensemble pruning method consists in selecting the first 20% of the classifiers from the ordered bagging ensemble. In the classification problems investigated, this pruning rule generates subensembles that consistently and significantly outperform the full bagging ensemble.

In the second pruning method, classifier aggregation is stopped when the first classifier whose weighted training error is above 0.5 is found (first boosting stop). This stopping rule leads to the selection of subensembles of very different sizes and whose classification performance is either equivalent to bagging, when the pruning procedure selects very small subensembles, or better than bagging, when the pruned ensembles are large. Using the weighted voting procedure specified by AdaBoost in the ordered bagging ensembles does not introduce significant variations with respect to unweighted voting. Experiments show that the differences in the generalization error of the pruned ensembles generated from bagging ensembles of different sizes (100 and 200) are small.

In the classification problems where boosting outperforms bagging, these pruned ensembles are slightly inferior to boosting ensembles. However, in noisy datasets the proposed pruned ensemble method outperforms both AdaBoost and bagging. The results obtained in both noisy datasets and in datasets where a percentage of the class labels of the examples have been modified, show that pruned ensembles inherit from bagging the robustness in classification performance. In summary, pruned bagging should generally be preferred to bagging and is a safe and robust alternative to AdaBoost when no information about the level of noise of the classification task is available.

Acknowledgements

The authors acknowledge financial support from the Spanish *Dirección General de Investigación*, project TIN2004-07676-C02-02.

References

Bakker, B., Heskes, T., Mar. 2003. Clustering ensembles of neural network models. Neural Networks 16 (2), 261–269.

Banfield, R. E., Hall, L. O., Bowyer, K. W., Kegelmeyer, W. P., 2005. Ensemble diversity measures and their application to thinning. Information Fusion Journal 6 (1), 49–62.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36 (1-2), 105–139.

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

Breiman, L., 1996a. Bagging predictors. Machine Learning 24 (2), 123–140.

Breiman, L., 1996b. Bias, variance, and arcing classifiers. Tech. Rep. 460, Statistics Department, University of California.

Breiman, L., 1998. Arcing classifiers. The Annals of Statistics 26 (3), 801–849.

Breiman, L., 2001. Random forests. Machine Learning 45 (1), 5–32.

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

Demir, C., Alpaydin, E., 2005. Cost-conscious classifier ensembles. Pattern Recognition Letters 26 (14), 2206–2214.

Dietterich, T. G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40 (2), 139–157.

Domingos, P., 1997. Knowledge acquisition from examples via multiple models. In: Proc. 14th International Conference on Machine Learning. Morgan Kaufmann, pp. 98–106.

Freund, Y., Schapire, R. E., 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In: Proc. 2nd European Conference on Computational Learning Theory. pp. 23–37.

Giacinto, G., Roli, F., 2001. An approach to the automatic design of multiple classifier systems. Pattern Recognition Letters 22 (1), 25–33.

Margineantu, D. D., Dietterich, T. G., 1997. Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning. Morgan Kaufmann, pp. 211–218.

Martínez-Muñoz, G., Suárez, A., 2004a. Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications. Acta Press, pp. 258– 263.

Martínez-Muñoz, G., Suárez, A., 2004b. Using all data to generate decision tree ensembles. IEEE Transactions on Systems, Man and Cybernetics part C 34 (4), 393– 397.

Martínez-Muñoz, G., Suárez, A., 2005. Switching class labels to generate classification ensembles. Pattern Recognition 38 (10), 1483–1494.

Prodromidis, A. L., Stolfo, S. J., 2001. Cost complexity-based pruning of ensemble classifiers. Knowledge and Information Systems 3 (4), 449–469.

Rätsch, G., Onoda, T., Müller, K.-R., Mar. 2001. Soft margins for AdaBoost. Machine Learning 42 (3), 287–320.

Tamon, C., Xiang, J., 2000. On the boosting pruning problem. In: Proc. 11th European Conference on Machine Learning. Vol. 1810. Springer, Berlin, pp. 404–412.

Webb, G. I., Aug. 2000. Multiboosting: A technique for combining boosting and wagging. Machine Learning 40 (2), 159–196.

Zhou, Z.-H., Tang, W., 2003. Selective ensemble of decision trees. In: Lecture Notes in Artificial Intelligence 2639, 2003, pp.476-483. Berlin: Springer, pp. 476–483.

Zhou, Z.-H., Wu, J., Tang, W., 2002. Ensembling neural networks: Many could be better than all. Artificial Intelligence 137 (1-2), 239–263.