Selection of Decision Stumps in Bagging Ensembles*

Gonzalo Martínez-Muñoz, Daniel Hernández-Lobato, and Alberto Suárez

Computer Science Department, Universidad Autónoma de Madrid
C/ Francisco Tomás y Valiente, 11
Madrid 28049, Spain
gonzalo.martinez@uam.es,daniel.hernandez@uam.es,alberto.suarez@uam.es

Abstract. This article presents a comprehensive study of different ensemble pruning techniques applied to a bagging ensemble composed of decision stumps. Six different ensemble pruning methods are tested. Four of these are greedy strategies based on first reordering the elements of the ensemble according to some rule that takes into account the complementarity of the predictors with respect to the classification task. Subensembles of increasing size are then constructed by incorporating the ordered classifiers one by one. A halting criterion stops the aggregation process before the complete original ensemble is recovered. The other two approaches are selection techniques that attempt to identify optimal subensembles using either genetic algorithms or semidefinite programming. Experiments performed on 24 benchmark classification tasks show that the selection of a small subset ($\approx 10-15\%$) of the original pool of stumps generated with bagging can significantly increase the accuracy and reduce the complexity of the ensemble.

1 Introduction

Numerous experimental studies show that pooling the decisions of classifiers in an ensemble can lead to improvements in the generalization performance of weak learners. Ensemble methods take advantage of instabilities in the algorithm used to induce a base learner. These instabilities are exploited to generate a collection of diverse classifiers that are expected to be complementary with respect to the classification task considered. Ensemble classification is then achieved by a majority voting scheme. In this context, complementarity means that the errors of the different hypothesis are not correlated, so that, when the classifiers are pooled, correct decisions are amplified. In this manner, the ensemble can achieve a better classification accuracy than a single learner.

In bagging ensembles [1] diverse classifiers are built by inducing each hypothesis from of a different bootstrap sample from the training data [2]. Typically, the

* This work has been supported by *Consejería de Educación de la Comunidad Autónoma de Madrid*, *European Social Fund*, and the *Dirección General de Investigación*, grant TIN2004-07676-C02-02.

J. Marques de Sá et al. (Eds.): ICANN 2007, Part I, LNCS 4668, pp. 319–328, 2007.

generalization error of a bagging classification ensemble decreases monotonically with the size of the ensemble. Asymptotically, this error tends to a constant level, which is considered the best result bagging can achieve. Nevertheless, this asymptotic level is reached only after a large number of hypotheses are included in the ensemble. To reduce the memory and processing requirements, the selection of a subset of classifiers from the original ensemble was first proposed in [3]. Several later investigations show that ensemble pruning can produce very good results [3,4,5,6,7,8,9]. In particular, by pruning the original ensemble, the speed and storage requirements can be significantly reduced without a significant deterioration, and sometimes with an improvement, of the generalization performance.

This article presents an empirical study of several ensemble pruning techniques applied to bagged decision stumps. Decision stumps are binary classification trees consisting in a single decision (i.e., the tree has a single internal node, the root node and two leaves). Stumps can be seen as classification rules based on one question with only two possible answers. Classification is achieved by making a class assignment for each different answer.

The paper is organized as follows: In Section 2 a short review of the different pruning techniques used in the experiments is given. In Section 3 we describe the experimental procedure carried out to compare the pruning procedures in the different classification problems considered and present the results of these experiments. Finally, Section 4 summarizes the conclusions of this work.

2 Methods

In this section we describe several heuristics designed to extract a subensemble with good generalization properties from an initial bagging ensemble. The ensemble pruning methods considered are of two types. A first family of methods is based on altering the order in which the classifiers in the original ensemble are aggregated. The original bagging ensemble is used as a pool of hypothesis from which a sequence of subensembles of increasing size is constructed. Starting from a subensemble of size $u-1$, a subensemble of size u is built by incorporating the classifier from the original bagging ensemble that is expected to maximize the generalization performance of the augmented subensemble. The final subensemble is obtained by selecting the first τ classifiers from the ordered ensemble. The value of τ is either fixed beforehand to achieve the desired amount of pruning or is estimated using the training data. The various pruning techniques considered differ by the heuristic rule that is used to guide the ordered aggregation. Heuristics based on individual properties of the classifiers have been proved not useful. Successful ordering heuristics need to take into account the complementarity between classifiers with respect to the classification task. The following is a description of the four different ordering heuristics that have been investigated in this work.

Reduce Error (RE): This heuristic was first proposed in [3]. It first selects the classifier with the lowest classification error on the training set. Then, classifiers are sequentially incorporated one by one, in such a way that the classification

error of the partially aggregated subensemble estimated on the training set is as low as possible. The algorithm designed in [3] includes the possibility of *backfitting* to correct the mistakes made by this greedy strategy. Nonetheless, *backfitting* has been shown not to reduce the generalization error significantly in bagging ensembles [7]. Furthermore, it dramatically increases the running time of the algorithm. For these reasons no backfitting was implemented in this study.

Complementariness Measure (CC): The heuristic introduced in [5] preferentially incorporates classifiers whose predictions are complementary to those of the existing subensemble. At each step in the aggregation this criterion selects the classifier that correctly classifies more examples in which the partially aggregated subensemble errs. This measure of disagreement is in fact the amount by which that classifier shifts the ensemble decision toward the correct classification.

Margin Distance Minimization (MD): This heuristic was first proposed in [5]. It is based on defining a signature vector $\mathbf{c}^t$ for each classifier t in the original ensemble. This vector has as many components as there are instances in the training set. Component c_i^t is 1 if classifier t correctly classifies the ith instance, and -1 otherwise. The average signature vector of the ensemble is defined as $\langle \mathbf{c} \rangle = T^{-1} \sum_{t=1}^{T} \mathbf{c}^t$. Note that the ith training example is correctly classified if the ith component of $\langle \mathbf{c} \rangle$ is strictly positive. As a result, if the average signature vector of a subensemble is in the first quadrant (all its components are strictly positive), it will correctly classify all examples in the training set. The greedy strategy progressively incorporates into the partially aggregated subensemble those classifiers that reduce the most the distance of the subensemble signature vector $\langle \mathbf{c} \rangle$ to a fixed point $\mathbf{o}$. This point is set to be in the first quadrant with all components equal to a small value p (e.g. $p \approx 0.1$) so that vector components corresponding to examples that are simple to classify quickly reach a value close to p and hence have less influence in future aggregation decisions. The value of the parameter p does not significantly affect the performance of the algorithm provided that it is small $p \in [0.05, 0.2]$ [5].

Boosting Based Ordering (BB): This approach was introduced in [8] and is based on the instance reweighting scheme proposed in the Adaboost algorithm [10]. The method is similar to boosting, except that, instead of generating new hypotheses from the weighted data at each iteration, the classifier with the lowest weighted error over the training set is chosen and aggregated to the partial subensemble. If there is no classifier whose weighted training error is below 50% the weights are set to be uniform. Unlike Adaboost, if the selected classifier has zero error over the training set the algorithm still continues to aggregate classifiers until all elements have been incorporated.

A second family of methods attempts to directly identify optimal subensembles using either global optimization heuristics, such as genetic algorithms, or semidefinite programming on a relaxed version of the problem:

Genetic Algorithm (GA): Following the approach described in [4], a population of individuals is evolved to identify quasi-optimal subensembles. The individuals in the population represent candidate solutions to the optimization

problem. A subensemble is represented by a binary string chromosome whose length is equal to the number of classifiers in the original ensemble, T: $\mathbf{b} \in \{0,1\}^T$. The allele value b_t indicates whether classifier t is included in the subensemble ($b_t = 1$) or not ($b_t = 0$). The configuration parameters of the GA were adjusted following the recommendations of [11], using the values proposed in [4] and information from exploratory experiments. A population of 100 individuals with diagonal initialization is considered. Uniform crossover (probability 0.65) and single bit-flip mutation (probability $5 \cdot 10^{-3}$) are used to generate variability in the population. The fitness of an individual is measured as the accuracy of the corresponding subensemble on the training set. The probability that an individual is selected for reproduction is proportional to its fitness. Elitism is used. The GA is halted after a fixed number of epochs (200) have elapsed.

Semi-definite Programming (SDP): This pruning technique has been recently introduced in [6]. It is based on building a matrix $\mathbf{G}$ whose element G_{ij} is the number of common errors between classifiers with labels i and j. The diagonal term G_{ii} is the error of the ith classifier. The elements on this matrix are then normalized $\tilde{G}_{ii} = N^{-1}G_{ii}$ and $\tilde{G}_{ij,\, i \neq j} = \frac{1}{2}\left(G_{ij}G_{ii}^{-1} + G_{ji}G_{jj}^{-1}\right)$, where N is the size of the training set. Intuitively, $\sum_i \tilde{G}_{ii}$ measures the ensemble strength and $\sum_{i \neq j} \tilde{G}_{ij}$ measures its diversity. The subensemble selection problem of size k is formulated as an integer programming problem. The goal is to minimize $arg_{\mathbf{x}}\, min\, \mathbf{x}^T \tilde{\mathbf{G}} \mathbf{x}$ s.t. $\sum_i x_i = k$ and $x_i \in \{0,1\}$, where the binary variable x_i indicates whether classifier ith should be included or not in the subensemble. The solution to this problem can be approximated very accurately by carrying out a convex semi-definite programming (SDP) relaxation.

3 Experiments

Experiments on 24 datasets from the UCI repository [12] have been performed to compare the performance of the different ensemble pruning methods when applied to bagged decision stumps. These datasets include synthetic and real-world problems from different fields of application, with different numbers of classes and attributes. Since stumps produce a binary decision, they are at a disadvantage when applied to multiclass problems, even when used in combination with bagging. In any case, it is interesting to investigate whether the pruned ensembles can achieve significant error reduction on the multiclass problems. The characteristics of the classification tasks are summarized in Table 1.

The results reported are averages over 100 executions. For each real-world dataset, these 100 executions correspond to 10 × 10-fold cross-validation. For the synthetic datasets (*Led24*, *Ringnorm*, *Twonorm* and *Waveform*) random sampling was applied to generate each of the training and testing sets. Specifically, for each dataset, the following steps were carried out: (i) Generate the training and testing sets by 10-fold-cv or random sampling (see Table 1) and generate 100 bagged decision stumps. The stumps were created using the CART growing tree algorithm [13] and bootstrap samples. The full ensemble generalization error is estimated in the unseen test set. (ii) Order the decision stumps

using: reduce-error (RE), complementariness measure (CC), Margin Distance Minimization using $p = 0.075$ (MD75) and boosting based pruning (BB). The values of the heuristics are calculaded using information only from the training set. (iii) The error of ordered ensembles is computed on the training and the test set for subensembles of sizes 1 to 100. (iv) Apply to the same original ensemble the selection approaches: GA and SDP. The SDP algorithm is set to select 21 decision stumps. Pruned subensembles with this number of stumps generally have a good performance in the classification problems investigated. The number of decision stumps selected by GA is not fixed.

Fig. 1 displays the curves that trace the dependence of the ensemble generalization error with respect to the number of stumps for randomly ordered bagging ensembles and for ensembles ordered with the different heuristics: RE, CC, MD75 and BB. The plots on the left (right) correspond to training (test) error curves. Note that in (randomly-ordered) bagging the error exhibits a monotonic decrease and eventual saturation at a constant level as the number of classifiers included in the ensemble increases. The effect of ordered aggregation is that the error initially decreases faster than in the original curve, it attains a minimum at intermediate subensemble sizes and then increases to the asymptotic error level of the complete bagging ensemble. If the aggregation process is stopped at the minimum of the error curve, the corresponding subensemble is smaller and has a better generalization accuracy than the original complete bagging ensemble. Plots are shown only for a selection of datasets, but they are representative of the qualitative behavior of error curves in all the problems investigated. The average error of the optimal subensembles with 21 classifiers selected by SDP approach is displayed in the same figure for comparison. Finally, the average error and the average size of the subensemble selected by GA are reported. Fig. 1 shows that the learning curves exhibit a qualitatively similar evolution of the error as a function of the number of stumps in both the training and test sets. In particular, the position of the minimum error for the ordering heuristics in the

Table 1. Characteristics of the datasets and testing method

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

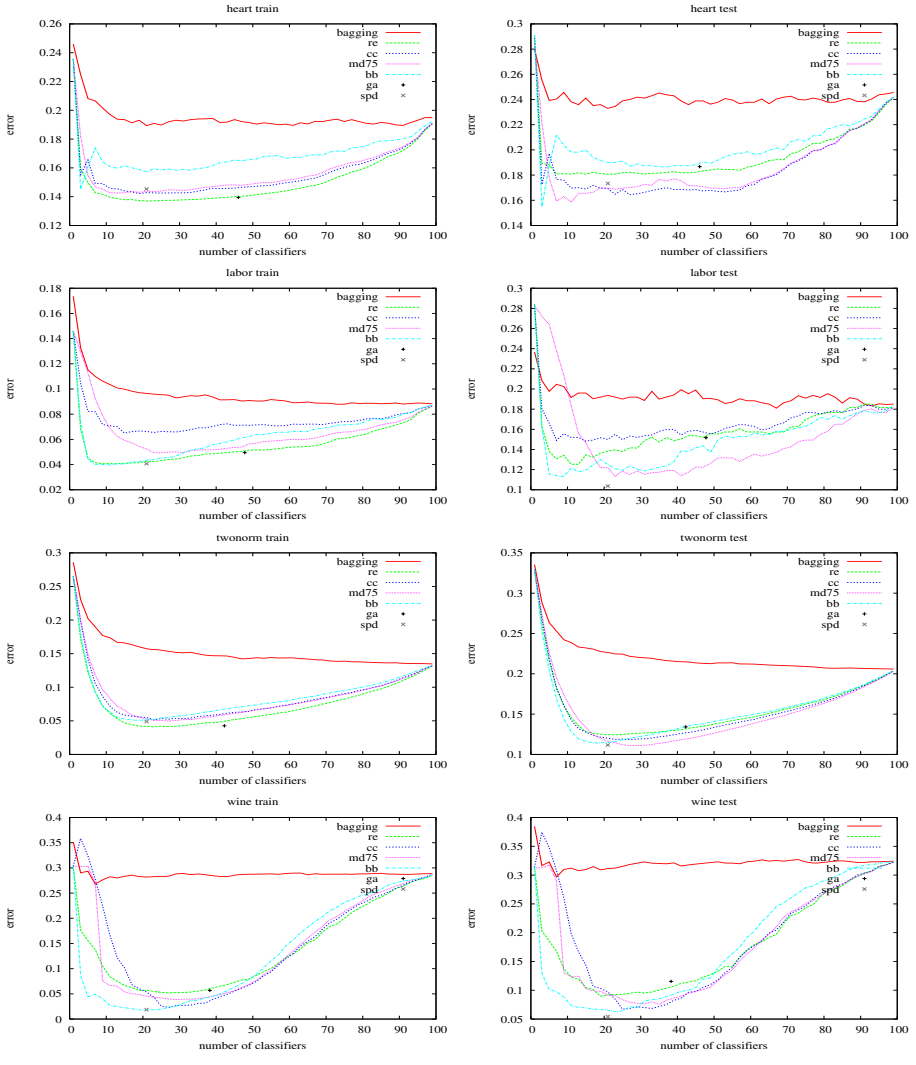

Fig. 1. Average train (*left column*) and test (*right column*) errors for *Heart* (*top row*), *Labor Negotiations* (*second row*), *Twonorm* (*third row*) and *Wine* (*last row*) datasets

training error curves tends to coincide with the position of the minimum in the test error curves. This fact makes it possible to estimate the optimum number of stumps to use in the final subensemble using information only from the training set. Note that the curves shown in Fig. 1 correspond to average values and not to single executions. Nonetheless, the proximity of the positions of the minima in the training and in the test sets is also observed in single runs. The coincidence of the minima is apparent in problems where data is abundant, which implies that

Table 2. Errors for full bagging (100 stumps) and the different ordering heuristics and selection procedures

Dataset	**bagging**	**RE**		**CC**		**MD75**		**BB**		**GA**	**SDP**
		Min	21	Min	21	Min	21	Min	21		
audio	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8	53.5±2.8
australian	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8	14.5±3.8
breast	7.0±3.7	**5.5±3.0**	**5.6±3.0**	**4.9±2.9**	**5.4±2.9**	**5.5±2.8**	**5.4±2.9**	**5.4±3.0**	**6.0±2.9**	**5.9±3.2**	**5.4±2.8**
diabetes	27.8±4.1	**26.3±4.3**	**26.2±4.3**	**26.4±4.1**	28.5±5.5	**26.3±4.2**	**26.0±4.4**	**26.5±4.1**	**26.9±4.7**	**27.0±4.3**	**26.9±4.7**
ecoli	35.4±1.8	35.5±2.0	35.4±1.8	35.3±2.3	37.6±5.1	35.4±1.9	35.5±1.9	35.5±2.0	35.4±1.8	35.4±1.8	36.6±4.1
german	30.0±0.0	31.0±2.8	30.0±0.0	31.0±2.9	30.0±0.0	31.0±2.9	30.0±0.0	31.0±2.9	30.0±0.0	30.0±0.0	30.0±0.0
glass	51.4±6.7	**38.0±7.8**	**37.8±8.0**	**36.4±7.4**	52.1±5.2	**36.0±7.5**	**36.0±8.1**	**38.9±8.4**	**48.1±9.8**	**38.3±8.2**	**39.2±8.2**
heart	24.2±10.1	**18.0±8.2**	**18.1±8.1**	**16.9±6.6**	**16.9±6.4**	**16.7±6.8**	**17.0±6.8**	**17.1±7.2**	**19.0±7.4**	**18.7±9.2**	**17.3±6.9**
horse-colic	18.6±6.3	18.7±6.2	18.6±6.3	18.7±6.2	18.6±6.3	18.6±6.3	18.6±6.3	18.7±6.2	18.6±6.3	18.6±6.3	18.6±6.3
ionosphere	17.2±5.2	**10.3±4.8**	17.0±4.7	**10.4±4.8**	16.4±5.1	**13.5±5.1**	17.3±5.1	**10.3±4.8**	17.2±5.2	**16.4±5.5**	17.5±5.3
labor	18.5±15	**11.4±13**	**13.8±13**	**13.5±13**	15.0±13	**12.3±14**	**12.2±13**	**9.9±12**	**12.5±14**	**15.2±14**	**10.4±12**
led24	77.6±4.8	**55.2±8.4**	**58.0±8.4**	**60.7±7.2**	**71.0±8.1**	**67.4±6.4**	**74.5±5.8**	**72.2±4.8**	78.0±4.6	**63.8±8.7**	**60.6±8.0**
liver	37.3±7.6	**33.3±8.2**	**33.6±8.1**	**32.5±7.5**	**30.9±7.1**	**33.1±7.5**	**32.1±7.4**	**32.2±8.0**	**32.7±7.8**	**34.3±7.7**	**32.4±8.3**
new-thyroid	22.9±4.4	**18.7±5.0**	**19.1±5.0**	**18.2±4.1**	**18.3±4.1**	**19.2±4.6**	**20.7±4.5**	**19.3±4.3**	21.7±4.5	**19.3±4.7**	**19.6±4.2**
ringnorm	43.9±2.6	**34.5±1.5**	**40.3±1.2**	**35.2±1.5**	**43.0±1.5**	**35.5±1.5**	**42.7±1.7**	**35.0±1.6**	**42.9±1.4**	**39.8±1.7**	43.3±1.6
segment	55.0±5.2	**44.0±4.0**	**45.2±5.3**	**43.3±2.5**	**50.1±7.4**	**43.3±2.5**	**52.6±6.6**	**50.8±7.0**	57.4±4.7	**43.1±2.1**	**44.9±5.3**
sonar	25.7±9.6	27.4±11.0	27.1±9.9	27.6±9.6	28.3±9.7	26.8±10.3	27.0±10.0	27.1±10.1	27.7±9.4	26.8±9.2	27.5±9.7
tic-tac-toe	30.1±4.8	30.1±4.8	30.1±4.8	30.2±4.7	30.5±4.6	30.1±4.8	30.1±4.8	30.2±4.7	30.4±4.7	30.1±4.8	30.4±4.7
twonorm	20.6±5.5	**12.2±2.0**	**12.4±2.0**	**11.6±1.9**	**12.0±1.8**	**11.3±2.2**	**11.6±1.8**	**11.3±1.9**	**11.5±2.2**	**13.4±2.9**	**11.2±1.8**
vehicle	59.6±2.7	**44.8±4.8**	**45.3±5.3**	**45.3±5.1**	**54.3±6.1**	**44.8±4.7**	**49.1±4.9**	**53.1±7.1**	**58.2±3.8**	**48.3±7.8**	**45.7±5.3**
votes	4.4±3.0	4.4±3.0	4.4±3.0	4.4±3.0	5.5±3.4	4.4±3.0	4.4±3.0	4.4±3.0	4.6±3.1	4.4±3.0	4.8±3.3
vowel	74.6±2.1	**67.2±3.2**	**68.3±3.1**	**70.8±3.7**	75.8±4.5	**69.5±3.1**	**72.9±3.4**	**72.9±2.9**	77.6±4.3	**67.5±4.2**	**70.5±3.8**
waveform	39.2±5.0	**27.5±4.8**	**29.1±6.2**	**27.8±5.0**	**29.2±5.9**	**28.9±5.0**	**30.3±5.2**	**28.0±4.9**	**30.4±6.7**	**32.4±7.1**	**29.3±6.3**
wine	32.4±7.6	**8.6±9.5**	**9.2±10.2**	**5.2±5.2**	**9.7±11.9**	**7.0±6.1**	**9.2±6.7**	**6.2±5.5**	**6.5±5.6**	**11.5±11.7**	**5.4±5.2**
Average	34.2±5.2	27.9±5.4	28.9±5.4	28.1±4.9	31.1±5.5	28.5±5.0	30.1±5.0	29.3±5.1	31.7±5.1	29.5±5.6	29.0±5.2

Table 3. Number of selected stumps for the different datasets and algorithms

Dataset	**RE**	**CC**	**MD75**	**BB**	**GA**
audio	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	50.0±5.0
australian	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	49.9±4.7
breast	12.2±19.3	8.8±9.3	11.7±10.0	6.6±5.9	49.0±5.0
diabetes	1.8±1.6	4.9±8.3	3.9±7.8	2.6±5.0	46.7±6.3
ecoli	1.0±0.0	1.1±0.4	1.0±0.0	1.0±0.0	50.8±4.7
german	1.0±0.1	1.1±0.6	1.1±0.5	1.1±0.4	50.6±4.5
glass	12.6±16.1	14.4±17.5	16.9±15.0	42.5±24.1	47.7±5.4
heart	14.4±12.0	20.0±14.0	19.4±13.9	14.0±18.2	46.1±5.7
horse-colic	1.0±0.1	1.0±0.1	1.0±0.0	1.0±0.1	50.0±4.8
ionosphere	2.0±0.2	2.2±2.0	8.8±8.6	2.1±0.6	44.4±6.3
labor	5.8±5.6	8.4±7.7	17.4±9.9	5.9±3.6	47.8±5.6
led24	22.7±11.7	34.0±18.8	37.6±20.8	23.6±23.1	45.3±6.3
liver	26.6±13.6	28.3±12.9	24.6±7.3	19.5±10.9	42.9±6.5
new-thyroid	3.3±2.4	14.7±26.7	17.2±28.0	11.7±23.9	46.9±6.2
ringnorm	3.7±5.6	3.6±1.6	3.8±2.9	3.4±1.3	41.1±7.5
segment	32.5±25.1	26.5±21.3	33.0±19.8	35.2±28.6	49.2±5.0
sonar	11.3±6.8	12.9±13.0	14.7±5.8	8.9±4.6	45.2±5.7
tic-tac-toe	1.0±0.0	1.0±0.2	1.0±0.0	1.0±0.2	50.0±5.2
twonorm	24.8±8.9	24.8±9.5	26.4±6.7	20.1±6.7	42.3±7.1
vehicle	14.4±11.5	24.9±17.2	18.8±16.0	19.0±19.8	41.0±7.1
votes	1.0±0.0	1.0±0.0	1.0±0.0	1.0±0.0	49.6±5.3
vowel	32.0±19.4	38.8±26.7	26.7±20.7	50.0±26.8	46.5±5.2
waveform	18.2±11.2	22.1±14.5	23.8±13.9	16.6±11.8	41.3±7.2
wine	16.9±10.4	22.3±9.1	33.4±10.2	20.0±7.7	38.3±7.7
Average	10.9±7.6	13.3±9.6	14.4±9.1	12.9±9.3	46.4±5.8

the sampling fluctuations are small (e.g. *Twonorm*). It is less clear in problems where few examples are available for either training or testing (e.g. *Labor*). This is in contrast with ensembles composed of more complex classifiers (e.g. CART trees). In such ensembles, the minimum in the training error curves is generally achieved for smaller subensembles than in the test curves [7].

Table 2 displays the test error in the different classification tasks, averaged over the 100 executions. The results obtained with the ordering heuristics, which produce a collection of subensembles of increasing size (i.e. RE, CC, MD75 and BB), are presented in two columns: the leftmost column indicates the test error for subensembles of a fixed size (21 stumps). The column immediately to the right displays the average test error of subensembles whose size is determined from the position of the minimum of the training error (column labeled with *Min*). Note that the values displayed in columns *Min* of the Table and the values of the test curves shown in Fig. 1 do not generally coincide: the former is an average for different subensemble sizes and the latter is an average for a fixed number of classifiers. Results for the selection approaches GA and SDP are also shown in Table 2. Values of the test error that are significantly better than bagging (using a paired t-test with $p - value < 0.01$) are highlighted in boldface. Errors

for cases where bagging performs significantly better (at a $p - value < 0.01$) are underlined. In general, pruned bagging ensembles composed of decision stumps perform better than the complete original ensemble. The largest improvements are obtained using the reduce error and complementariness ordering heuristics to aggregate decision stumps until a minimum in the training error is reached. The improvements in accuracy are fairly large in some domains: In *Wine* the error drops from 32.4 of full bagging to 5.2 of complementariness. In *Twonorm* the generalization error nearly halves. In *Heart* the error achieved by MD75 (Min) is 16.7 and bagging obtains 24.2, etc. As anticipated, the generalization error in classification tasks with more than 2 classes is rather large. Notwithstanding, the pruning procedures manage to significantly reduce the generalization error in multiclass classification tasks: *Glass, Led24, New-thyroid, Segment, Vehicle, Vowel, Waveform* and *Wine*. In any case, the performance in multiclass problems generally remains rather poor.

Table 3 presents the average number of stumps selected for each problem by the different ordering heuristics, using the minimum error in the training dataset and by the GA stumps selection approach. SDP subensembles have a fixed prespecified size of 21 decision stumps. The tabulated figures show that the amount of pruning varies accross methods and datasets. However, on average, the number of selected stumps is fairly small (10–15%) for all ordering heuristics. The smallest subensembles are generally obtained using the reduce error ordering heuristic. This method also obtains low generalization errors. Note that for several datasets (*Audio, Australian, E-coli, ...*) the number of selected stumps is on average 1 and that, on these problems, the average generalization error of full bagging and just one stump is very similar. For these classification tasks bagging in not able to generate enough diversity and most of the decision stumps in the ensemble are actually equal. For most of the problems analyzed the genetic approach selects subensembles with approximately 50% of the original bagged stumps. Thus, GA tends to select suboptimal subsensembles that are too large.

4 Conclusions

This paper presents an empirical evaluation of different techniques used to prune ensembles of decision stumps generated with bagging. Six ensemble pruning heuristics are tested: reduce error (RE), complementariness (CC), margin distance minimization (MD75), boosting based pruning (BB), genetic algorithms (GA) and semidefinite programing (SDP). The first four heuristics are greedy approaches that determine the order in which the classifiers generated in bagging are incorporated into the ensemble according to some rule that exploits the complementarity of the base learners. Pruning is performed by selecting the first τ classifiers in the ordered ensemble either by specifying a fixed subemsemble size (21 in this investigation) or by estimating the optimum number of classifiers in some manner. For decision stumps the minimum of the curve that displays the dependence of the test error with the size of the subensemble is fairly close to the minimum in the corresponding curve for the training error. This is probably

related to the fact that decision stumps are simple classifiers and do not tend to overfit the data. Hence, the optimal stopping point for the ordered aggregation is close to the minimum of the error curve in the training dataset. The GA and SDP approaches attempt to identify a single subensemble that is optimal according to some estimate of the generalization performance. The GA approach solves the selection problem by a genetic algorithm without prescribing a target size for the optimal subensemble. In SDP a convex semi-definite programming relaxation is used to approximately maximize a goal function dependent on both accuracy and diversity for a prespecified ensemble size. For most of the datasets the pruning techniques investigated significantly reduce the generalization error of bagged decision stumps by selecting a fairly small subset of classifiers. The best performances are for ordered bagging, where the order of aggregation is determined by either the reduce error or complementariness heuristic, and aggregation stops at the minimum of the error curve for the training data. SDP obtains good overall results despite the fact that the number of classifiers needs to be fixed *a priori*. The performance of GA in terms of generalization error and percentage of pruning achieved is significantly worse than the other methods.

References

1. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
2. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall/CRC (1994)
3. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning, pp. 211–218. Morgan Kaufmann, San Francisco (1997)
4. Zhou, Z.H., Tang, W.: Selective ensemble of decision trees. In: Wang, G., Liu, Q., Yao, Y., Skowron, A. (eds.) RSFDGrC 2003. LNCS (LNAI), vol. 2639, pp. 476–483. Springer, Heidelberg (2003)
5. Martínez-Muñoz, G., Suárez, A.: Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications, pp. 258–263. Acta Press (2004)
6. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research 7, 1315–1338 (2006)
7. Martínez-Muñoz, G., Suárez, A.: Pruning in ordered bagging ensembles. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 609–616 (2006)
8. Martínez-Muñoz, G., Suárez, A.: Using boosting to prune bagging ensembles. Pattern Recognition Letters 28(1), 156–165 (2007)
9. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence 137(1-2), 239–263 (2002)
10. Freund, Y., Schapire, R.E.: 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 (1995)
11. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer, Berlin (2003)
12. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
13. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)