Aggregation Ordering in Bagging

Gonzalo Martínez-Muñoz and Alberto Suárez
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco
Madrid 28049, Spain
email:{gonzalo, alberto.suarez}@ii.uam.es

ABSTRACT
The order in which classifiers are aggregated in ensemble methods can be an important tool in the identification of subsets of classifiers that, when combined, perform better than the whole ensemble. Ensembles with randomly ordered classifiers usually exhibit a generalization error that decreases as the number of classifiers that are aggregated increases. If an appropriate order for aggregation is chosen, the generalization error reaches, at intermediate numbers of classifiers, a minimum, which lies below the asymptotic error of the ensemble. This work presents some heuristics that exploit the relations between the classifiers in a bagging ensemble to identify the appropriate ordering and then select a subset for aggregation according to a desired amount pruning. The resulting subensembles are smaller and improve the classification performance of the original ensemble.

KEY WORDS
Machine learning, bagging, ensemble pruning, decision trees

1 Introduction

Classification methods based on pooling the decisions of an ensemble of classifiers have demonstrated great potential for improvement in many regression and classification problems [1, 2, 3, 4, 5, 6, 7, 8, 9]. These techniques make use of instabilities in the algorithms that generate the base classifiers in order to construct an ensemble of diverse classifiers whose joint action may improve the classification performance of a single classifier.

One of the common procedures to generate classifier ensembles is *bagging* [3] (*Bootstrap sampling and aggregation*). In bagging, the diversity is obtained by using different data for induction: Each classifier is constructed using as training set a different bootstrap sample of the original training data. The bootstrap sample has the same size as the train set, and is obtained by random sampling with replacement from it. Each sample contains on average 63.2% of the original training set. The rest of the sample are repeated examples. The aggregation phase combines the decisions of the classifiers using a simple voting method, where all classifiers are allocated equal weights.

In bagging ensembles, the generalization error usually decreases as the number of classifiers increases. The error tends asymptotically to a constant level, which is assumed to be the best result bagging can produce. This monotonic error descent is a consequence of the randomness inherent in bagging: the ordering of the aggregated classifiers in a bagged ensemble derives from the random order of the bootstrap process, where the possible cooperation between complementary classifiers is not taken into account.

The main idea of the present work is to exploit the correlations between the individual classifiers in bagging to select a subset of the ensemble that outperforms the whole ensemble. Assume we have generated an ensemble composed of T different classifiers. There are $2^T - 1$ possible subensembles, some of which have a generalization performance that is better than that of the full ensemble. Our objective is to design a procedure that can select the subensemble with the lowest generalization error. In [10], Tamon *et al.* proved that, assuming that minimizing the training error leads to the minimization of the generalization error, the problem of selecting the best subensemble is NP-complete. In order to simplify the search in the space of subensembles, it is assumed that the best subensemble of size $u - 1$ is included in the best subensemble of size u, and that we can construct a sequence of best subensembles of increasing size by including one classifier at a time. Although this assumption is not necessarily true, it seems likely that, in general, the best subensembles of sizes $u - 1$ and u share most of their classifiers. The problem is then to design a rule to determine the order in which classifiers are aggregated.

In Section 2 we briefly describe previous work on ensemble pruning. Section 3 introduces the learning algorithm and the rules that we use to order the classifiers in bagging: Reduce-Error ordering, Complementariness measure, and Margin Distance Minimization. The first one is a simplified version of the reduce-error pruning rule presented in [11] without backfitting, while the last two procedures are novel proposals. The procedures designed are empirically tested in ten datasets, synthetic and real-world, included in the UCI repository [12]. Finally, the conclusions and perspectives of this research are summarized.

2 Previous Work

As noted by Margineantu & Dietterich in [11], a drawback of ensemble methods is the large amount of memory required to store all the classifiers in the ensemble. For a real world application, it is difficult to justify the usage of a classifier when this requires more storage capacity than the database from which it is generated, especially when a simple dictionary lookup or nearest-neighbors may also perform well. This observation prompted the authors to investigate whether all classifiers generated with Adaboost [1] are essential for its accuracy. The decrease in storage requirements is not the only benefit that can be obtained from the reduction of the ensemble size. There are also gains in the speed of classification, which is a critical issue for inline applications. In [11], Margineantu & Dietterich propose some interesting heuristics for selecting the essential classifiers in an Adaboost ensemble for a given degree of pruning. Most of these heuristics for ensemble pruning are based on measures of diversity and performance. The experiments they present indicate that one can significantly reduce the number of classifiers (up to 60 to 80% pruning in some domains) without a substantial degradation of the classification performance.

A different approach is taken by Zhou *et al.* [13, 14]. These authors propose the use of a genetic algorithm to determine an optimal set of weights for the classifiers in an ensemble by minimizing a function related to the generalization ability of the ensemble. The optimization problem is solved by using a standard Genetic Algorithm with a floating-point scheme for real-valued weights in neural network ensembles and with a binary scheme for 0/1 weights in decision tree ensembles. They then eliminate from the ensemble those nets or trees whose optimized weights are below a specified level. Their experiments were made using rather small ensembles (20 elements).

Prodromidis *et al.* [15] propose a method for reducing the number of base learners of any ensemble based in the cost complexity pruning technique of the CART algorithm [16]. They train a CART tree with the outputs of the base classifiers as attributes and the final decision of the ensemble as the class label. Then this meta-learner is pruned using the cost complexity methodology of CART. Finally the base classifiers that are not used in the pruned tree are removed from the ensemble and the ensemble is restructured.

Our approach avoids both the computationally complex optimization process in the space of weights [13, 14] and the pruning of a representation of the original ensemble [15]. The proposed pruning procedure is based on ordering the predictors in the ensemble according to a number of straightforward rules that exploit the complementariness of the individual classifiers.

3 Learning Algorithm

In this section we present a brief introduction of bagging, which is the method used in this work to generate an ensemble of classifiers, followed by a detailed discussion of the rules for selecting the order in which the classifiers in the ensemble should be aggregated.

The input for the learning algorithm is the set of training data, which consists in a collection of N_{train} labeled examples $L = \{(\boldsymbol{x}_i,\ y_i),\ i = 1, 2, ..., N_{train}\}$. Each training example is characterized by a feature vector $\boldsymbol{x}_i$ and a class label y_i. The goal of the classification algorithm is to generate a hypothesis, $H(\boldsymbol{x})$, which, given a feature vector $\boldsymbol{x}$ as an input, predicts its class label y, using the knowledge contained in the training dataset L. For the sake of simplicity, in this section, we consider only binary classification problems where $y_i = \pm 1$. The results can be easily extended to multiclass problems.

Instead of inducing a single hypothesis from L, ensemble methods create a diversity of hypotheses and then combine their classifications to yield a final decision. Bagging generates a set of hypotheses $h_t(\boldsymbol{x}) : t = 1, \ldots, T$, by constructing a classifier for each of a number of different bootstrap samples from L. The final decision of the bagged ensemble is taken by simple majority with unweighted voting. Assuming a binary classification problem, where $h_t(\boldsymbol{x}) = \pm 1$, the combined ensemble hypothesis is

$$H(\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} h_t(\boldsymbol{x})\right) . \qquad (1)$$

Both the process that bagging uses for generating the different hypotheses (*bootstrap*) and the voting procedure (*aggregation*) do not take into account the relations among the different classifiers in the ensemble.

In this work we present a strategy to improve the generalization performance of bagging by selecting subsets of classifiers from the original ensemble. Starting with a subensemble of size $u - 1$, a subensemble of size u is obtained by adding an extra classifier, which is selected according to a specified rule. This process amounts to finding the appropriate aggregation ordering of the bagged predictors. The original random ordering $t = 1, 2, \ldots, T$ of classifiers in the bagging ensemble is replaced by a different ranking $s_1, s_2, \ldots, s_T$, where s_j is the original position index of the classifier that occupies the $j^{th\cdot}$ position in the newly ordered ensemble. Finally, the M first classifiers are retained, depending on the desired amount of pruning.

Some exploratory experiments were made in order to ascertain whether characteristics of the individual classifiers are useful criteria to order the elements in the ensemble. In particular, different estimates of the individual classification error were used to determine the aggregation ordering. The resubstitution error on the training set is not a reliable indicator of the generalization performance of a given classifier and does not lead to any useful ordering in the ensemble. Experiments where a single, sufficiently

large validation set, independent of the training data, is used to estimate the generalization error also failed to produced a useful ordering. Based on these results, we conclude that ranking procedures guided by individual performance measures do not lead to the identification of subsets of classifiers that outperform the whole bagging ensemble. In order to design useful ordering strategies, it is important to take into account the complementary action of the classifiers. Indeed, if we bring together very accurate classifiers, but with a large degree of similarity, their classification accuracy does not improve, while if we pool the results from classifiers that compensate each other's errors we do obtain an increased accuracy [8].

The successful ordering methods make use of the complementariness of different ensemble elements in the classification process. An individual classifier may have a poor classification performance but its contribution can be important when combined with other classifiers. We proceed now to describe in detail some ordering rules that have proved useful: Reduce-Error pruning, Complementariness measure, and Margin Distance Minimization. The rules make use of a selection set of N_{sel} labeled examples $\{(\boldsymbol{x}_i,\, y_i),\, i = 1, 2, ..., N_{sel}\}$, which could in principle coincide with the training set.

Reduce-Error pruning: This method is equivalent to the one presented in [11] without backfitting. It works as follows: the smallest (size one) subensemble in the series is the classifier with the lowest classification error on a selection set. Classifiers are then added to the ensemble one by one, in such a way that the classification error estimated on the selection set of the partial ensemble is as low as possible. Hence, the selected classifier in the u^{th} iteration is the one that maximizes the expression:

$$s_u = \underset{k}{argmax} \sum_{i=1}^{N_{sel}} sign\left(h_k(\boldsymbol{x}_i) + \sum_{t=1}^{u-1} h_{s_t}(\boldsymbol{x}_i)\right) y_i \ , \tag{2}$$

where k index runs throughout the classifiers that have not been included in the subensemble of size $u-1$.

Complementariness measure: This procedure favors the inclusion of classifiers whose performance is complementary to that of the subensemble. As in the previous procedure the subensembles grow from the classifier with the lowest error on the selection set. From a subensemble of size $u-1$, the subensemble of order u is obtained by incorporating into the subensemble the classifier that maximizes

$$s_u = \underset{k}{argmax} \sum_{i=1}^{N_{sel}} I\Bigg(y_i = h_k(\boldsymbol{x}_i)\ AND \ y_i \neq sign\Big(\sum_{t=1}^{u-1} h_{s_t}(\boldsymbol{x}_i)\Big)\Bigg) \ , \tag{3}$$

where k runs through the classifiers that have not been selected up to that point, and $I(true) = 1,\ I(false) = 0$. This quantity can be thought of as the amount by which one classifier shifts the decision of the ensemble towards the correct classification. This criterion favors the inclusion in the ensemble of classifiers that correctly classify examples in which the partial subensemble errs.

Margin Distance Minimization: Given a labeled selection set of size N_{sel}, consider the quantity $\boldsymbol{c}_t$ defined for each of the classifiers in the ensemble. It is an N_{sel}-dimensional vector whose i^{th} component is

$$(c_t)_i = y_i h_t(\boldsymbol{x}_i),\ i = 1, 2, \ldots, N_{sel} \ . \tag{4}$$

The quantity $(c_t)_i$ is equal to 1 if the t^{th} classifier correctly classifies the i^{th} training example and -1 otherwise. The accuracy of the ensemble can be expressed as an average over the vectors $\boldsymbol{c}_t$. The i^{th} example is correctly classified by the ensemble if the i^{th} component of the average vector $\langle \boldsymbol{c} \rangle = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{c}_t$ is positive. Note that $\langle \boldsymbol{c} \rangle$ is a vector whose components are equal to the example margins according to the definition given in Ref. [5] and are therefore bounded between -1 and 1. For a subensemble that correctly classifies all the data in the selection set, vector $\langle \boldsymbol{c} \rangle$ is in the first quadrant of the N_{sel}-dimensional hyperspace (i.e., all components are positive). Our goal is to select a subensemble where this vector is as close as possible to a reference position placed somewhere in the first quadrant. We arbitrarily select this objective position as a constant vector with equal components:

$$o_i = p \qquad i = 1, \ldots, N_{sel} : \text{with } 0 < p < 1 \ . \tag{5}$$

Classifiers are added to the ensemble in order to reduce most the distance from the vector $\langle \boldsymbol{c} \rangle$ to the objective point $\boldsymbol{o}$. The u^{th} selected classifier is the one that minimizes

$$s_u = \underset{k}{argmin}\, d\left(\boldsymbol{o}, \frac{1}{T}\left(\boldsymbol{c}_k + \sum_{t=1}^{u-1} \boldsymbol{c}_{s_t}\right)\right) \ , \tag{6}$$

where k runs throughout the classifiers outside the subensemble and where $d(\boldsymbol{u}, \boldsymbol{v})$ is the usual quadratic distance between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

The value of p should be chosen small (i.e. $p \sim (0.05, 0.25)$). In such manner, easy examples (i.e. those correctly classified by most of the classifiers) will promptly achieve a value near p and, consecuently, their influence in the selection of the next classifiers will be reduced, increasing the influence of the harder examples. On the other hand, if a value of p near 1 is selected, there is a similar attraction for all examples throughout the selection process, making the method less effective.

4 Experimental Results

The proposed ordering rules have been tested on ten different machine learning problems. Two of these are synthetic datasets (Twonorm and Ringnorm) proposed by Breiman in [17]. The remaining problems are included in the UCI

Table 2. Average train error in % for ensembles of 10%, 20% and 40% classifiers. Results having standard deviations higher than bagging are shown in italics.

	Bagging	**Reduce-Error**			**Complementariness**			**Distance ($p = 0.075$)**		
Ensemble size	100%	10%	20%	40%	10%	20%	40%	10%	20%	40%
Breast W.	3.0±0.6	0.5	0.9	1.5	1.1	1.4	1.9	0.8	1.8	2.5
Diabetes	20.5±1.5	*11.0*	*12.6*	*15.0*	*12.5*	*14.2*	16.5	*12.1*	*14.7*	17.3
German	20.4±1.2	*12.1*	13.0	14.5	*13.4*	14.5	15.9	*13.2*	14.7	16.7
Heart	11.4±2.4	1.8	2.8	4.9	3.2	4.7	6.8	2.7	5.2	8.2
Ringnorm	2.9±1.0	0.3	0.3	0.5	0.6	0.6	0.9	0.4	0.7	1.6
Segment	3.7±1.6	0.1	0.2	0.7	0.8	0.8	1.3	0.2	0.7	1.7
Thyroid	2.6±1.2	0.0	0.0	0.2	0.2	0.3	0.5	0.0	0.2	1.0
Twonorm	0.6±0.6	0.0	0.0	0.0	0.3	0.1	0.1	0.0	0.1	0.5
Waveform	9.9±2.6	0.8	1.2	2.8	1.9	2.4	4.2	1.5	3.1	6.3
Wine	0.9±1.0	0.0	0.0	0.0	0.4	0.4	0.4	0.0	0.0	0.1

Table 1. Characteristics of the used datasets

Dataset	**Train**	**Test**	**Attribs**	**Classes**
Breast W.	500	199	9	2
Diabetes	468	300	8	2
German	700	300	20	2
Heart	170	100	13	2
Ringnorm	400	7000	20	2
Segment	210	2100	19	7
Thyroid	140	75	5	3
Twonorm	400	7000	20	2
Waveform	300	4700	21	3
Wine	100	78	13	3

repository [12]: Breast Cancer Wisconsin, Pima Indian Diabetes, German Credit, Heart, Image Segmentation, Wine, Thyroid and Waveform. Table 1 displays the number of examples used for training and testing, the number of attributes and the number of classes for each dataset.

For each dataset 100 experiments were carried out. Each experiment involved the following steps:

1. Generate a stratified random partition of the data in training and testing (see table 1 for sizes).
2. Create a bagging ensemble of 200 CART trees.
3. Order the decision trees using the three procedures described in the previous section (Error-Reduction, Complementariness measure and Margin Distance Minimization), using as selection set the training set used in the creation of the ensemble. For the Margin Distance Minimization procedure a value of $p = 0.075$ was chosen based on some preliminary tests [1].
4. Finally, the ordered ensembles were evaluated for sizes of 10%, 20% and 40% (i.e., 90%, 80% and %60 percent pruning) of the original ensemble, using the testing set.

[1] Similar accuracies are obtained with $p = 0.05$ and $p = 0.25$. With $p = 0.25$, however, larger sub-ensembles are needed.

Tables 2 and 3 present a summary of the train and test errors obtained by the different ordering procedures at the different pruning levels for the analyzed datasets. The numbers reported are averages over the 100 experiments. The first column of the table shows the dataset name. The second column reports the test error of bagging when all classifiers are used. The standard deviation is shown after the $\pm$ sign. The following groups of columns present the average test error for Reduce-Error, Complementariness and Margin Distance Minimization ($p = 0.075$) methods for ensemble sizes of 10%, 20% and 40% of the original. Standard deviations are not shown. However, they are lower than in bagging for most cases except for some results (marked with italics in the tables) in the German and Diabetes datasets and never more than 0.1-0.3 higher.

Table 3 shows the lowest test error for every dataset in boldface. In all cases the improvement over bagging is statistically significant (p-value under 10^{-7} in a two-sided paired Student's t-test). Margin Distance Minimization is the most efficient method in eight of the tested domains and is very close to the best result in the Diabetes and Breast W. problems. Complementariness is the best method in these two sets and obtains results similar to Reduce-Error method in the other sets (except for Twonorm).

In table 3 we observe that the proposed methods generally reduce the classification error of the whole bagging ensemble for the studied datasets. This improvement is achieved for a large range of pruning values. The generalization error usually goes below the asymptotic bagging error starting with small subensembles, which contain less than 10% of the classifiers. Another important aspect is that, in general, the method that exhibits better generalization errors (Margin Distance Minimization with 20% of the classifiers) does not coincide with the method with the best training accuracy (Reduce-Error with 10% of the classifiers). As an extreme example note that in the Heart dataset

Table 3. Average test error in % for ensembles of 10%, 20% and 40% classifiers. Best method is shown in bold face. Results having standard deviations higher than bagging are shown in italics.

	Bagging	**Reduce-Error**			**Complementariness**			**Distance ($p = 0.075$)**		
Ensemble size	100%	10%	20%	40%	10%	20%	40%	10%	20%	40%
Breast W.	4.6±1.6	4.1	4.0	4.0	**3.8**	4.0	4.1	3.9	3.9	4.3
Diabetes	25.1±1.9	24.3	24.3	24.3	24.1	**24.0**	24.3	24.2	24.1	24.2
German	25.3±2.1	*24.1*	*23.9*	*24.0*	*23.9*	23.7	*23.8*	24.0	**23.6**	*24.0*
Heart	19.7±4.7	19.8	18.7	18.5	19.0	18.4	17.8	19.4	17.7	**17.3**
Ringnorm	10.2±1.9	9.2	8.9	8.8	9.0	8.6	8.5	8.3	**7.9**	8.4
Segment	9.7±1.7	8.1	8.0	8.2	8.5	8.3	8.4	**7.8**	**7.8**	8.4
Thyroid	7.3±3.2	6.4	6.3	6.2	6.2	6.2	6.1	5.9	**5.6**	5.8
Twonorm	8.4±2.7	9.0	7.8	**7.4**	9.6	8.6	8.2	7.8	**7.4**	7.9
Waveform	22.2±2.1	20.5	20.0	20.0	20.3	19.7	19.7	20.1	**19.2**	19.7
Wine	6.7±4.2	6.1	5.9	6.4	5.8	6.0	5.9	5.2	**3.8**	4.1

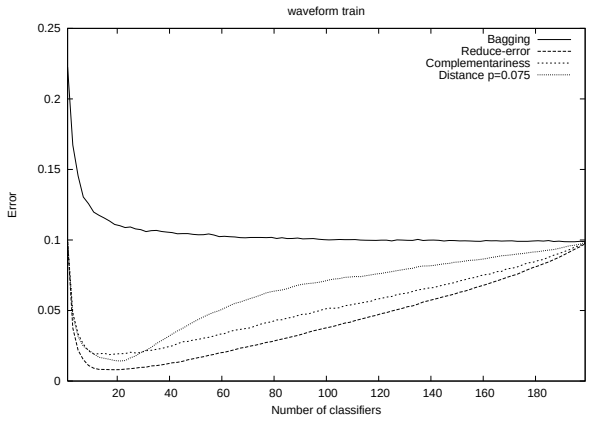

Figure 1. Train error against number of classifiers for the presented methods in Waveform (averaged over 100 runs).

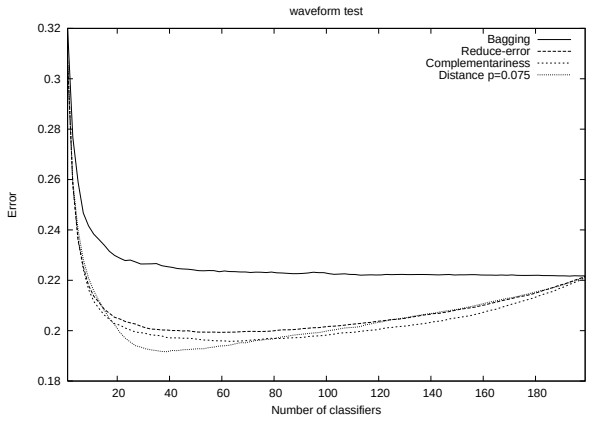

Figure 2. Test error against number of classifiers for the presented methods in Waveform (averaged over 100 runs).

the worst training error figure (Margin Distance Minimization with 40% of the classifiers) corresponds to the lowest test error. Conversely, the pruning method leading to the lowest training error (Reduce-Error with 10% of the classifiers) exhibits the largest error on the test set. These results indicate that a method such as Reduce-Error, based exclusively on the reduction of the training error does not fully exploit the cooperation among different classifiers.

Figures 1 and 2 display the training and test error, respectively, as a function of the number of classifiers in the subensemble for the Waveform dataset. The different curves correspond to different aggregation orderings: The solid line corresponds to the initial (random) ordering from bagging. The discontinuous lines correspond to orderings based on the Reduce-error, Complementariness and Margin Distance Minimization ($p = 0.075$) procedures. These figures illustrate the typical dependence of the classification error on the number of bagged predictors. As expected, for randomly ordered ensembles the test error diminishes monotonically as the number of classifiers in the subensemble is increased, until it reaches asymptotically a constant error level. In contrast, ensembles ordered by the ranking procedures based on the complementary action of the classifiers lead to test error curves that exhibit a minimum for intermediate numbers of classifiers. For all but the smallest subensembles, the test error curves displayed in figure 2 lie below the asymptotic bagging error, making it easy to select a subensemble that outperforms the original bagging ensemble. In general, as can be seen from figures 1 and 2 and tables 2 and 3 training error minima tend to occur for smaller subensembles than test error minima. This introduces some difficulty in the selection of the pruning percentage that leads to the best generalization error.

5 Conclusions and perspectives

We propose to use aggregation ordering to select an optimal subset of classifiers from an ensemble. If classifiers are

ordered by rules that take into account their complementariness, the graph of the generalization error (estimated on an independent test set) as a function of the number of aggregated classifiers exhibits a characteristic shape with a minimum. This minimum corresponds to a subensemble with fewer classifiers and a lower generalization error than the whole ensemble. We present the results obtained for three different ordering rules: Reduce-Error ordering, Complementariness measure, and Margin Distance Minimization. These rules should be useful in ensembles where predictors do not have a prescribed aggregation ordering; i.e., they should work for bagging [2, 3], random forests [18], iterated growing and pruning ensembles [19], etc. but not for boosted ensembles.

Rules that order classifiers based on some characteristic of the individual classifiers, such as individual training error, do not lead to any improvement over bagging. This is because these ranking procedures do not take into account the complementariness among classifiers for constructing the ensemble. This feature is addressed explicitly in the proposed methods: the Margin Distance Minimization aims to make the margins for each of the examples in the training set positive. Complementariness measure selects classifiers based on their joint action. The experiments carried out show that the Margin Distance Minimization with 80% percent pruning achieves a good average performance, better in most cases than the other methods considered.

We have also observed that the ordered ensembles exhibit a generalization error that is lower than the full bagging test error for a large range of pruning rates. This means that with the proposed methods a smaller and better-performing ensemble can be easily obtained and consequently, their use in bagging is advisable. Nevertheless, since the training set error seems to underestimate the size of the subensemble where the generalization error minimum occurs, further work is needed to determine the optimal subensemble size.

References

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

[2] J. Quinlan, Bagging, boosting, and C4.5, *Proc. 13th National Conference on Artificial Intelligence*, Cambridge, MA, 1996, 725–730

[3] L. Breiman, Bagging predictors, *Machine Learning*, 24(2), 1996, 123–140

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

[5] R. Schapire, Y. Freund, P. Bartlett and W. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, *The Annals of Statistics*, 12(5), 1998, 1651–1686

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

[7] A.J.C. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, (London, Springer-Verlag, 1999)

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

[9] G. Rätsch, T. Onoda and K.R. Müller, Soft margins for AdaBoost, *Machine Learning*, 42(3), 2001, 287–320

[10] C. Tamon and J. Xiang, On the boosting pruning problem, *Proc. 11th European Conference on Machine Learning*, 2000, 404–412

[11] D.D. Margineantu and T.G. Dietterich, Pruning adaptive boosting, *Proc. 14th International Conference on Machine Learning*, 1997, 211–218

[12] C.L. Blake and C.J. Merz, UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html] (1998)

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

[14] Z.H. Zhou and W. Tang, Selective ensemble of decision trees, *Lecture Notes in Artificial Intelligence 2639, 2003, pp.476-483*, Berlin: Springer, 2003

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

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

[17] L. Breiman, Bias, variance, and arcing classifiers, Technical Report 460, Statistics Department, University of California, 1996

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

[19] G. Martínez-Muñoz and A. Suárez, Using all data to generate decision tree ensembles, *IEEE Transactions on Systems, Man and Cybernetics C*, accepted for publication, 2003