A double pruning algorithm for classification ensembles

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

Universidad Autónoma de Madrid, EPS, Calle Francisco Tomás y Valiente, 11,
Madrid 28049 Spain

Abstract. This article introduces a double pruning algorithm that can be used to reduce the storage requirements, speed-up the classification process and improve the performance of parallel ensembles. A key element in the design of the algorithm is the estimation of the class label that the ensemble assigns to a given test instance by polling only a fraction of its classifiers. Instead of applying this form of dynamical (instance-based) pruning to the original ensemble, we propose to apply it to a subset of classifiers selected using standard ensemble pruning techniques. The pruned subensemble is built by first modifying the order in which classifiers are aggregated in the ensemble and then selecting the first classifiers in the ordered sequence. Experiments in benchmark problems illustrate the improvements that can be obtained with this technique. Specifically, using a bagging ensemble of 101 CART trees as a starting point, only the 21 trees of the pruned ordered ensemble need to stored in memory. Depending on the classification task, on average, only 5 to 12 of these 21 classifiers are queried to compute the predictions. The generalization performance achieved by this double pruning algorithm is similar to pruned ordered bagging and significantly better than standard bagging.

Key words: ensemble pruning, instance-based pruning, ensemble learning, decision trees

1 Introduction

There is extensive empirical evidence that combining the predictions of complementary classifiers is a successful strategy to build robust classification systems with good generalization performance [1–3]. The main disadvantages of ensemble methods are the difficulties in the interpretation of the decisions of the ensemble and their large computational requirements. In particular, the training cost, the storage needs and the time of prediction increase linearly with the number of classifiers that are included in the ensemble. If the errors of the classifiers in the ensemble were uncorrelated, averaging over larger ensembles should improve the accuracy of the predictions, because the errors of a given classifier would be compensated by the correct predictions of other classifiers in the ensemble.

In practice, the ensemble members tend to make errors in the same examples. Nonetheless, in a wide range classification problems, the accuracy of parallel ensembles, such as bagging, improves with the number of classifiers that are included in the ensemble. However, larger ensembles have larger storage needs and longer times of prediction. To alleviate these shortcomings, different ensemble pruning methods can be used [4–13]. The goal of these methods is to reduce the memory requirements and to speed-up the classification process while maintaining or, if possible, improving the level of accuracy of the original ensemble. Most ensemble pruning methods replace the original ensemble by a representative subset of predictors. Besides needing less storage space and predicting faster, pruned subensembles can actually outperform the original classification ensembles from which they are extracted [5–8, 10].

Using a different approach, the time needed for prediction using a parallel ensemble, such as bagging [14], can be reduced by a dynamical pruning method called instance-based (IB) pruning [11]. In IB pruning the number of classifiers that need to be queried to estimate the final ensemble prediction is determined for each instance separately. Assuming that simple majority voting is used, it is possible to estimate the final decision by querying only a subset of the classifiers in the ensemble. Given a test instance that needs to be classified, the aggregation of the outputs of the ensemble members is halted when the probability that the remaining predictions do not change the current majority class is above a specified confidence level α. Since the overhead needed to determine whether the aggregation process should be halted is negligible, the reduction in the number of queries directly translates in a speed-up of the classification process. This method does not reduce the storage requirements, because all the classifiers in the original ensemble need to be available for potential queries. Since the differences in prediction are below the threshold $1 - \alpha$, the differences between the errors of the dynamically pruned ensemble and of the original ensemble are also necessarily below $1 - \alpha$. Therefore, the generalization performance of the ensemble is only slightly modified by IB-pruning.

The theoretical analysis of majority voting on which IB-pruning is grounded relies on the fact that in parallel ensembles the individual classifiers are generated under the same conditions and independently of each other. The goal of this investigation is to determine whether IB-pruning can be also used in sequential ensembles. When the ensemble is sequential, the classifier that is added at one point in the sequence depends on the classifiers that have been included in the ensemble up to that point. As a result, one introduces correlations among classifiers, which can result in biases in the estimation of the final ensemble prediction on the basis of the outputs of the initial classifiers in the sequence. The results of experiments on benchmark classification problems carried out in this investigation show that the biases introduced by IB-pruning can cause some distortions in the estimation of the error rate of the complete ensemble when ordered aggregation is used. By contrast, IB-pruning is remarkably effective when it is used to halt the aggregation process not in the complete ordered ensemble, but in the subensemble that is obtained by selecting the first $\approx 20\%$ classifiers in the reordered sequence. We conjecture that the reason for this different behavior is related to the properties of ordered bagging. The curves that trace the dependence of the error rate on the size of the ordered ensemble exhibit a minimum at intermediate ensemble sizes. This means that the first and the last classifiers included in the ordered bagging ensemble have rather different properties. As a matter of fact, the last classifiers that are included in the ordered sequence cause a deterioration instead of an improvement of the error rate. In consequence, estimations based on the first classifiers in the ensemble can be very different from the final decision, which takes into account all the classifiers in the ensemble. By contrast, in the second case, the test error rate monotonically decreases with the size of the ensemble, which implies that the trends detected in the output of the first classifiers tend to be reinforced by the subsequent predictions.

In summary, we propose a double pruning algorithm, in which dynamical IB-pruning is applied to a pruned bagging ensemble built with ordered aggregation. The method combines the advantages of pruning by ordered aggregation and instance-based pruning; namely the generalization performance of the ensemble is improved, the storage requirements are reduced, because only the classifiers in the pruned ensemble need to be stored in memory, and, finally, the efficiency classification process is significantly ameliorated, not only because the number of classifiers of the pruned ensemble is smaller, but also because IB-pruning speeds up the prediction of the class label for individual instances.

The paper is organized as follows: Section 2 provides a review of ordered aggregation. The dynamical pruning algorithm IB-pruning is described in Section 3. Section 4 summarizes the results of experiments on benchmark classification tasks that demonstrate the effectiveness of the double pruning algorithm proposed. Finally, the conclusions of this work are exposed in Section 5.

2 Ensemble pruning based on ordered aggregation

A possible approach to ensemble pruning is to select from the original ensemble a subset of representative classifiers whose combined performance is equivalent or better than the complete ensemble. There are two sources of difficulties in the realization of this goal. The first handicap is that the selection of classifiers has to be based on estimates on the training data. However, the objective is to identify a subensemble that has good generalization performance. Even if we can compute accurate estimates of the generalization accuracy on the basis of the training data only, finding the optimal subensemble is a computationally expensive problem that involves comparing all the possible $2^T - 1$ non-empty subensembles that can be extracted from the original ensemble. A feasible approach is to use a greedy strategy based on modifying the order in which classifiers are aggregated in the ensemble [6, 15, 10]. Starting from an initial pool of classifiers, in which no particular ordering for combination is specified, ordered aggregation builds a nested sequence of ensembles of increasing size by incorporating at each step the classifier that improves the performance of the enlarged ensemble the most. The first classifier in the sequence is generally the one with the lowest training error. From the subensemble $\mathcal{S}_{t-1}$ of size $t-1$, the subensemble of size $\mathcal{S}_t$ is constructed by incorporating a single classifier from the remaining pool of classifiers. This classifier is selected by maximizing a measure that is expected to be correlated with the generalization performance of the ensemble. The measures that are effective for this selection take into account the complementarity of the classifiers selected, not only their individual accuracy or their diversity. In this article we use boosting-based ordered bagging [15], which uses the weighted training error defined in boosting to direct the ordering process. 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 [16]. The classifier that minimizes the weighted training error is then incorporated into the ensemble. The results of extensive experimental evaluation using bagging to generate the initial pool of classifiers show that early stopping in the aggregation process allows to identify pruned ensembles, whose size is $\approx 20\%$ of the complete ensemble, which outperform bagging and retain baggings resilience to noise in the class labels of the examples.

3 Instance-based pruning

Consider a binary classification problem. Assume that we have built a parallel ensemble composed of T classifiers built independently of each other. Each classifier is induced from the same learning data by repeated applications of a learning algorithm that involves some form of randomization. Consider an arbitrary instance $\mathbf{x}$ that needs to be classified. Assume that only t classifiers have been queried, and that the current (partial) vote count is t_1 for class 1 and t_2 for class 2 ($t_1+t_2=t$). Without loss of generality we can assume that $t_1 \geq t_2$. Using the fact that the classifiers in a parallel ensemble are generated independently of each other, the probability that the class labels predicted by the subensemble of size $t<T$ and by the complete ensemble of size T coincide is [11]

$$\tilde{\mathcal{P}}(t_1,t,T) = \sum_{T_1=max\{t_1,1+\lfloor T/2\rfloor\}}^{T-t_2} \frac{(T-t)!}{(T_1-t_1)!(T_2-t_2)!} \frac{(t_1+1)_{T_1-t_1}(t_2+1)_{T_2-t_2}}{(t+2)_{T-t}} \tag{1}$$

where $T_1+T_2=T$, and $(a)_n = a(a+1)\cdots(a+n-1)$ is the Pocchammer symbol, or rising factorial, with a and n nonnegative integers. If it is acceptable that, with a small probability $1-\alpha$, the predictions of the partially polled ensemble and of the complete ensemble disagree, the voting process can be stopped when the probability (1) exceeds the specified confidence level α. The final classification is estimated as the combined decision of the polled classifiers only. In particular, the querying process can be halted after t classifiers have been queried, if the vector of class predictions of the current subensemble $t_1^\star(t;T,\alpha)$ is such that $\tilde{\mathcal{P}}(t_1^\star,t,T) \geq \alpha$. For an ensemble of $T=101$ classifiers and a confidence level $\alpha = 99\%$ the first few values of $t_1^\star(t;T=101,\alpha=0.99)/t$ are $6/6, 7/7, 8/8, 8/9, 9/10, 10/11, 10/12, 11/13, \ldots$

4 Experiments

In this section we perform experiments to determine whether IB-pruning can be used in combination with ordered bagging. In the first set of experiments IB-pruning is applied to a standard (randomly ordered) bagging ensemble. As expected, the results of these experiments confirm the effectiveness of IB-pruning in parallel ensembles. A second batch of experiments show that IB-pruning is not effective when applied to ordered bagging because of the differences between the classifiers that appear in the first positions in the ordered ensemble and those that appear in the last positions. Finally, IB-pruning applied to a pruned ensemble that is obtained by selecting the first $\approx 20\%$ classifiers in the ordered bagging ensemble. This last series of experiments illustrates the effectiveness of IB-pruning on the pruned ensemble in the problems investigated.

All the experiments are performed on twelve binary classification problems from the UCI repository [17]. The same learning setup is used to make comparisons possible. In all cases the results reported are averages over 10 independent 10-fold cross validation estimates. The protocol followed in each execution for a partition of the data into training and test is as follows: (i) Build a bagging ensemble composed of $T = 101$ CART trees [18] using the training set. The standard settings for the generation of the decision trees are used. The ordering of the initial ensemble is determined by the order of generation, which is random in bagging. (ii) Estimate the generalization error in the test set for the whole ensemble and for the first 21 trees in the randomly ordered ensemble. Apply IB-pruning to the complete ensemble using $\alpha = 99\%$ recording the test error and the average number of trees used to classify the instances. (iii) Modify the sequence of aggregation of the trees in the ensemble using boosting-based ordering [15]. This method is similar to boosting. However, instead of generating new classifiers at each step, one selects the classifier from the original ensemble that minimizes a weighted error on the training set. The weights of the instances in the formula for the weighted error are specified according to the prescription given by boosting. The test error for the ordered bagging using the first 21 trees of the ensemble. This value for the number of selected trees produces consistently good results in a wide range of datasets [10]. Apply IB-pruning to ordered bagging ensemble of $T = 101$ using $\alpha = 99\%$. Compute the average test error and the average number of classifiers used to classify the instances. (iv) Finally, apply IB-pruning to the first 21 trees of the ordered ensemble ($T = 21$ and $\alpha = 99\%$) recording the number of trees and classification error.

The results of applying IB-pruning to bagging are summarized in Table 1. For each dataset, the table shows the average test error for bagging (BAG101), bagging using the first 21 randomly generated classifiers (BAG21) and IB-pruning applied to the full bagging ensemble (IB-BAG101). The average number of trees used to classify each instance in IB-BAG101 is shown in the last column of the table. The corresponding standard deviations are displayed after the ± sign. These experiments confirm the results reported in [11]. Table 1 shows that the generalization error of a bagging ensemble with 101 trees is generally better than a bagging ensemble composed of 21 classifiers. This also confirms the observa-

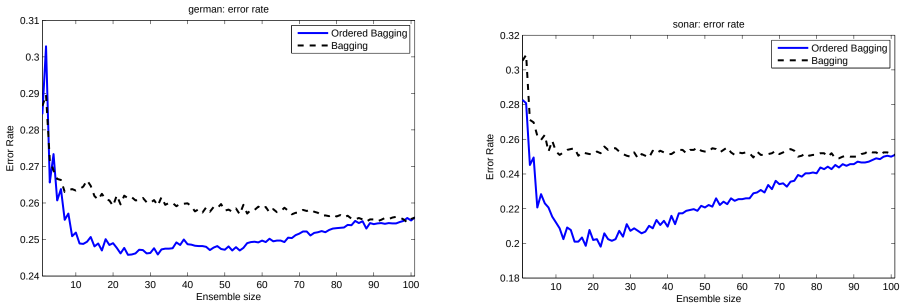

Fig. 1. Test error curves with respect to the number of classifiers for bagging and bagging ordered using boosting based ordering

Table 1. Results for bagging (best methods are highlighted in boldface)

	Test error			# trees IB
Problem	BAG101	BAG21	IB-BAG101	($\alpha = 99\%$)
australian	**14.5±3.8**	**14.5±3.8**	**14.5±3.8**	7.4±1.0
breast	**4.8±2.8**	**4.8±2.6**	**4.8±2.8**	8.6±1.3
diabetes	**24.9±3.9**	**24.9±4.0**	**24.9±3.9**	14.3±2.3
german	**25.6±3.1**	25.9±3.5	25.7±3.1	18.0±2.6
heart	19.6±8.0	19.9±8.0	**19.4±7.8**	18.5±4.8
horse-colic	17.8±6.3	17.9±6.0	**17.7±6.2**	9.9±3.0
ionosphere	**9.7±4.6**	9.8±4.5	**9.7±4.5**	8.6±1.9
labor	**13.4±12.8**	14.1±12.9	13.7±12.6	17.7±8.7
liver	**31.1±6.2**	31.9±6.9	**31.1±6.2**	21.1±5.2
sonar	25.1±9.7	25.1±9.2	**25.0±9.7**	19.4±5.4
tic-tac-toe	**1.6±1.3**	2.2±1.5	**1.6±1.3**	10.1±1.3
votes	**4.4±3.0**	**4.4±3.0**	**4.4±3.0**	6.1±0.3

tion that increasing the size of parallel ensembles in which the generation of the individual classifiers involves some form of randomization generally improves the generalization performance of the ensemble [19]. In contrast, when IB-pruning is used to determine when to stop querying for the classification of new instances, a performance comparable to BAG101 is achieved in the studied datasets using on average a fairly small fraction of the classifiers. In particular, the average number of trees that need to be queried in IB-BAG101 ranges from 6.1 for *Votes* to 21.1 for *Liver*.

Table 2 compiles the results of the application of IB-pruning to ordered bagging ensembles. The column labeled BAG101 displays the average and, after the ± symbol, the standard deviation of the test error rate obtained by a bagging ensemble composed of 101 trees. The second column presents the results of IB-pruning when applied to the complete ordered bagging ensemble (IB-OB101). The average number of trees used by IB-OB101 is given in the fifth column.

Table 2. Results for ordered bagging (best methods are highlighted in boldface)

	Test error				# trees IB ($\alpha = 99\%$)	
Problem	BAG101	IB-OB101	OB21	IB-OB21	IB-OB101	IB-OB21
australian	14.5±3.8	14.3±3.9	**13.7±3.9**	**13.7±4.0**	11.3±1.7	7.0±0.5
breast	4.8±2.8	4.5±2.6	4.1±2.6	**4.0±2.6**	8.7±1.2	5.9±0.3
diabetes	24.9±3.9	24.7±4.0	**24.3±3.9**	**24.3±3.9**	17.2±2.3	8.7±0.6
german	25.6±3.1	25.2±3.3	24.8±3.7	**24.7±3.8**	21.1±2.5	9.3±0.6
heart	19.6±8.0	18.9±7.6	**18.6±7.2**	**18.6±7.1**	20.2±4.0	9.4±1.0
horse-colic	17.8±6.3	17.5±6.2	**16.3±6.6**	**16.3±6.5**	9.8±2.1	6.6±0.7
ionosphere	9.7±4.6	8.5±4.4	**7.5±4.2**	**7.5±4.1**	10.9±2.0	6.7±0.6
labor	13.4±12.8	10.0±11.3	**8.3±10.0**	8.5±10.0	14.8±7.5	7.9±1.9
liver	31.1±6.2	29.5±6.2	**28.2±6.5**	28.4±6.7	28.0±4.6	11.8±0.9
sonar	25.1±9.7	23.6±9.5	**20.2±10.7**	**20.2±10.7**	26.1±5.3	11.2±1.3
tic-tac-toe	1.6±1.3	**1.4±1.2**	**1.4±1.2**	1.5±1.2	9.4±1.0	6.5±0.4
votes	**4.4±3.0**	**4.4±3.1**	4.7±3.2	4.6±3.2	7.0±0.8	5.6±0.3

The results for a pruned ensemble composed of the first 21 trees of the ordered bagging ensemble are given in the column labeled OB21. These results show that the performance of ordered bagging with 21 classifiers is better than that of full bagging for all the datasets investigated except for *Votes*. Ordered bagging has two advantages over bagging: faster classification, because only a small fraction ($\approx 20\%$) of the original classifiers is used, and, in general, better accuracy in the test set. Instead of using a fixed number of classifiers, IB-pruning individually determines the number of classifiers that are needed to estimate the complete ensemble prediction for each particular instance. When IB-pruning is used in conjunction with ordered bagging (column IB-OB101 in Table 2), the number of queried classifiers is generally lower than the 21 trees used in pruned bagging (OB21). However, it is over the number of elements queried by IB-pruning for randomly ordered bagging (right most column of Table 1). In addition, the accuracy improvement with respect to bagging is not as ample as the improvement of OB21 over BAG101. This poorer performance is a consequence of the fact that IB-OB101 is making inference about the predictions of the complete ensemble on the basis of the predictions of only the first classifiers in the ordered sequence. These classifiers follow a distribution that is different from the overall distribution of classifiers in bagging. These results can be understood analyzing the plots displayed in Fig. 1. The curves depicted trace the dependence of the test error with the size of the ensemble using bagging and ordered bagging for the classification tasks *German* and *Sonar*. This figure shows that by stopping the aggregation of classifiers at $\approx 20 - 30\%$ of the total number of elements in the ensemble, a significant reduction in the classification error is obtained. These error curves are representative of the general behavior of bagging and ordered bagging in all the datasets investigated.

In the final batch of experiments IB-pruning is applied to a pruned ensemble composed of the first 21 classifiers in ordered bagging. The results of these exper-

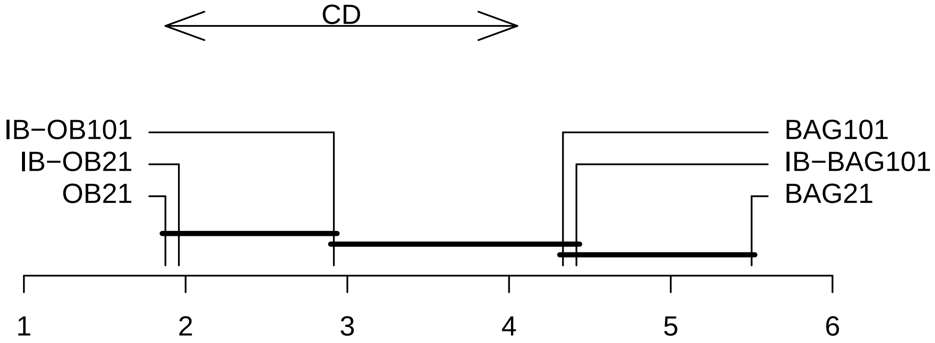

Fig. 2. Comparison of the different methods using the Nemenyi test. Classification systems whose performance are not significantly different according to this test (p-value < 0.05) are connected by a line segment in the diagram.

iments are displayed in the fourth column of Table 2 (IB-OB21). The last column shows the average number of trees used by IB-OB21. These results, show that the generalization error of IB-pruning applied to OB-21 is equivalent to that of OB21 in the problems analyzed. Small variations of one or two tenths of a percent point both positive and negative can be observed for some datasets. Therefore, the improvements obtained by IB-OB21 over complete bagging (BAG101) are of the same magnitude as the improvements obtained by the pruned ensemble obtained by early stopping in ordered aggregation (OB21). The number of trees that need to be stored in memory is also reduced from 101 to 21 trees. Finally, the average number of trees that need to be queried is further reduced by the application of IB-pruning to the pruned ensemble OB21. Specifically, IB-OB21 employs an average number of trees that ranges from 5.6 (*Votes*) to 11.8 (*Liver*). In summary, the application of IB-pruning to the pruned ensemble obtained from ordered aggregation (OB21) improves the accuracy and reduces the memory requirements of bagging as much as OB21 does. It has the additional advantage that it predicts even faster than OB21.

The overall generalization performance of the different ensemble methods in the classification tasks analyzed is compared using the methodology proposed by Demšar [20]. Fig. 2 displays the rank of each method averaged over the results in the different classification tasks. In this diagram, the differences between methods connected with a horizontal line are not significant according to a Nemenyi test (p-value$<$ 0.05). The critical difference (CD=2.2 for 6 methods, 12 dataset and p-value$<$ 0.05) is shown for reference. The best overall performance corresponds to OB21 and IB-OB21. The performance of these two methods is equivalent in the classifications tasks investigated. According to this test the performance of OB21 and IB-OB21 in terms of average rank is significantly better than standard bagging. The performances of the remaining methods are not significantly different from bagging.

5 Conclusions

In this article we propose to combine two existing pruning strategies to reduce the computational costs associated with the use of ensembles of classifiers for prediction and to improve their generalization performance. The first strategy selects a subset of complementary classifiers from the original ensemble to reduce the storage requirements, speed-up the classification process and improve the generalization performance. The algorithm is based on modifying the order of aggregation of the classifiers in the ensemble: a nested sequence of ensembles of increasing size is built by incorporating at each step the classifier that is expected to improve the classification error the most. The pruned subensemble is obtained by early stopping in the ordered aggregation process. In this article, the weighted error function used in boosting is used to guide the ordered aggregation. Nonetheless, other criteria based on the complementarity of the classifiers incorporated in the ensemble can also be used. Experiments in benchmark classification problems have shown that this strategy can improve the generalization error of bagging ensembles by keeping only $20-30\%$ of the classifiers, the first ones in the ordered sequence. The second strategy, instance-based pruning (IB-pruning), does not require any manipulation of the ensemble. It is applied dynamically when a new instance is classified. Using the fact that the classifiers in a parallel ensemble, such as bagging, are generated independently of each other, it is possible to compute the probability that the majority class obtained after having queried t classifiers will not change when the output the remaining classifiers becomes known. If this probability is above a specified confidence level α. The application of this method does not reduce the storage requirements (all the classifiers need to be stored in memory for potential queries), but it leads to a significant reduction in the average number of queries of ensemble classifiers without a significant modification of the accuracy of the ensemble.

In the problems investigated, IB-pruning applied to the original (randomly ordered) bagging ensemble obtains error rates similar to the complete ensemble and reduces the average number of queries more than the pruned ensemble that is built by selecting the first 21 classifiers in the ordered ensemble. However, its accuracy is lower than the pruned ensemble. When IB-pruning is applied to the complete ordered ensemble its generalization accuracy is better than the complete ensemble. Nevertheless, this accuracy is still inferior to the pruned ordered ensemble. This is due to the fact that the distribution of the predictions of classifiers that appear first in the ordered ensemble is different from the last classifiers included. Therefore, one of the basic assumptions of IB does not hold, leading to suboptimal performance. Finally, when IB-pruning is applied to the pruned ordered ensemble itself, a significant speed-up is achieved with a performance that is similar to the pruned ensemble and much better than bagging. The result is a double pruning algorithm that significantly improves the performance of bagging, achieving similar accuracy as pruned ordered bagging. Furthermore, it reduces the memory requirements, because only the classifiers that are selected in the pruned ordered ensemble need to be accessible for potential queries, and predicts much faster than standard bagging.

References

1. Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems: First International Workshop. (2000) 1–15
2. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40**(2) (2000) 139–157
3. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
4. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. of the 14th International Conference on Machine Learning, Morgan Kaufmann (1997) 211–218
5. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence **137**(1-2) (2002) 239–263
6. Martínez-Muñoz, G., Suárez, A.: Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications, Acta Press (2004) 258– 263
7. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proc. of the 21st International Conference on Machine Learning, New York, NY, USA, ACM Press (2004) 18
8. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: Ensemble diversity measures and their application to thinning. Information Fusion **6**(1) (2005) 49–62
9. Zhang, Y., Burer, S., Street, W.N.: Ensemble pruning via semi-definite programming. Journal of Machine Learning Research **7** (2006) 1315–1338
10. Martínez-Muñoz, G., Hernández-Lobato, D., Suárez, A.: An analysis of ensemble pruning techniques based on ordered aggregation. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(2) (2009) 245–259
11. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: Statistical instance-based pruning in ensembles of independent classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(2) (2009) 364–369
12. Latinne, P., Debeir, O., Decaestecker, C.: Limiting the number of trees in random forests. In: Multiple Classifier Systems. (2001) 178–187
13. Sharkey, A., Sharkey, N., Gerecke, U., Chandroth, G.: The Test and select approach to ensemble combination. In: Multiple Classifier Systems. (2000) 30–44
14. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
15. Martínez-Muñoz, G., Suárez, A.: Using boosting to prune bagging ensembles. Pattern Recognition Letters **28**(1) (2007) 156–165
16. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proc. of the 2nd European Conference on Computational Learning Theory. (1995) 23–37
17. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
18. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)
19. Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32
20. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30