Some Enhancements of Decision Tree Bagging

Pierre Geurts

University of Liège, Department of Electrical and Computer Engineering
Institut Montefiore, Sart-Tilman B28, B4000 Liège, Belgium
geurts@montefiore.ulg.ac.be

Abstract. This paper investigates enhancements of decision tree bagging which mainly aim at improving computation times, but also accuracy. The three questions which are reconsidered are: discretization of continuous attributes, tree pruning, and sampling schemes. A very simple discretization procedure is proposed, resulting in a dramatic speedup without significant decrease in accuracy. Then a new method is proposed to prune an ensemble of trees in a combined fashion, which is significantly more effective than individual pruning. Finally, different resampling schemes are considered leading to different CPU time/accuracy tradeoffs. Combining all these enhancements makes it possible to apply tree bagging to very large datasets, with computational performances similar to single tree induction. Simulations are carried out on two synthetic databases and four real-life datasets.

1 Introduction

The bias/variance tradeoff is a well known problem in machine learning. Bias relates to the systematic error component, whereas variance relates to the variability resulting from the randomness of the learning sample and both contribute to prediction errors. Decision tree induction [5] is among the machine learning methods which present the higher variance. This variance is mainly due to the recursive partitioning of the input space, which is highly unstable with respect to small perturbations of the learning set. Bagging [2] consists in aggregating predictions produced by several classifiers generated from different bootstrap samples drawn from the original learning set. By doing so, it reduces mainly variance and indirectly bias, and hence leads to spectacular improvements in accuracy when applied to decision tree learners. Unfortunately, it destroys also the two main attractive features of decision trees, namely computational efficiency and interpretability.

This paper approaches three topics on which improvements can be obtained with tree bagging either in terms of computation time or in terms of accuracy: discretization of continuous attributes, tree pruning, and sampling schemes. A very simple discretization procedure is proposed, resulting in a dramatic speedup without significant decrease in accuracy. Then a new method is proposed to prune an ensemble of trees in a combined fashion, which is significantly more effective than individual pruning. Finally, different resampling schemes are considered leading to different CPU time/accuracy tradeoffs.

D.A.Zighed, J.Komorowski, and J.Żytkow (Eds.): PKDD 2000, LNAI 1910, pp. 136-147, 2000.

The paper is organized as follows. Section 2 introduces bagging. Section 3 describes the proposed enhancements and Section 4 is devoted to their empirical study, reporting results in terms of accuracy, variance, tree complexity and computation time.

2 Bootstrap Aggregating (Bagging)

Let us denote by X the random input variable (attribute vector) and Y the (scalar) output variable, and by $P(X,Y)$ the probability distribution from which the data are sampled. We assume that the learning sample is a sequence ($LS = \{(x_1,y_1),...,(x_N,y_N)\}$) of independent and identically distributed observations drawn from $P(X,Y)$. Let us denote by $f_N(x)$ the (random) function which is produced by a learning algorithm in response to such a sample and by $\overline{f_N}(x) = E_{LS}\{f_N(x)\}$ the averaged model over the set of all learning sets of size N. Bias denotes the discrepancy between the best model (the one which minimizes a given loss criteria, also called the bayes model) and the averaged model while variance denotes the variability of the predictions with respect to the learning set randomness. Both, bias and variance, lead to prediction errors and thus should be minimized.

The averaged model $\overline{f_N}(x)$, by definition, has no variance (as it does depend on a particular learning set) and the same bias as the original model. So, if we could compute it, it would certainly have smaller prediction errors than a single model. In this context, bagging [2] has been suggested as a way to approximate this averaged model. As we do not have access to an infinite source of learning sets, the process of sampling from nature is approximated by bootstrap sampling from the original learning set. More precisely, starting from a learning set (LS) of size N, bagging consists in randomly drawing n subsamples of size N with replacement from LS. Let us denote by ls_1,...,ls_n these subsamples. Then from each ls_i, a model is learned denoted by $f_{ls_i}(x)$. Finally, the bagged model $f_{LS}(x)$ is obtained by aggregating the $f_{ls_i}(x)$. When output Y is discrete (classification), the final prediction is determined either by voting:

$$f_{LS}(x) = \{C_k | k = \arg\max_j (\sum_{i=1}^{n} \delta(f_{ls_i}(x), C_j))\}, \quad (1)$$

or by averaging class-conditional probability estimates if they are available:

$$f_{LS}(x) = \{C_k | k = \arg\max_j (\hat{P}_{LS}(C_j|x)) = \arg\max_j (\frac{1}{n}\sum_{i=1}^{n} \hat{P}_{ls_i}(C_j|x))\}, \quad (2)$$

where C_k denotes one of the classes and $\hat{P}_{ls_i}(C_k|x)$ the class-conditional probability estimates given by the i^{th} model. The two approaches have been shown to give very similar results.

For a fixed individual model complexity, this way of doing indeed reduces significantly variance while having little effect on the bias term. So bagging is mostly effective in conjunction with unstable predictors like decision trees which present high variance.

3 Proposed enhancements of decision tree bagging

While very effective, bagging in conjunction of decision trees learners destroys also the two main attractive features of decision trees, namely computational efficiency and interpretability. In this section we propose three enhancements of the tree bagging algorithm which try to improve its performance in terms of computation time or prediction accuracy only, not taking into account interpretability.

3.1 Median discretization

In another paper [8], we have investigated different ways to reduce the variance due to the discretization of continuous attributes in the context of top down induction of decision trees. It turns out from this paper that a very simple discretization algorithm which always chooses the median to split a local learning subset gives at least comparable results to the classical discretization algorithm. At the same time, the use of the median allows to reduce significantly variance of the probability estimates of the trees and computation times. However, we point out that the median comes with a loss of interpretability as the threshold is not even related to the class in the learning subset. While this loss of interpretability is a drawback in the context of single decision tree induction, it has no importance in the context of bagging.

Usually node splitting is carried out in two stages: the first stage selects for each input attribute an optimal test and the second stage selects the optimal attribute. In the case of numerical attributes, the first stage consists in selecting a discretization threshold for each attribute. Denoting by a an attribute and by $a(o)$ its value for a given sample o, this amounts to selecting a threshold value a_{th} in order to split the node according to the test $T(o) \equiv [a(o) < a_{\text{th}}]$. To determine a_{th}, normally a search procedure is used so as to maximize a score measure evaluated using the subset $ls = \{o_1, o_2, ..., o_n\}$ of learning samples which reach the node to split. Supposing that the ls is already sorted by increasing values of a, most discretization techniques exhaustively enumerate all thresholds $\frac{a(o_i)+a(o_{i+1})}{2}(i = 1...n-1)$. Denoting the observed classes by $C(o_i), (i = 1, \ldots, n)$, the score measures how well the test $T(o)$ correlates with the class $C(o)$ on the sample ls. In the literature, many different score measures have been proposed. In our experiments we use the following normalization of Shannon information (see [13] for a discussion):

$$C_C^T = \frac{2I_C^T}{H_C + H_T}, \tag{3}$$

where H_C denotes class entropy, H_T test entropy (also called split information by Quinlan), and I_C^T their mutual information.

Our modification of this classical discretization algorithm simply consists in evaluating for each numerical attribute only one threshold value, the one which splits the learning set into two sets of the same size. According to the previous notation, we can compute this threshold as $\frac{a(o_{n/2})+a(o_{n/2+1})}{2}$ if n is even

or $\frac{a(o_{(n+1)/2})+a(o_{(n+3)/2})}{2}$ if n is odd. We then split the node according to the pair attribute-threshold which gives the test receiving the best score.

The median discretization is of course faster than the classical one. First, we only have to compute the score for one threshold value when we need to do this computation n times in the classical algorithm. Second, we do not have to sort the local learning set for each numerical attribute. Indeed, there exist algorithms linear in the number of samples which is obviously better than the $N\log(N)$ order needed for sorting. Actually this second argument is not always relevant as it is necessary to sort the learning set only once for each attribute before any splits are made. However this pre-sorting has the disadvantage of needing a lot of memory space to store the sorted learning sets (pointers) and thus is not always possible to implement in the case of very large databases. In our implementation, we use pre-sorting to compute the median and so the difference between the two discretization algorithms is essentially due to the number of score computations. To give a first idea of the time which can be saved, we discretized 50 random samples of size 1000 from the Waveform database. Classical discretization took about 5000 ms while the median discretization only took 300 ms.

3.2 Combined pruning

Decision tree pruning. Like bagging, pruning is a way to handle the bias-variance tradeoff in decision tree induction. Pruning aims at cuting useless part of decision trees. By doing so, it reduces complexity significantly and variance to some extent, but it also increases bias. Thus, it generally improves only slightly interpretability and accuracy.

There are two ways to prune a tree : stop-splitting and post-pruning. The former approach consists in evaluating best test significance on the current learning set and decide whether to split or not the node according to this measure. One problem of this approach is that it often relies on a parameter, the significance threshold, and the optimum value of this parameter could be application dependant and difficult to appreciate. The latter approach consists first in building a full tree and then use cross-validation to prune useless parts of the tree in a bottom-up fashion. This approach is not parametric anymore but it is necessary to save some samples to form the validation set.

In our implementation of DT induction, stop-splitting is based on a hypothesis test. Indeed, assuming the independence of the test with respect to the classification in the learning sample, Kvålseth ([9]) has shown that the following quantity:

$$G^2 \triangleq 2N.\ln 2.\hat{I}_C^T,$$

follows a χ-square distribution with $(m-1)(p-1)$ degrees of freedom, where m is the number of classes and p is the number of test issues. Thus we choose not to split a node if the probability of G^2 being greater than the observed value is larger than an a priori fixed value α. To $\alpha = 1.0$ correspond fully developed trees and to $\alpha = 0.0$ correspond only trivial trees.

Post-pruning usually relies on a quality measure of trees where a parameter β weights complexity versus reliability of a tree. The quality measure we used is:

$$Q_\beta(\mathcal{T}, LS) \triangleq N\hat{I}_C^{\mathcal{T}}(LS) - \beta.\delta(\mathcal{T}),$$

where N is the number of learning states in the learning set LS, $\hat{I}_C^{\mathcal{T}}(LS)$ the information provided by the tree $\mathcal{T}$ about the classification in the learning set, $\delta(\mathcal{T})$ the tree complexity which is defined as the number of terminal nodes of the tree minus one. Then for increasing values of β from 0 to ∞, we compute the optimally pruned tree $\mathcal{T}_\beta$ which maximizes $Q_\beta(\mathcal{T}_\beta, LS)$. From the additive nature of the quality criterion, it follows that the β-optimal pruned trees form a nested sequence for increasing β (and thus there are at most k of them, if k is the number of test nodes) and that the sequence of trees can be computed by a simple recursive bottom up algorithm (see [13] for a complete discussion). This yields the sequence $S = \{(\beta_1, \mathcal{T}_1), (\beta_2, \mathcal{T}_2), \ldots, (\beta_k, \mathcal{T}_k)\}$ where $\beta_i < \beta_{i+1}$, $\mathcal{T}_1$ is the full tree, $\mathcal{T}_k$ the trivial tree and $\mathcal{T}_i$ $(1 < i \leq k)$ is the best pruned tree for $\beta_{i-1} < \beta \leq \beta_i$. Among this sequence of trees, we extract the one which gives the best result on the pruning cross-validation sample.

Pruning with bagging. In the context of DT ensembles, we could imagine two ways of decreasing the model complexity: first by removing some trees from the ensemble, second by pruning the constituting trees. The first approach has been successfully applied to boosted ensembles in [10] but we believe it is not likely to give good results with bagging as models are i.i.d. and generalization errors have been shown to be monotone functions of the number of aggregated models. To decrease constituting trees complexity, we could first apply the above pruning techniques individually to each tree of the ensemble, i.e. not taking into account their future averaging. However, because bagging reduces variance, the optimal complexity of averaged trees should be higher than the optimal complexity of a single tree induced from the same learning set. So individual pruning should yield trees which are not complex enough given the new bias/variance tradeoff resulting from averaging. So we propose here a new method to post-prune the trees from a bagged ensemble in a combined way.

The algorithm proceeds as follows. First, we compute for each tree $\mathcal{T}_j$ of the ensemble the sequence $S_j = \{(\beta_{j,1}, \mathcal{T}_{j,1}), (\beta_{j,2}, \mathcal{T}_{j,2}), \ldots, (\beta_{j,k_j}, \mathcal{T}_{j,k_j})\}$ according to the previously defined single tree post-pruning algorithm. We then let increase β from 0 to ∞ and get from each individual pruning sequence S_j the tree $\mathcal{T}_{j,i}$ such that $\beta_{j,i-1} < \beta \leq \beta_{j,i}$. This yields a sequence of pruned trees ensembles. Among this sequence, we select the ensemble which yields the best error rates on the validation set. Of course, only critical values of β (i.e. corresponding to a $\beta_{j,i}$) need to be considered and if there are n trees with maximum k test nodes each, the length of the sequence will be less than kn.

In our experiments, we will compare stop-splitting and individual postpruning in the context of bagging to combined pruning. We expect that pruning the ensemble in a combined way would give better results than individual pruning and also, because of the non zero variance, will be better than no pruning at all.

Table 1. datasets

Dataset	#Variables	#Classes	#Samples	#LS	#PS	#TS	domain
Omib	6	2	20000	16000	2000	2000	Power system [13]
Waveform	21	3	5000	3000	1000	1000	artificial [5]
Satellite	36	6	6435	3000	1435	2000	soil type recognition [11]
Pendigits	16	10	10992	5000	2494	3498	digit recognition
Dig44	16	10	18000	6000	3000	9000	digit recognition [11]
Vst	136	2	4041	2430	815	796	voltage stability [13]

3.3 Resampling method

In average, bootstrap samples leave out about 37% of the examples. So, a question which could be raised is what happens if we replace this "with replacement, full size" sampling with a "without replacement, smaller size" sampling. Theoretical studies of bagging (in [12] for bagging of linear models and more recently in [6] for non-linear models) have shown that without replacement sampling could give similar or better results than bootstrap sampling. In the mean time, the computation times could be reduced significantly as we build trees from smaller learning sets. In our experiments, we propose to study in an empirical way the effect of various resampling scheme on real problems in terms of accuracy and computation times improvements.

4 Empirical studies

In this section, we evaluate the proposed enhancements. We first consider two artificial problems where thorough experiments are pursued and then look at four real-life datasets. A description of the databases is given in Table 1. All input variables are numerical and the datasets were selected to provide large enough samples.

4.1 Artificial problems (Omib and Waveform)

Experimentation protocol. To evaluate variants of decision tree induction and tree bagging and be able to assess their variance, we carried out experiments in the following way. First, the database is split into three disjoint parts: a set used to pick random samples for tree growing (LS), a set used for cross-validation during tree pruning (PS) and a set used for testing (TS) (the divisions for each database are shown in table 1). Then 50 random subsets of size 1000 are drawn without replacement from the pool LS, yielding LS_1,LS_2,...,LS_{50}. For each method:

- A model $\mathcal{M}_i$ (either a single tree or a set of trees) is grown from each LS_i according to the studied variant.
- Average test set error rate $\overline{P}_e$ and complexity $\overline{C}$ of the 50 models are computed.

Table 2. Effect of median discretization

method	Omib				Waveform			
	$\overline{P_e}\%$	$\overline{C}$	$\overline{var}$	CPU	$\overline{P_e}\%$	$\overline{C}$	$\overline{var}$	CPU
Single trees (post-pruned)								
classic	11.20	67.6	0.0572	6	27.30	45.96	0.0434	16
median	10.39	103.92	0.0383	4	27.30	66.04	0.0382	7
Bagging, 10 trees (unpruned)								
classic	7.71	967.0	0.0158	21	20.39	1618.4	0.0172	87
median	7.55	2026.0	0.0134	8	20.61	3032.9	0.0160	27

– As proposed in [7], we estimate classification model variance by computing the variance of the class-probability estimates given by the model. To this end, we determine the conditional class-probability estimates $\hat{P}_{\mathcal{M}_i}(C|o)$ for each case o belonging to the test sample and the following quantity is computed:

$$\hat{Var}(\hat{P}_{\mathcal{M}_i}(C|.)) = \frac{1}{\#TS} \sum_{o \in TS} \{\frac{1}{50} \sum_{i=1}^{50} (\hat{P}_{\mathcal{M}_i}(C|o))^2 - (\frac{1}{50} \sum_{i=1}^{50} \hat{P}_{\mathcal{M}_i}(C|o))^2\}.$$

We also give the average CPU time[1] in seconds needed to build one model of each type. The complexity of a single tree is the total number of nodes and the complexity of a bagged ensemble is the sum of the complexities of the component trees. The conditional class-probability estimates for bagging are computed by averaging the conditional class-probability estimates of each tree:

$$\hat{P}_{\mathcal{M}_i}(C|o) = \frac{1}{n} \sum_{j=1}^{n} \hat{P}_{T_{i,j}}(C|o),$$

where $T_{i,j}, j = 1, ..., n$ are the trees constituting the model $\mathcal{M}_i$. The predictions on test samples are done according to these estimates (attributing the maximum probability class to each case).

Results

Discretization: median vs classical score. Table 2 presents results obtained with classical discretization and median discretization, on one hand with single decision tree induction and pruning, and, on the other hand with bagging of full trees. The median does not decrease or increase the error rates significantly but it decreases the variance especially with single trees. On the other hand, the complexity of the resulting trees are multiplied by a factor 2 (when we do not use pruning). Note here also the positive effect of tree bagging which decreases dramatically the variance and significatively the error rates. In terms of computation times, we observe that given our (non optimal) implementation, median

[1] The system is implemented in Common Lisp and runs on a SUN Ultrasparc 2 - $300MHz$ microprocessor.

Table 3. Effect of pruning

method	Omib				Waveform			
	$\overline{P_e}\%$	$\overline{C}$	$\hat{var}$	CPU	$\overline{P_e}\%$	$\overline{C}$	$\hat{var}$	CPU
$\alpha = 1.0$	7.71	967.0	0.0158	21	20.39	1618.4	0.0172	87
$\alpha = 0.05$	7.77	873.4	0.0156	21	20.21	1523.9	0.0167	86
$\alpha = 0.01$	7.96	660.9	0.0149	21	20.15	1208.4	0.0156	79
$\alpha = 0.005$	8.21	589.0	0.0145	18	20.05	1057.9	0.0153	76
$\alpha = 0.001$	8.37	469.0	0.0137	17	20.11	780.4	0.0137	70
$\alpha = 0.0005$	8.54	429.3	0.0139	17	20.15	688.4	0.0127	66
$\alpha = 0.0001$	8.87	353.6	0.0134	16	20.35	535.6	0.0115	59
individual pruning	7.90	659.9	0.0143	30	20.03	668.4	0.0122	91
combined (classic)	7.67	831.2	0.0153	32	20.02	777.4	0.0134	93
combined (median)	7.51	1557.5	0.0123	17	19.79	1296.4	0.0113	32

Table 4. Effect of resampling method (with median and combined pruning)

method	Omib				Waveform			
	$\overline{P_e}\%$	$\overline{C}$	$\hat{var}$	CPU	$\overline{P_e}\%$	$\overline{C}$	$\hat{var}$	CPU
replacement	7.51	1557.5	0.0123	17	19.79	1296.4	0.0113	32
$p = 0.1$	10.27	344.9	0.0112	8	20.96	205.4	0.0103	8
$p = 0.3$	8.19	807.1	0.0108	12	20.17	568.3	0.0102	18
$p = 0.5$	7.91	1138.5	0.0120	15	20.11	817.3	0.0099	29
$p = 0.7$	7.65	1506.8	0.0140	18	20.30	1284.2	0.0122	35

makes bagging about three times faster than classical discretization although the resulting models are two times bigger.

Stop-splitting vs individual pruning vs combined pruning. To assess the effect and usefulness of pruning in the context of bagging, we made experiments using stop-splitting with different values of α, with individual tree pruning, and two versions of combined tree pruning (classical discretization and median discretization). These results are summarized in Table 3.

Results are different from one dataset to the other. On the Omib dataset, it appears that the more complex the trees are, the better are the error rates. On the Waveform database, we observe an optimal value of α smaller than 1.0 which justifies pruning in this case. However, the interest of combined pruning with respect to individual pruning does not yet appear from these experiments, as it gives more complex trees without really improving error rates. Nevertheless, together with median discretization it allows to decrease error rates and variance.

With replacement vs without replacement. Table 4 summarizes several experiments with different resampling schemes. The one called *replacement* is the classical bootstrapping method. The other ones are sampling without replacement and with learning sets of size p times the original learning set size. All experiments use median discretization and combined pruning.

Results are very interesting. First we note a decrease of variance as we reduce the fraction p. Of course, this decrease of variance comes with an increase of bias as the trees are much smaller. We thus observe a bias/variance tradeoff

Table 5. Results on real-life datasets

method	Satellite			Pendigits			Dig44			Vst		
	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU
Single trees (post-pruned)												
classic	13.95	163	27	7.52	363	19	16.12	455	127	11.81	21	178
median	15.5	269	19	11.29	893	17	18.378	601	45	14.32	331	59
Bagging, 25 trees, classical dis.												
$\alpha = 1.0$	11.24	9465	465	5.36	7413	305	9.71	21746	2010	8.48	4868	2332
ind. prun.	12.54	3074	492	5.47	6593	358	10.41	11398	2081	8.67	1325	2362
comb. prun.	11.44	6689	501	5.38	7042	343	9.71	19030	2109	8.12	3022	2386
Bagging, 25 trees, median dis.												
comb. prun.	11.13	11392	318	6.14	15465	271	10.47	39862	767	9.27	5476	916
Without replacement, combined pruning, classical dis.												
$p = 0.1, n = 10$	15.04	624	42	9.56	877	46	14.6	1581	135	12.59	282	125
$p = 0.1, n = 25$	13.54	1631	106	8.89	2247	111	12.75	4360	330	10.84	721	293
$p = 0.3, n = 25$	12.10	3579	261	6.38	4275	216	10.52	10057	956	8.89	1915	972
$p = 0.5, n = 25$	11.73	5741	397	5.82	6054	296	10.06	15219	1588	8.40	2513	1730

regulated this time by p. From this tradeoff results an optimal value of p of 0.5 on the Waveform database while on the Omib database the optimal value seems to be the greatest. These experiments show that it is possible to tune this parameter, on one hand to get better results and on the other hand to decrease the computation times. One other interesting fact is that, whatever the dataset, using 10 times smaller learning sets to build 10 decision trees is better (slightly for Omib, significantly for Waveform) than building a big one using the whole learning set. While CPU times are similar, building a big tree should be more demanding in terms of resources.

4.2 Real-life datasets

We also have pursued experiments on four real-life datasets (the last four datasets of Table 1). Each database was split into three disjoint parts: a learning set (LS), a pruning set (PS) and a test set (TS). Results are summarized in Table 5 (from top to bottom):

- A single decision tree was built from LS, then post-pruned using PS with classical and median discretization.
- Using the same 25 bootstrap samples, we built 25 fully developed trees ($\alpha = 1.0$) with classical discretization, we pruned them individually, then in a combined way and tested the three bagged sets of trees using class-probability estimates averaging. We also report the result of the same experiment with median discretization in the context of combined pruning only.
- We made as well experiments with sampling without replacement using $p = 1.0$ and 10 samples and $p = 0.1, 0.3, 0.5$ and 25 samples, each time with combined pruning and classical discretization.

Table 6. Random splitting

method	Satellite			Pendigits			Dig44			Vst		
	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU	$\overline{P_e}\%$	$\overline{C}$	CPU
Single trees (post-pruned)												
classic	19.85	374	6	14.17	987	6	24.14	1187	36	15.61	146	30
median	17.51	593	5	15.89	2232	8	29.66	2995	18	16.57	176	10
Bagging, 25 trees, combined pruning												
classic	10.68	12552	101	4.71	17122	130	7.33	35409	384	9.18	4014	335
median	11.59	23489	102	4.95	43810	178	8.82	80787	368	10.68	5498	132

Results on each dataset are summarized in terms of error rates, complexity and CPU times in seconds (which include building and pruning). When bagging was used, displayed values are the average of ten experiments with different sets of bootstrap samples. We comment here these results separately in terms of the three enhancements.

Median discretization. On all datasets, median discretization gives significantly worse error rates than classical discretization while building single trees. On the other hand, in conjunction with bagging, median still allows to obtain competitive results with respect to the classical discretization. Anyway the main interest of median discretization is the saving of time which is already significant on the Satellite and Pendigits datasets but becomes very impressive on the other two datasets (we gain a factor three). The smaller gain in computation time with the first two examples can be explained by the fact that on these datasets numerical attributes are integer valued between 0 and 256 and so the number of discretization thresholds to evaluate is much smaller than (and not related to) the number of samples in the learning set.

Since the naive median discretization works well with bagging, we pursued some more experiments with an even more naive splitting procedure in order to see how far we can go. To select a test, we first draw randomly an attribute, then compute the threshold. To ensure not to choose attribute too much unrelated to the classification, this procedure is repeated until we find an attribute which yields a score greater than an a priori fixed threshold (here 0.1). Table 6 shows results obtained with this random splitting on the four datasets, first with single trees, then with bagging of 25 trees (10 experiments). While random splitting deteriorates significantly error rates in the case of single trees, we obtain better results with bagging on the first three domains with a strong reduction of computation times but an increase of complexity. Our feeling at this point is that, in the context of bagging, the optimality of node splitting can be very much relaxed, thus allowing to decrease very strongly computation times.

Combined pruning. From Table 5, it is clear that individual pruning tends to produce trees which are not complex enough given the reduction of variance due to bagging. On the other hand, combined pruning always decreases complexity (by 20% on average) without notable change in accuracy with respect to full trees. However, individual pruning still could be preferred as it decreases much

more complexity (50% on average) without very important decreases in error rates.

Resampling. without replacement bagging with $p = 0.5$ gives similar results than with replacement bagging but is better in terms of CPU times because learning sets and thus trees are a bit smaller. On the other hand, the sampling fraction p allows to make a tradeoff between ensembles accuracy (and ensembles complexity) and computation times. As on the two artificial problems, averaging ten small trees sometimes is better than building a single big one (as on Dig44) but we see here that it can also significantly reduces accuracy (as on Pendigits). On the other hand, even if building ten small trees is done in the same amount of time as building a big one, the latter is more demanding in terms of memory space and real computation times should be higher in this case.

5 Discussion

In this paper we have investigated three enhancements of decision tree bagging on which we give the following conclusions and future work directions:

Discretization. While the "median" discretization method does not work generally as good as the classical one when building single decision trees, the loss it introduces vanishes when we use it in conjunction with bagging. Our feeling is that bagging has the effect of counterbalancing the sub-optimality of the node-splitting procedure used to build the individual trees. Thus, there is definitely an opportunity here to reduce computing times of tree bagging, without sacrificing accuracy, especially when dealing with large datasets. Although we did mainly focus in this paper on the discretization part of node splitting, our results with random-splitting suggest that further simplification of the node-splitting algorithm can be obtained without notable degradations of accuracy. This opens a new field of investigations to design ad hoc tree growing procedures in the context of bagging.

Pruning. Breiman [4] and others [1] have come to the conclusion that unpruned trees were better than individually pruned trees in the context of bagging, since bagging reduces variance only. However, our experiments have shown that in some problems (e.g. the Waveform database) pruning the trees in a combined way taking into account the model aggregation step, may lead to more accurate models than fully grown trees. We thus believe it is a good idea to use it in conjunction with tree bagging since it never decreases significantly performances in terms of error rates, and it is still very efficient in terms of computing time.

Resampling scheme. Our experiments with different resampling schemes agree with the theoretical developments in [6]. Sampling without replacement can give similar results than sampling with replacement but with decreased computation times. From these experiments comes the idea of alternative uses of bagging already proposed in [3], for example as a way to handle very large datasets or to learn incrementally a model. In the first goal, we could randomly partition

a large database into several small parts (say ten times smaller than the whole set), build a tree from each part and then average the resulting models. Our experiments and the one in [3] show that this procedure would work at least as well as building only one tree. On the other hand, the appropriate use of the scheme in an incremental and adaptive way to handle time-varying data has still to be investigated, but seems to be a very promising direction of complementary work.

Allover, the various improvements suggested in this paper can be combined to make tree bagging a very attractive procedure in the context of data mining of very large databases, more accurate than classical tree induction and of the same or even higher computational efficiency and scalability.

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms : Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.
2. L. Breiman. Bagging predictors. Technical report, University of California, Department of Statistics, September 1994.
3. L. Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36:85–103, 1999.
4. L. Breiman. Using adaptive bagging to debias regressions. Technical report, Statistics Department, University of California, Berkeley, February 1999.
5. L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth International (California), 1984.
6. J. H. Friedman and P. Hall. On bagging and nonlinear estimation. Technical report, Statistics Department, Standford University, January 2000.
7. J.H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.
8. P. Geurts and L. Wehenkel. Investigation and reduction of discretization variance in decision tree induction. In *Proc. of the 11th European Conference on Machine Learning (ECML-2000), Barcelona*, pages 162–170, May 2000.
9. T.O. Kvålseth. Entropy and correlation : Some comments. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-17(3):517–519, 1987.
10. Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In Morgan Kaufmann, editor, *Proc. of Fourteenth International Conference on Machine Learning (ICML-97)*, 1997.
11. D. Michie and D.J. Spiegelhalter, editors. *Machine learning, neural and statistical classification.* Ellis Horwood, 1994.
12. Peter Sollich and Anders Krogh. Learning with ensembles : How over-fitting can be useful. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 190–196. MIT Press, 1996.
13. L. Wehenkel. *Automatic learning techniques in power systems.* Kluwer Academic, Boston, 1998.