Ensemble-Trees: Leveraging Ensemble Power inside Decision Trees

Albrecht Zimmermann

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
Albrecht.Zimmermann@cs.kuleuven.be

Abstract. Decision trees are among the most effective and interpretable classification algorithms while ensembles techniques have been proven to alleviate problems regarding over-fitting and variance. On the other hand, decision trees show a tendency to lack stability given small changes in the data, whereas interpreting an ensemble of trees is challenging to comprehend. We propose the technique of *Ensemble-Trees* which uses ensembles of rules *within* the test nodes to reduce over-fitting and variance effects. Validating the technique experimentally, we find that improvements in performance compared to ensembles of pruned trees exist, but also that the technique does less to reduce structural instability than could be expected.

1 Introduction

Decision trees have been a staple of classification learning in the machine learning field for a long time [1–3]. They are efficiently inducible, relatively straightforward to interpret, generalize well, and show good predictive performance. All of these benefits, however, are diminished by the fact that small changes in the data can lead to very different trees, over-fitting effects, and varying accuracies when employed to classify unseen examples.

To deal with problems in classification learning such as over-fitting, ensemble methods have been proposed. While the details vary, the main mechanism remains the same: a so-called "weak" learner, such as the afore-mentioned decision trees, is trained on a particular sample distribution of the available training data. The subset is then altered, possibly according to the performance of the learned classifier, and another classifier trained. In the end, an ensemble of weak classifiers is combined to classify unseen examples. The use of ensembles results in more stability w.r.t. predictive accuracy since over-fitting effects of several classifiers are balanced against each other. This advantage is however traded off both against difficulties with interpreting the complete classifier, since it can easily number tens of weak classifiers, and against increased training times.

To improve on the weaknesses of both decision trees and ensembles, we propose to essentially invert the ensemble construction process: Instead of inducing

complete classical decision trees and combining those into a classifier, our solution – *Ensemble-Trees* – induces small ensembles of rules as tests in each inner node. The expected benefits would be relatively small and interpretable trees compared to ensembles of trees, and better structural stability w.r.t. changes in the data and higher predictive accuracy compared to classical decision trees.

The paper is structured as follows: in the following section, we will develop our approach, going over the main details of decision tree induction, k-best rule mining, and combination of rule predictions in the final tree. In Section 3, the connection to existing works will be made, followed by an experimental evaluation of our hypotheses regarding the behavior of *Ensemble-Trees* in Section 4. Finally, we discuss the results of this evaluation, and conclude, in Section 5.

2 From Decision Trees to *Ensemble-Trees*

Decision trees in different forms [1–3] have been one of the biggest success stories of classification learning, with Quinlan's C4.5 one of the most effective symbolic classification algorithms to date. Even when adapted to allow for regression or clustering, the main algorithm for inducing a decision tree stays more or less the same: For a given subset of the data, a "best" test according to an interestingness measure is selected, the data split according to this test, and the process repeated on the two subsets thus created. The interestingness measures used (e.g. information gain, intra-class variance) typically reward tests that make the subsets more homogeneous w.r.t. a target variable (e.g. the class label, or a numerical value). In general, a test can have any number of outcomes but for reasons that will be discussed later we limit ourselves to binary tests. The pseudo-code of a decision tree induction algorithm is shown as Algorithm 1.

Algorithm 1 The General Decision Tree Algorithm

Input: a data set $\mathcal{D}$, interestingness measure ϕ, minimum leaf size m
Output: a decision tree T
T =make_node($\mathcal{D}, \phi, m$)
return T

make_node($\mathcal{D}, \phi, m$)
Find best test t on $\mathcal{D}$ according to ϕ
$\mathcal{D}_{yes} = \{d \in \mathcal{D} \mid t \text{ matches } d\}$
$\mathcal{D}_{no} = \{d \in \mathcal{D} \mid t \text{ does not match } d\}$
if $|\mathcal{D}_{yes}| \geq m \wedge |\mathcal{D}_{no}| \geq m$ **then**
  n_{yes} =make_node($\mathcal{D}_{yes}, \phi, m$)
  n_{no} =make_node($\mathcal{D}_{no}, \phi, m$)
  return node(t, n_{yes}, n_{no})
else
  return leaf(prediction($\mathcal{D}$))
end if

Note that the best test as well as the prediction given in a leaf is not specifically instantiated in this pseudo- code and depends on the data mining task at hand. In the popular C4.5 approach, the leaf prediction is the majority class in $\mathcal{D}$, and a binary test takes the form of the attribute-value pair that leads to the largest increase in homogeneity, measured by information gain. This means, however, that small changes in the data can have a strong effect on the composition of the tree. Consider ϕ to be information gain, and a data set with the following characteristics:

index	A_1	A_2	...	class
1	v_2	v_1	...	+
2	v_1	v_2	...	−
3	v_2	v_1	...	+
4	v_1	v_1	...	+
5	v_1	v_2	...	−
6	v_1	v_2	...	−
7	v_1	v_1	...	−
8	v_2	v_1	...	+
...	...	...	...	...

Obviously, a change in either the value of A_1 in instance 4, or the value of A_2 in instance 7 would improve the strength of these attributes as a test, respectively. Similarly, removal of one of those two instances, e.g. as part of a cross-validation, changes which attribute is chosen. Since the choice of test affects how the data is split, this effect ripples down through the rest of the tree. Assuming that A_1 and A_2 perform equally well on the rest of the data, the decision between them comes down to an arbitrary tie-breaker, giving the tree potentially rather different composition – and performance.

While attempts have been made to alleviate this problem, using linear combinations of attributes-value pairs in nodes [4], *option trees* [5], which in case of tests with very similar performance keep *all* of them and sort the data down all paths, or *random forests* [6], which randomly sample a subset of attributes as candidates for tests, these techniques strongly decrease comprehensibility of the tree, trading it off against better performance.

2.1 Ensembles of Rules and Efficient k-best induction

Ensembles of conjunctive rules, used together with conflict-resolution techniques such as (weighted) voting [7], or double induction [8] have been shown to perform better than decision lists, balancing the bias/over-fitting of single rules.

If this stabilizing effect of using a (small) ensemble of rules is brought to bear for tests inside a decision tree node, we expect that splits effected by those ensembles are less susceptible to changes in the data than splits effected by individual tests. This in turn should lead to trees having rather similar composition and structure, and ideally also similar rule ensembles in the test nodes. This also explains the name of our technique, *Ensemble-Trees*, since the tree-structure serves to tie the smaller ensembles together.

Additionally, over-fitting effects of individual tests can be expected to be reduced, potentially removing the need for post-pruning. Using conjunctive rules as tests always leads to binary splits, since a conjunction can be either *true* or *false*, thus the formulation of a test as binary in the general algorithm.

Using a branch-and-bound search based on the pruning techniques for convex interestingness measures (such as information gain, and intra-class variance) pioneered by Morishita and Sese [9], it is possible to efficiently induce the k best conjunctive rules w.r.t. the measure used. The branch-and-bound technique is based on calculating an upper bound for each rule encountered during enumeration. Only rules whose upper bound exceeds the kth-best score at the time of potential specialization *are* actually specialized. For further details, we refer the reader to Morishita and Sese's work.

Thus, the generic "Find best test" can be instantiated by "Find the k best rules". Since conjunctive rules induced in this way are quantified w.r.t. a *target value* such as the class label, the split becomes more complex than the straightforward "match" used in the general algorithm. We will discuss the majority voting technique used in our approach in the next section.

2.2 Majority Splitting

Splitting the data according to a single attribute-value pair is straight-forward in that the pair either matches an instance, or does not match it, and the class prediction is implicit in how the distribution of class labels changes in the subsets produced. We adopt the convention that the left branch of a test node is the "yes" branch, i.e. instances that are matched by a test are sorted to the left, unmatched instances to the right.

Once several tests are used to split the data, the situation becomes a bit more complex however. Consider the following three rules for a binary prediction problem:

1. $A_1 = v_1 \wedge A_2 = v_1 \Rightarrow +$
2. $A_3 = v_1 \wedge A_4 = v_1 \Rightarrow -$
3. $A_1 = v_1 \wedge A_5 = v_2 \Rightarrow -$

and a subset of instances:

index	A_1	A_2	A_3	A_4	A_5	class
1	v_1	v_1	v_2	v_1	v_1	$+$
2	v_1	v_1	v_1	v_1	v_1	$+$
3	v_1	v_2	v_1	v_1	v_2	$-$
4	v_1	v_1	v_1	v_2	v_2	$-$
5	v_2	v_1	v_2	v_1	v_1	$+$

Considering the first instance, we see that rule 1 matches, therefore advocating that the instance be sorted to the *left*. Rules 2 and 3 do not match, thus advocating to sort the instance to the *right*. Since they form the majority, it would be sorted to the right. However, their prediction is *opposite* to the prediction of the first rule. Therefore, the match/non-match of rules 1, and rules 2 and

3 have a different semantic interpretation. In the interest of making class distribution more homogeneous to facilitate classification, it is intuitive to interpret the non-match of rules 2 and 3 as in effect advocating that instance 1 be sorted to the *left*. All three rules are then in agreement and instance 1 is sorted left.

The convention that matched instances are sorted to the left therefore is transformed into the convention that instance that are matched by rules *that predict the first class value* (or not matched by rules predicting the second class label) are sorted to the left, while in the opposite case being sorted to the right. There is of course the alternative of having three branches – one for instances for whom the first class is predicted, one for those with class two, and one for unmatched instances, which could be evaluated in an extension of our technique.

Looking at instance 2, we observe that it is matched by both rules 1 and 2, garnering one vote for being sorted into either direction. Rule 3 acts as a tie-breaker in this case, leading to instance 2 being sorted left. This example illustrates that the k that governs the amount of rules used in a test node should be odd so that ties do not have to be broken arbitrarily.

2.3 Inducing Ensemble Trees

Having covered the basic ingredients for inducing *Ensemble-Trees* in the preceding sections, Algorithm 2 shows the pseudo-code of the complete algorithm.

Note, that the Ensemble Tree algorithm includes another pre-pruning/stopping criterion: If it is not possible to find the user-specified amount of rules, not enough different splitting tests improving homogeneity can be induced, and the node is turned into a leaf. This can occur since typically k-best interestingness mining rejects rules which have the same score as one of their generalizations, since this translates into conjuncts that do not add any information. Similarly, and additionally, to the minimum size of a leaf m, this criterion pre-prunes test nodes that are not supported by enough data to be expected to generalize well.

3 Related Approaches

As mentioned above, *Ensemble-Trees* can be considered as extensions of classical decision trees such as the trees induced by the C4.5 algorithm, in that the main difference lies in the replacement of the attribute-value pairs in test nodes by small ensembles of conjunctive rules. Our approach is not the first to attempt and modify the tests in nodes however.

3.1 Decision Trees With Complex Tests

To our knowledge the first that recognized the problem of similarly performing tests, and proposed a solution in the form of *option trees* were Kohavi and Kunz [5]. Their solution consists of not choosing one out of several tests arbitrarily but instead creating "option nodes" having children nodes including all of these tests, and splitting the subset at each node. For classification, unseen examples are propagated downwards through the tree, and in case of an option node, the label predicted by the majority of child nodes chosen. While their approach

Algorithm 2 The Ensemble Tree Algorithm

Input: data set $\mathcal{D}$, interestingness measure ϕ, minimum leaf size m, number of rules k
Output: an Ensemble Tree T
T =make_node($\mathcal{D}, \phi, m, k$)
return T

make_node($\mathcal{D}, \phi, m, k$)
Attempt to find k best conjunctive rules $\mathcal{R} := \{r_i\}$ on $\mathcal{D}$ according to ϕ
if $|\mathcal{R}| < k$ **then**
  return majority class in $\mathcal{D}$
end if
$\mathcal{D}_{left} = \{d \in \mathcal{D} \mid$ at least $\lceil k/2 \rceil$ rules vote d is sorted left$\}$
$\mathcal{D}_{right} = \{d \in \mathcal{D} \mid$ at least $\lceil k/2 \rceil$ rules vote d is sorted right$\}$
if $|\mathcal{D}_{left}| \geq m \wedge |\mathcal{D}_{right}| \geq m$ **then**
  n_{left} =make_node($\mathcal{D}_{left}, \phi, m, k$)
  n_{right} =make_node($\mathcal{D}_{no}, \phi, m, k$)
  return node($\mathcal{R}, n_{left}, n_{right}$)
else
  return leaf(majority_class($\mathcal{D}$))
end if

improved significantly on the accuracy of C4.5 and an ensemble of bagged trees, the option trees themselves turned out to easily consist of one thousand nodes or more, comparable to the accumulated size of 50 bagged trees.

In [4], the authors suggest an algorithm consisting of a mixture of uni-variate (classical) decision tree tests, linear and non-linear combinations of attributes, using neural networks. The resulting trees are reported to be smaller than uni-variate trees and to improve on the accuracy in several cases. Obviously however, such combinations of attributes will be harder to interpret for end users.

In [10] and [11], data mining techniques were employed to mine complex fragments in graph- or tree-structured class-labeled data, which are then used as tests in a decision tree classifier. The motivation in these works is similar to ours in that the need for complex tests is acknowledged. Both techniques limit themselves to a single test in each node, however, not attempting to correct over-fitting effects during the induction of the trees.

3.2 Ensembles of Trees

Ensemble methods have been used with decision trees before, with the trees acting as the weak classifiers in the ensemble scheme. One of the best-known ensemble techniques, Bagging [12], was proposed specifically with decision trees as weak classifiers. In bagging, repeated boot-strap sampling of a data set is performed, decision trees learned on each of these samples, and their predictions combined using a majority vote. While Bagging helps in reducing over-fitting effects, the fact that instead of one tree several are created can be an impediment to interpreting the resulting classifiers.

The BAGGING idea is taken one step further in the construction of RANDOM FORESTS [6]. Not only are instances resampled but for each test the set of attributes that are evaluated as potential tests is chosen randomly. These two effects, in addition to using the trees unpruned, essentially ensures that nothing beyond attribute-interactions can be understood when interpreting the final classifier – trading off accuracy against comprehensibility even further.

BOOSTING, as embodied for instance by the well-known ADABOOST system [13] on the other hand, iteratively induces weak classifiers on data sets whose distribution is altered according to the performance of preceding classifiers. Mostly, this takes the form of resampling misclassified instances, or re-weighting them. While boosting technique can be proven to approximate the actual underlying classification function to an arbitrary degree, the resulting ensemble is again rather difficult to interpret, especially given the changes in the underlying distribution that are effected during the learning process.

3.3 ART – Changing Distributions

Finally, the ART approach [14] is related to our work in the sense that it uses a "classical" mechanism for changing underlying class distributions – *sequential covering* – in a novel way, inducing several rules on each subset. The stated intentions are the same as ours but the resulting classifier is rather different. We believe that the main insight (one that the authors did not make explicit) is that existing techniques for creating subsets of the data to mine patterns or rules on, can be used in far more general ways than have been explored so far.

4 Experimental Evaluation

An experimental evaluation is used to answer

Q1 How well *Ensemble-Trees* are suited to the classification task, especially in the absence of post-pruning.

Q2 Whether the use of ensembles of rules as tests in the inner nodes leads to more stable trees in the presence of changes in the underlying data, and whether the resulting trees are smaller (and better interpretable) than ensembles of trees.

We use several UCI data sets to compare *Ensemble-Trees* to the WEKA [15] implementations of C4.5, BAGGING, and ADABOOST, each with C4.5 as the weak classifier.

Since we limit ourselves to nominal attribute values in this work, numerical attributes were discretized, using ten equal-width bins. With the branch-and-bound technique used for inducing the rule ensembles limited to binary classes, we used only data sets with two labels. However, given that every classification problem can be translated into a number of *one-against-one*, or *one-against-all* problems, we do not see this as a significant drawback of our approach.

We performed experiments with the minimum leaf size parameter m set to $2, 3, 4, 5, 10$ and in case of large data sets, 10% of the training data. As the interestingness measure ϕ, information gain was used. BAGGING and ADABOOST were set to 10 iterations of inducing weak classifiers, and we built *Ensemble-Trees*

with $k = 3$, and $k = 5$, respectively. AdaBoost was used both in the resampling and the re-weighting mode. Decision tree and ensemble method results are reported using pruned trees while *Ensemble-Trees* are always unpruned.

4.1 Predictive Accuracy

Predictive accuracy was evaluated using a class-validated 10-fold cross-validation, with the folds being the same for each method. Table 1 reports mean and standard deviation for a minimum leaf size $m = 5$, except for the Trains data set, where $m = 2$. While details vary, the main trends can be observed for all minimum leaf sizes evaluated in the experiments.

Table 1. Predictive accuracies for decision trees/*Ensemble-Trees*, minimum leaf size 5

Data set	C4.5	$ET_{k=3}$	$ET_{k=5}$	Bagging	AdaBoost$_{RS}$	AdaBoost$_{RW}$
Breast-Cancer	$73.42 \pm 5.44\bullet$	78.69 ± 4.34	80.14 ± 6.16	73.77 ± 6.98	$67.09 \pm 10.10\bullet$	$66.77 \pm 6.81\bullet$
Breast-Wisconsin	94.56 ± 2.93	95.28 ± 1.35	95.14 ± 1.80	95.42 ± 2.76	96.28 ± 1.81	95.99 ± 1.14
Credit-A	84.20 ± 2.93	85.51 ± 2.56	85.51 ± 2.56	85.22 ± 2.35	82.75 ± 3.45	$82.03 \pm 4.44\bullet$
Credit-G	$71.90 \pm 3.96\bullet$	80.33 ± 2.00	79.10 ± 5.09	$74.40 \pm 4.06\bullet$	$72.60 \pm 3.24\bullet$	$70.30 \pm 4.00\bullet$
Heart-Statlog	82.96 ± 8.04	81.85 ± 5.08	79.63 ± 6.82	80.74 ± 8.52	80.37 ± 7.82	78.52 ± 7.57
Hepatitis	84.50 ± 6.22	89.58 ± 5.61	90.92 ± 5.54	$83.25 \pm 5.35\bullet$	$83.17 \pm 7.07\bullet$	$82.71 \pm 8.27\bullet$
Ionosphere	88.60 ± 5.88	91.44 ± 3.82	88.92 ± 5.80	91.15 ± 4.37	90.87 ± 5.69	91.15 ± 6.25
Molec. Biol. Prom.	78.09 ± 14.57	83.73 ± 9.28	83.64 ± 11.36	88.00 ± 13.00	85.73 ± 10.96	88.64 ± 5.94
Mushroom	$100 \pm 0\circ$	99.95 ± 0.06	99.95 ± 0.06	96.31 ± 5.94	$100 \pm 0\circ$	$100 \pm 0\circ$
Tic-Tac-Toe	$91.76 \pm 3.81\circ$	76.20 ± 1.47	72.86 ± 4.59	$96.87 \pm 1.55\circ$	$98.02 \pm 1.59\circ$	$96.97 \pm 2.62\circ$
Trains	60.00 ± 51.64	50.00 ± 52.70	50.00 ± 52.70	40.00 ± 51.64	80.00 ± 42.16	50.00 ± 52.70
Voting-Record	$95.85 \pm 2.83\bullet$	98.39 ± 1.11	98.62 ± 1.19	$95.62 \pm 2.29\bullet$	$94.94 \pm 3.04\bullet$	$96.08 \pm 3.09\bullet$

The table shows that *Ensemble-Trees* perform mostly well w.r.t. classification. In several cases, *Ensemble-Trees* are significantly better than ensemble methods ($\bullet$ denotes a significant loss of a technique at the 5%-level using paired t-test), while being outperformed only on Mushroom (barely), and Tic-Tac-Toe ($\circ$ denotes a significant win at the 5%-level).

Inspection of the standard deviations, however, gives a first indication that *Ensemble-Trees* do not turn out to be more stable performance-wise when the data composition changes. In fact, *Ensemble-Trees* using only three rules per test-node ensemble show *less* standard deviation, i.e. less variance, w.r.t. accuracy than *Ensemble-Trees* with larger ensembles.

4.2 Size and Stability of Induced Trees

The second question we evaluated was concerned with stability of induced trees w.r.t. changes in the data, and the comprehensibility of *Ensemble-Trees* and ensembles of trees, respectively. We used the 10-fold cross-validation mechanism to simulate changes in the underlying data, and report on size characteristics of the trees in Tables 2 and 3. For C4.5 decision trees and *Ensemble-Trees*, we report the mean and standard deviation of sizes (number of nodes) and maximal depths, i. e. length of the longest branch, of the different trees. Since ensemble methods induce a different tree in each iteration, averaging over iterations *and* the folds becomes a difficult endeavor, and we report on the accumulated number of nodes (of all trees per fold).

Table 2. Number of nodes per tree for C4.5/*Ensemble-Trees*, accumulated number of nodes for all trees of the respective ensembles

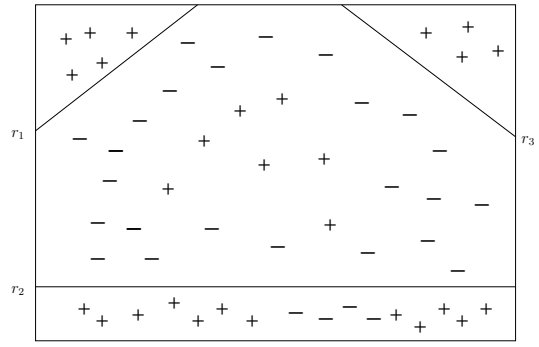
Data set	C4.5	$ET_{k=3}$	$ET_{k=5}$	Bagging	AdaBoost$_{RS}$	AdaBoost$_{RW}$
Breast-Cancer	13.6 ± 6.4	7.6 ± 3.5	9.4 ± 5.1	425.4 ± 19.4	485.2 ± 18.2	509.4 ± 14.7
Breast-Wisconsin	14.8 ± 1.5	13.4 ± 2.8	9.0 ± 2.1	154.2 ± 12.2	334.4 ± 12.2	331.8 ± 15.4
Credit-A	19.2 ± 4.2	16.4 ± 6.2	10.6 ± 6.1	226.6 ± 8.5	698.6 ± 17.0	725.2 ± 30.4
Credit-G	86.4 ± 9.2	22.8 ± 6.1	18.8 ± 8.9	896.0 ± 23.7	1212.2 ± 25.7	1239.2 ± 27.9
Heart-Statlog	14.4 ± 2.3	11.2 ± 2.2	10.8 ± 3.9	193.2 ± 14.5	302.6 ± 14.3	310.0 ± 14.5
Hepatitis	6.2 ± 3.0	10.6 ± 4.1	9.2 ± 3.0	86.6 ± 10.4	153.6 ± 7.9	165.4 ± 7.9
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0	165.6 ± 7.6	243.4 ± 8.9	253.0 ± 9.2
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3	98.2 ± 8.3	83.4 ± 3.9	85.8 ± 6.0
Mushroom	17	17	21	153.8 ± 5.6	17	169.4 ± 15.0
Tic-Tac-Toe	53.4 ± 2.8	3	2.8 ± 0.6	573.4 ± 9.2	700.8 ± 31.4	736.6 ± 23.7
Trains	3	3	3	30.2 ± 0.6	23.5 ± 7.5	28.4 ± 0.8
Voting-Record	8.2 ± 1.0	13.0 ± 3.4	13.2 ± 4.0	75.6 ± 6.7	184.2 ± 18.0	181.6 ± 15.7

Inspection of Table 2 shows that *Ensemble-Trees* always have a lot less nodes than ensembles of trees. Even if the accumulated sizes are normalized by dividing by the number of trees, this difference is still pronounced in most cases. Given that the ensemble methods do not improve markedly on the accuracy of *Ensemble-Trees* in most cases, quite a few more bags/iterations would be needed for better performance (except for the Mushroom data set), in turn leading to even less comprehensible final classifiers.

Ensemble-Trees also typically have somewhat fewer nodes than classical C4.5 decision trees, due to the more expressive tests in the nodes. The cases where this trend does not hold (the Hepatitis, and Voting-Record data sets) or is exaggerated (the Breast-Cancer, and Credit-G data sets) are also the ones where *Ensemble-Trees* perform best compared to the other approaches. However, in the case of the Tic-Tac-Toe data set, the small number of nodes is actually a symptom of an underlying characteristic, with another being the atrocious performance of *Ensemble-Trees*, compared to the other methods.

To explain the mechanism at work here, Figure 1 shows binary class data points in two-dimensional space, and the decision surfaces of three rules r_1, r_2, r_3. As can be seen, each of these rules predicts the positive class, advocating that

Fig. 1. Binary class data and decision surfaces of three discriminatory rules



the instances they cover be sorted to the left. Additionally, however, all of them advocate the sorting of the instances covered by the other two rules to the *right*. The result of the majority vote on this is that all instances are sorted to the right, the left subset is empty and thus its size less than m, leading to the formation of a leaf. The rather aggressive pre-pruning effected by this stopping criterion becomes problematic on data sets with small, non-overlapping subregions in the data. Tic-Tac-Toe is such a data set, as indicated by the rather large number of nodes (and therefore leaves) in – pruned – C4.5 decision trees. The expected stabilization of trees w.r.t. changes in the data, however, cannot

Table 3. Averaged maximal depths for C4.5 trees/*Ensemble-Trees*

Data set	C4.5	$ET_{k=3}$	$ET_{k=5}$
Breast-Cancer	4.1 ± 1.5	2.90 ± 1.2	3.5 ± 2.2
Breast-Wisconsin	4.4 ± 0.5	4.2 ± 0.9	3.9 ± 0.9
Credit-A	6.7 ± 1.3	5.8 ± 2.7	4.0 ± 2.7
Credit-G	23.9 ± 2.3	7.9 ± 2.1	18.8 ± 8.9
Heart-Statlog	3.2 ± 0.8	3.8 ± 0.8	3.6 ± 1.0
Hepatitis	1.1 ± 1.5	4.7 ± 2.0	4.0 ± 1.3
Ionosphere	6.9 ± 1.6	5.6 ± 3.0	4.6 ± 1.8
Molec. Biol. Prom.	2.8 ± 1.0	2.2 ± 0.4	2.7 ± 0.7
Mushroom	4	5	5
Tic-Tac-Toe	6.0	1	0.9 ± 0.3
Trains	1	1	1
Voting-Record	2.6 ± 0.5	4.2 ± 0.8	4.4 ± 1.1

be observed. Neither in the number of nodes, nor in the maximal depths of trees do *Ensemble-Trees* markedly decrease the standard deviation over folds, compared to C4.5 pruned trees. Quite contrary, while the trees are shallower, and mostly smaller, *Ensemble-Trees* often show greater variance in both characteristics than classical decision trees do. Since the size measurements were extracted

Table 4. Number of nodes per tree for unpruned C4.5/*Ensemble-Trees*

Data set	C4.5	$ET_{k=3}$	$ET_{k=5}$
Breast-Cancer	41.0 ± 8.7	7.6 ± 3.5	9.4 ± 5.1
Breast-Wisconsin	19.4 ± 4.9	13.4 ± 2.8	9.0 ± 2.1
Credit-A	35.8 ± 11.2	16.4 ± 6.2	10.6 ± 6.1
Credit-G	147.0 ± 10.1	22.8 ± 6.1	18.8 ± 8.9
Heart-Statlog	26.8 ± 3.3	11.2 ± 2.2	10.8 ± 3.9
Hepatitis	24.8 ± 2.6	10.6 ± 4.1	9.2 ± 3.0
Ionosphere	17.2 ± 3.7	12.2 ± 6.0	10.4 ± 4.0
Molec. Biol. Prom.	11.6 ± 1.0	5.4 ± 0.8	6.4 ± 1.3
Mushroom	21	17	21
Tic-Tac-Toe	68.2 ± 7.3	3	2.8 ± 0.6
Trains	3	3	3
Voting-Record	8.8 ± 1.1	13.0 ± 3.4	13.2 ± 4.0

after post-pruning, we finally compare unpruned C4.5 trees to *Ensemble-Trees* in an attempt to understand how much of a stabilization effect post-pruning provides to the decision trees. The number of nodes is shown in Table 4, with the characteristics of *Ensemble-Trees* duplicated from Tables 2. Due to the page limit, we do not show the effects of pruning on the depth of trees here.

While Table 4 suggests that the reduction in variance for the size of a decision tree is a result of the post-pruning operation, and similar effects exist regarding maximal depth of the tree, this still means that *Ensemble-Trees* are structurally not more stable w.r.t. changes in the data than classical decision trees. While the use of ensembles of conjunctive rules as tests could bring more stability to the final tree, this promise remains unfulfilled. A potential improvement could lie in a better solution for the combination strategy, an issue we will investigate in future work.

A final subject that needs to be addressed is the interpretability of found solutions. Classical decision trees and ensemble methods are the borders of an interval that ranges from

- Easy (decision trees): paths are interpreted as conjunctions, tests as disjunctions, via
- Challenging (BAGGING): individual trees are supposed to describe broadly the same phenomena, with the final classifier being an average of all, to
- Hard (BOOSTING): individual trees model local phenomena, with the final classifier being a weighted combination,

with *Ensemble-Trees* residing somewhere on the easier side of BAGGING. As stated above, divide-and-conquer (the decision tree mechanism) is a way of changing class distributions. While decision trees are usually claimed to be easily interpretable, especially when written down in rule-form, this is deceiving in that each test is meaningful on a different distribution than the ones preceding it. So removing the tree structure is a blessing in disguise since a relevant part of the full model is not seen by the user anymore. Once the tree structure is kept, however, with conjunctive rules in the nodes themselves having a clear semantic, an *Ensemble-Tree* is rather straight-forward to interpret, especially when small ensembles are used.

5 Conclusion

In this work we develop the classification technique of *Ensemble-Trees*, an attempt to leverage the power of ensemble techniques in dealing with over-fitting and variance effects, while retaining as much of the easy interpretability of decision trees as possible.

Our experimental evaluation showed that over-fitting effects seem to be prevented more efficiently, leading to better classification accuracy in several cases. The structural stabilization of individual trees that seems possible if more complex tests are used that balance each other's bias, does not materialize however.

While the advantages of *Ensemble-Trees* are clear – no need for post-pruning, shallower trees with fewer nodes, and far easier interpretability than existing ensemble methods using decision trees as weak classifiers, they come with a caveat. There exist data sets where the aggressive pre-pruning that results from the use of ensembles of rules is detrimental to the performance of the classifier. Potential remedies, such as changing the combination strategy, or dynamically adjusting the k parameter, will be explored in future work.

References

1. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman & Hall, New York (1984)
2. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
3. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees. Artif. Intell. **101** (1998) 285–297
4. Murthy, S.K.: On Growing Better Decision Trees from Data. PhD thesis, John Hopkins University, Baltimore, Maryland, USA (1997)
5. Kohavi, R., Kunz, C.: Option decision trees with majority votes. In Fisher, D.H., ed.: ICML, Morgan Kaufmann (1997) 161–169
6. Breiman, L.: Random forests. Machine Learning **45** (2001) 5–32
7. Zaki, M.J., Aggarwal, C.C.: XRules: an effective structural classifier for XML data. In Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C., eds.: KDD, Washington, DC, USA, ACM (2003) 316–325
8. Lindgren, T., Boström, H.: Resolving rule conflicts with double induction. In: 5th International Symposium on IDA, Berlin, Germany, Springer (2003) 60–67
9. Morishita, S., Sese, J.: Traversing itemset lattices with statistical metric pruning. In: PODS, Dallas, Texas, USA, ACM (2000) 226–236
10. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, H., Washio, T.: Performance evaluation of decision tree graph-based induction. In Grieser, G., Tanaka, Y., Yamamoto, A., eds.: Discovery Science, Sapporo, Japan, Springer (2003) 128–140
11. Bringmann, B., Zimmermann, A.: Tree^2 - decision trees for tree structured data. In Jorge, A., Torgo, L., Brazdil, P., Camacho, R., Gama, J., eds.: PKDD, Springer (2005) 46–58
12. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55** (1997) 119–139
14. Galiano, F.B., Cubero, J.C., Sánchez, D., Serrano, J.M.: Art: A hybrid classification model. Machine Learning **54** (2004) 67–92
15. Frank, E., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (1999)