Seeing the Forest through the Trees: Learning a Comprehensible Model from an Ensemble

Anneleen Van Assche and Hendrik Blockeel

Computer Science Department, Katholieke Universiteit Leuven, Belgium
{anneleen.vanassche,hendrik.blockeel}@cs.kuleuven.be

Abstract. Ensemble methods are popular learning methods that usually increase the predictive accuracy of a classifier though at the cost of interpretability and insight in the decision process. In this paper we aim to overcome this issue of comprehensibility by learning a single decision tree that approximates an ensemble of decision trees. The new model is obtained by exploiting the class distributions predicted by the ensemble. These are employed to compute heuristics for deciding which tests are to be used in the new tree. As such we acquire a model that is able to give insight in the decision process, while being more accurate than the single model directly learned on the data. The proposed method is experimentally evaluated on a large number of UCI data sets, and compared to an existing approach that makes use of artificially generated data.
Key words: ensembles, decision trees, comprehensibility

1 Introduction

In the process of knowledge discovery, one seeks to extract useful knowledge from data bases. But for knowledge to be useful, predictive accuracy is not sufficient: the extracted models also need to be understood by human users in order to trust them and accept them. Moreover users often construct models to gain insight in the problem domain rather than to obtain an accurate classifier only. For this reason, researchers have advocated machine learning methods which yield comprehensible models, such as decision tree learners and rule learners. Another issue in knowledge discovery is the stability of the classifier. A classifier may be very accurate and also comprehensible, but as long as the model is very instable, meaning that small changes in the data might have a large influence on the predictions of the model, users may have a hard time accepting it.

For quite some years now, a lot of interest has been shown to a class of learning methods called ensembles. The main goal in the design of ensemble methods is to increase the predictive accuracy of the classifier. And indeed, studies have shown the discrimination power of ensemble methods both theoretically and empirically [1, 2, 6], and in propositional as well as relational learning [12, 11, 5, 17]. Ensemble methods are learning algorithms that first construct a set of (diverse) classification models and then classify new data points by combining the predictions of each of these models. Exactly by doing so, they are often able to increase the stability and predictive accuracy significantly over the single models. On the other hand the comprehensibility of the learned hypothesis drops significantly, since the result of an ensemble method is a large set of models with (weighted) votes connected to each of them, which obviously becomes harder to interpret.

Some authors have pointed out that striving for comprehensibility is one of the important issues in ensemble learning requiring future investigations [1, 14]. Quite some successful research has been carried out already in this area. More in particular researchers have tried to obtain comprehensibility by means of extracting a new interpretable model from an existing (ensemble) model without sacrificing too much accuracy. Craven and Shavlik [4] presented an algorithm Trepan for extracting comprehensible, symbolic representations from trained neural networks. Trepan extracts a decision tree using the network as an oracle to answer queries during the extraction process. Domingos [7] proposed Combined Multiple Models (Cmm). Cmm first builds an ensemble of multiple models and then reapplies the base learner to recover the partitioning implicit in the multiple model ensemble. This is achieved by giving the base learner a new training set, composed of a large number of examples generated and classified according to the ensemble. Zhou et al. [19] utilize neural network ensembles to generate new instances and then extract symbolic rules from those instances. Ferri et al. [8] describe a method to learn a comprehensible model from an ensemble by selecting the single hypothesis from a multi-tree that is most similar to the combined hypothesis according to a similarity measure.

The approaches described above all rely on artificially generated data to tackle the problem of finding an interpretable model that approximates the ensemble: either by classifying this new data by the ensemble and constructing a new interpretable model on it, or by using this new data to measure similarity between the ensemble model and candidate interpretable models. However in some domains, generating artificial data is not straightforward: for instance when dealing with relational data, data distributions are far more complex and examples no longer have a fixed set of attributes.

For this reason, we propose a method to learn a single interpretable model from an ensemble without the need of generating artificial data. Instead of first generating artificial data and computing class distributions for different possible tests on this data, class distributions are estimated directly from the information available from the different decision trees in the ensemble in order to decide which tests are to be used in the new tree. In this paper we investigate how well this method performs in a propositional context, though most of its benefit might lie in a relational context, where other methods, based on artificial data, become hard to apply. We describe the proposed approach in detail in the next section. Afterwards it is evaluated on a large number of benchmark data sets comparing accuracy, complexity and stability to the original single tree and the ensemble classifier as well as to the single tree obtained by using the Cmm method [7] that makes use of artificially generated data. In the last section we formulate conclusions and some ideas for future research.

2 Algorithm

In the remainder of the paper we will propose a method to learn a single decision tree from an ensemble that is constructed via bagging but the method applies to other ensemble learners such as boosting [9] or random forests [3] as well.

2.1 Motivation

Dietterich [6] describes three fundamental reasons why an ensemble may work better than a single classifier. The first reason is statistical: without sufficient data, a learner may find several different hypotheses that correctly classify all training data. Constructing an ensemble consisting of these different hypotheses and then averaging over them may reduce the risk of choosing a wrong hypothesis. The second reason is computational. By performing some form of local search, learning algorithms may end up in different local optima and an ensemble consisting of these classifiers may give a better approximation to the true unknown function than the individual classifiers. The last reason is representational. Often the true function cannot be represented by any of the hypotheses in the hypotheses space of the learner. By applying weighted voting among these hypotheses the hypotheses space may be expanded. However, in the particular case of ensembles of decision trees, the reason of representability does not hold: actually a single decision tree in itself is able to represent any function described over the instance space as it can separate the instance space completely if necessary. The improvement obtained by ensembles of decision trees thus follows from the statistical and computational reasons. We aim to retain at least part of this improvement by constructing a tree that approximates the ensemble.

2.2 Constructing a Single Tree Exactly Representing an Ensemble

Assume E is an ensemble of N decision trees, which we would like to represent by one single decision tree. The ensemble E gives predictions to new examples x according to a combination of the predictions of each of its base trees, in this case the average:

$$L_E(x) = argmax_{C_i}(\frac{1}{N}\sum_{k}^{N} P_k(C_i|x)) \tag{1}$$

As a decision tree is able to represent any function described over the instance space, there also exists a decision tree that exactly represents the function described by the ensemble. Depending on the order of the tests in the nodes, the tree that represents the same concept as the ensemble, can consist of up to 2^d leaves, with d the number of different tests in the ensemble. So although it represents the same concept as the ensemble, such a tree would not satisfy our needs, namely be interpretable and give insight in the ensemble's decisions, simply because it is too big. Very often even the smallest tree exactly representing the concept in the ensemble might be far too large to be considered interpretable and an approximation to the ensemble will be preferred over the exact representation of the ensemble.

2.3 Computing Heuristics from the Ensemble

In order to give insight into the ensemble, the tests placed in the nodes of the new tree should be the '*most informative*' nodes at that point according to the ensemble. Usually tests with a higher information gain or gain ratio are considered more informative than tests with a lower value for these heuristics. So we need a way of computing these heuristics from the ensemble. The aim is to approximate the concept in the ensemble, therefore the class distributions predicted by the ensemble are used and not those directly available in the training data. Let's say we want to compute which test has the highest information gain according to the ensemble in a certain node n of the new tree. Suppose B is the conjunction of tests that occurred along the path from the root until node n, then the regular formula of information gain IG for a certain test T in n is

$$\begin{aligned} IG(T|B) = \mathit{entropy}(B) \quad & -P(T|B)\mathit{entropy}(T \wedge B) \\ & -P(\neg T|B)\mathit{entropy}(\neg T \wedge B) \end{aligned} \tag{2}$$

where

$$\mathit{entropy}(A) = \sum_{i=1}^{c} -P(C_i|A)\log_2 P(C_i|A) \tag{3}$$

with c the total number of classes, C_i the ith class and A any set of conditions.

A decision tree is constructed to model class distributions in the data and thus can be used to estimate these $P(C_i|A)$. Suppose we have a decision tree DT_k, now for any A we can estimate $P(C_i|A)$ by propagating it through the tree, applying the law of total probability in each node, until we end up in the leaves. Then we get:

$$P_k(C_i|A) = \sum_{\text{leaves } l_{kj} \text{ in } DT_k} P(C_i|Y_{kj} \wedge A)P(Y_{kj}|A) \tag{4}$$

where Y_{kj} is the conjunction of tests from the root of tree DT_k until leaf l_{kj} and $P_k()$ is the estimate of $P()$ by tree DT_k. The class probability estimate $P_E(C_i|A)$ of the ensemble E is then the average over the class probability estimates $P_k(C_i|A)$ of the N trees in E.

Now the probability $P(C_i|Y_{kj} \wedge A)$ corresponds to the probability estimate $P_k(C_i|Y_{kj})$ given by leaf l_{kj} (by using example frequencies in that leaf). Indeed, either $A \subseteq Y_{kj}$ and $P(C_i|Y_{kj}) = P(C_i|Y_{kj} \wedge A)$ or , $A \nsubseteq Y_{kj}$ and then we can assume that the class C_i is conditionally independent from the tests in A given the tests in Y_{kj}, because if not, the leaf l_{kj} would have been split at least once more on a test $T \in A \setminus Y_{kj}$.

For the other probability $P(Y_{kj}|A)$, we can distinguish 3 possible cases:

- $Y_{kj} \subseteq A$: then $P(Y_{kj}|A) = 1$ and for other leaves $Y_{kl} : P(Y_{kl}|A) = 0$ hence $P_k(C_i|A) = P_k(C_i|Y_{kj})$
- $\exists T : T \in Y_{kj}, \neg T \in A$: then $P(Y_{kj}|A) = 0$ and leaf l_{kj} of tree DT_k will not contribute in the probability $P_k(C_i|A)$

– else: $0 < P(Y_{kj}|A) < 1$ and leaf l_{kj} of tree DT_k partly contributes to the probability $P_k(C_i|A)$

To be able to estimate these probabilities $P(Y_{kj}|A)$ from the trees, we need to make the assumption that Y_{kj} is conditionally independent from $A \setminus \{Te \in A|Te \in Y_{kj} \vee \neg Te \in Y_{kj}\}$. Then $P(Y_{kj}|A) = P(Y_{kj}|\{Te \in A|Te \in Y_{kj} \vee \neg Te \in Y_{kj}\})$, which can be estimated from the tree by looking at the proportion of examples in the tree fulfilling these constraints. This assumption is exactly the same as made by Quinlan [13], when classifying instances with missing values by a tree. But since decision trees are not specifically constructed to model distributions among the tests, another, maybe better way to estimate the probabilities $P(Y_{kj}|A)$ is by computing them on the training set (if the test set is already available, using both training and test set might provide an even better estimate). The same holds for the $P(T|B)$, as requested in equation 2.

Using the method described above to compute the information gain for tests according to an ensemble E, a decision tree is built representing the ensemble E, each time replacing a leaf n in the tree by an internal node as long as we can find a test T where $IG_E(T|n) \geq IG_E(T_i|n)$ and $IG_E(T|n) > 0$ for all possible tests T_i. On the other hand, if $IG_E(T_i|n) = 0$ for all tests T_i, all examples ending up in n will be labeled the same by the ensemble, and no further splitting is required to represent the ensemble.

2.4 Stop Criteria

The tree obtained using the method described above (where all probabilities are estimated from the ensemble), represents the ensemble but is often very large and also constructing it will be very time consuming. For that reason, it will be necessary to impose some stop criteria to avoid the tree from becoming too big. First we describe a way to do safe prepruning in order to avoid splits to be added that will not have an influence on the predicted class. This will not change the eventual predictions of the tree. Next, we will impose a none equivalence preserving stop criterion, to make the tree even more interpretable.

Safe prepruning At the end of the tree construction, some redundant splits will still be present in the tree, as they might change the class distributions in the leaves but not the eventual classes predicted by the leaves. Usually in decision tree algorithms, these nodes are removed by postpruning. The same can be applied here, but since the tree deduced from an ensemble usually becomes rather large, quite some time might be spent in adding all these redundant splits and it would be desirable to preprune the tree if possible.

We will check whether indeed all possible examples that end up in a node n of the new tree will be predicted the same class by the ensemble as follows. Examples that end up in n are examples that succeed for D_n, where D_n is the conjunction of tests along the path from the root until node n of the new tree. For each tree DT_k of the ensemble we keep track of the possible leaves l_{kj} in that tree where these examples fulfilling D_n might end up. Each of these leaves l_{kj}

gives a prediction of the probability of a class $P(C_i|Y_{kj})$, with Y_{kj} the tests along the path from the root of tree k to leaf l_{kj}. Then we can define a lower bound on the class probability prediction of the ensemble E for examples fulfilling D_n as follows:

$$P_{Emin}(C_i|D_n) = \frac{1}{N} \sum_k \min_{l_{kj}^{D_n}} P(C_i|Y_{kj})$$

and equivalently an upper bound:

$$P_{Emax}(C_i|D_n) = \frac{1}{N} \sum_k \max_{l_{kj}^{D_n}} P(C_i|Y_{kj})$$

where $l_{kj}^{D_n}$ are all the leaves in DT_k where examples satisfying D_n can possibly end up. Then if

$$\exists C \in \mathcal{C}, \forall C_i \in \mathcal{C} \setminus \{C\} : P_{Emin}(C|D_n) > P_{Emax}(C_i|D_n)$$

(where $\mathcal{C}$ contains all possible class values), all examples in n will be predicted as C by the ensemble, and the tree should not be split any further in that node.

None equivalence preserving stop criterion As mentioned before, the tree constructed using safe prepruning might still be very large. Because of this, we will introduce another stop criterion, such that a more comprehensible approximation to the ensemble is obtained. If all training examples ending up in a node are classified the same by the ensemble, no further splitting will be performed and the node will become a leaf. As a consequence, the training examples will be labeled the same by the new tree as by the ensemble, but for other new examples this is not necessarily the case.

3 Implementation

The method described above was implemented in the Weka data mining system [18]. Bagging with J48 (Weka's C4.5) as base learner was used as the ensemble to start from. Theoretically it is sufficient to use the tests that were used in the ensemble in order to represent the hypothesis of the ensemble by a single tree. And also intuitively, it makes sense only to use these tests to construct a new tree as these were probably the most important amongst all possible tests. Moreover in practice, tests not appearing in the ensemble will always have a lower information gain than the tests appearing in the ensemble and would never be chosen. As detailed in Section 2.3, to find the best test in a certain node of the new tree, each of the possible candidate tests are assigned a heuristic value by propagating them through the ensemble trees, and using the class probability estimates of the leaves they end up in. To estimate the probability that examples satisfying a set of tests A end up in a certain leaf Y_{kj} (so $P(Y_{kj}|A)$) or to find the probability of a certain test T given the tests B appearing already in the new tree along the path from the root until the current node, we can either use the ensemble or the data as indicated before. We distinguish the following three cases in the implementation:

1. both $P(Y_{kj}|A)$ from equation 4 and $P(T|B)$ from equation 2 are estimated from the trees in the ensemble. Then the assumption is made that Y_{kj} is independent from $A \setminus \{Te \in A | Te \in Y_{kj} \vee \neg Te \in Y_{kj}\}$, the same for $P(T|B)$. This means we assume tests are independent from each other unless the tree indicates they're not. [**Ism_t**]
2. $P(Y_{kj}|A)$ from equation 4 is estimated from the ensemble assuming independencies as above, but $P(T|B)$ is estimated from the available data. [**Ism_td**]
3. both $P(Y_{kj}|A)$ and $P(T|B)$ are estimated from the available data, and only the conditional class probabilities are estimated from the ensemble. [**Ism_d**]

Without using the stop criterion based on the data, applying ISM_t will result in a (often very large) tree that exactly represents the decisions of the ensemble. This is no longer necessarily the case when applying ISM_td or ISM_d as probabilities on the data are used and they might be 0 if no data is available for certain conditions. As 'available data' we usually have the training data to estimate certain probabilities. Sometimes the test set is also available on beforehand, or in a transductive learning setting, unlabeled examples might be at our disposal. When data is used to compute probabilities, the predictions of the new tree will be equivalent to those of the ensemble at least for all provided (unlabeled) instances. For other unseen examples there is no guarantee as the (shorter) tree is now only an approximation to the ensemble.

We also implemented CMM as described by Domingos [7]. But while the original version describes it for rule sets, for comparison with our method, we applied the general framework to decision trees. N artificial examples are generated as follows: suppose we have m trees obtained via bagging, then CMM generates N/m examples from each tree. For each leaf in the tree, if it classified r of the s examples in the bootstrap sample it was induced from, $(r/s)(N/m)$ examples covered by it will be generated. For each example this is done by ensuring that it satisfies all the tests from the root of the tree until that leaf, and beyond that by setting the values of its attributes according to a uniform distribution. The label of the new example is the prediction given by the ensemble. After having constructed a new training set, J48 is again applied to learn a new tree.

4 Empirical Evaluation

We will investigate empirically whether the proposed method can indeed retain some of the accuracy and stability gains of bagging while providing a more interpretable model. Experiments were carried out using a representative sample of 34 data sets from the UCI repository [10]. We omitted a few data sets where bagging performed worse than a single model, as these are cases where it actually makes no sense to apply the method described above. We used 25 iterations of bagging. This was also described by Domingos [7] to be a sufficiently large number: 50 models did not perform significantly better, while 10 models led to poor results. The number of artificial examples generated by CMM to learn

Table 1. Win-draw-loss table using a 90% two-tailed significance test comparing the different algorithms on 34 benchmark data sets from the UCI repository.

(w/d/l)	Bagging	Ism_t	Ism_td	Ism_d	Cmm(up)	Cmm(p)	Ism_tu	Ism_tdu	Ism_du
J48	(19/15/0)	(9/25/0)	(9/25/0)	(6/23/5)	(5/25/4)	(5/29/0)	(17/17/0)	(19/15/0)	(18/16/0)
Bagging	/	(0/13/21)	(0/13/21)	(0/12/22)	(0/12/22)	(0/9/25)	(0/31/3)	(0/31/3)	(0/31/3)
Ism_t	/	/	(5/26/3)	(1/22/11)	(1/22/11)	(2/23/9)	(19/15/0)	(19/15/0)	(19/14/1)
Ism_td	/	/	/	(1/19/14)	(0/24/10)	(1/25/8)	(18/16/0)	(18/16/0)	(18/16/0)
Ism_d	/	/	/	/	(4/23/6)	(8/23/3)	(19/15/0)	(20/14/0)	(19/15/0)
Cmm(up)	/	/	/	/	/	(8/25/1)	(21/13/0)	(21/13/0)	(21/13/0)
Cmm(p)	/	/	/	/	/	/	(24/10/0)	(24/10/0)	(24/10/0)
Ism_tu	/	/	/	/	/	/	/	(2/32/0)	(2/32/0)
Ism_tdu	/	/	/	/	/	/	/	/	(0/34/0)

a new tree was set to 1000, augmented with the training examples[1]. As Cmm applies J48 after generating new data, C4.5's pruning procedure could be used as well: we both build a pruned (Cmm(p)) and an unpruned tree (Cmm(up)) on the data. We compare the different versions of the Ism method (Ism_t, Ism_td, Ism_d as detailed in Section 3) all using the stop criteria described in Section 2.4. For deciding on stopping and (if necessary) estimating probabilities, these versions are only provided with the training data. Moreover we also evaluate the same algorithms in a transductive learning setting where also the unlabeled test instances are provided (Ism_tu, Ism_tdu, Ism_du). All results were obtained averaging over five 10-fold cross-validations. Binary decision trees were used in all methods.

4.1 Predictive accuracy

Table 1 describes the significant wins and losses on the 34 data sets comparing the different algorithms to each other, to J48, Bagging and Cmm.

Comparing the different versions of Ism we find that Ism_d is clearly inferior to the other two versions that make more use of information available in the ensemble instead of the data. It is significantly outperformed on 11 data sets by Ism_t and on 14 by Ism_td while it outperforms those only on one data set. Moreover Ism_d was also significantly outperformed by J48 on some data sets, while this is not the case for the other versions of Ism. For space limitations and as it is clearly inferior we will not consider Ism_d in further results. Table 2 shows more detailed accuracy results for all data sets. As can be seen, the other versions were able to do at least as good as J48, but in almost all cases accuracies were higher (9 significantly).

Surprisingly the unpruned version of Cmm did not improve the results of J48. This might support the claim of Domingos [7] that a good match is needed between the learner's bias and that of the probability estimation procedure and that estimating probabilities from the frequencies in the leaves of the trees in

[1] Note that the results for Cmm and bagging here do not easily compare to the results obtained by Domingos [7]. This is because both the base learner of bagging as the learner for Cmm differ from [7], moreover a different set of data sets was used.

Table 2. Test accuracy of the different algorithms on some benchmark data sets from the UCI repository.

Data Set	J48	Bagging	Ism_t	Ism_td	Cmm(up)	Cmm(p)	Ism_tu	Ism_tdu
anneal.ORIG	92.65	94.07 ○	93.52	93.94	94.63 ○	94.28 ○	93.76	94.12 ○
audiology	78.26	80.50 ○	81.47 ○	80.94 ○	80.86 ○	80.13	80.50 ○	80.50 ○
autos	76.21	79.59	79.88 ○	80.58 ○	78.49	77.32	79.10	79.59
balance-scale	77.59	81.30 ○	79.15 ○	79.05 ○	79.09 ○	79.98 ○	81.20 ○	81.20 ○
breast-cancer	69.95	71.67	70.49	70.20	67.48	69.65	71.67	71.67
breast-w	94.99	96.02 ○	95.19	95.16	95.22	95.22	96.02 ○	96.05 ○
car	97.16	97.35	97.50	97.42	97.15	97.05	97.35	97.35
cmc	52.96	52.74	52.23	51.62	49.69	52.34	52.73	52.74
colic	85.57	85.25	85.35	85.46	82.70 ●	85.52	85.41	85.41
credit-a	83.91	85.10	83.97	83.77	81.86 ●	83.91	85.13	85.10
credit-g	69.70	74.30 ○	72.38 ○	71.88 ○	67.38 ●	71.04	74.30 ○	74.30 ○
cylinder-bands	73.52	78.78 ○	75.81	76.93	72.48	72.85	78.89 ○	78.78 ○
dermatology	95.35	96.11	95.41	95.52	95.18	95.73	96.11	96.11
glass	70.97	73.65	68.02	69.44	68.70	67.36	73.65	73.65
heart-c	76.88	80.28 ○	79.55	79.15	79.02	79.09	80.41 ○	80.28 ○
ionosphere	90.59	92.70	91.92	92.04	91.51	91.16	92.70	92.70
iris	94.67	95.33	95.33	95.20	95.07	95.47	95.33	95.33
kropt	77.95	80.88 ○	80.94 ○	80.75 ○	77.39 ●	78.16 ○	80.81 ○	80.82 ○
letter	88.31	93.50 ○	89.67 ○	89.69 ○	88.23	88.16	93.50 ○	93.50 ○
liver-disorders	65.16	70.50 ○	66.36	67.86	66.94	67.46	70.50 ○	70.50 ○
lymph	73.05	81.27 ○	77.20	76.64	76.62	75.32	81.27 ○	81.27 ○
mfeat-pixel	88.30	93.00 ○	89.17	89.55 ○	87.36	87.53	93.00 ○	93.00 ○
nursery	99.31	99.59 ○	99.56 ○	99.56 ○	99.43 ○	99.36 ○	99.56 ○	99.56 ○
optdigits	90.94	95.59 ○	90.40	90.83	90.99	90.88	95.59 ○	95.59 ○
page-blocks	96.89	97.33 ○	97.01	97.05	96.86	97.05	97.33 ○	97.33 ○
primary-tumor	43.07	45.96 ○	43.83	43.18	43.07	44.07	45.78	45.95 ○
segment	96.93	97.52 ○	96.61	97.01	96.85	96.87	97.52 ○	97.52 ○
sonar	74.05	80.05	70.32	71.63	70.00	71.55	80.05	80.05
soybean	91.81	92.39	91.25	91.42	91.16	91.63	91.22	91.48
tae	58.25	60.81	60.97	60.97	61.07	60.43	60.81	60.81
tic-tac-toe	94.26	97.60 ○	96.79 ○	96.89 ○	96.30 ○	96.43 ○	97.60 ○	97.60 ○
vehicle	73.87	75.32	74.47	74.82	72.67	72.60	75.32	75.32
yeast	55.46	61.07 ○	57.18 ○	56.35	55.72	56.52	61.07 ○	61.07 ○
zoo	91.09	91.67	91.89	91.89	91.47	91.87	91.67	91.67
average accuracy	80.58	83.2	81.49	81.6	80.54	81	83.14	83.17
(○/ / ●)		(19/15/0)	(9/25/0)	(9/25/0)	(5/25/4)	(5/29/0)	(17/17/0)	(19/15/0)

○, ● statistically significant improvement/degradation over J48

the ensemble is not the best way to go. Pruning of these Cmm trees on the other hand increased overall performance.

Although Ism also estimates class probabilities from the frequencies in the leaves, it is less disturbed by it. Ism seems to retain more of the accuracy gains of bagging than Cmm(p) did: it outperforms J48 in more cases, its average accuracy is slightly higher, and moreover it is in less cases significantly different from bagging than Cmm(p). Also comparing Ism and Cmm(p) directly to each other we find that while Ism_t and Ism_td win 9 and 8 times significantly of Cmm(p), Cmm(p) wins respectively only 2 and 1 time.

If we take a look at the results for the transductive learning setting (rightmost two columns in Table 2), where the unlabeled test set is also provided to Ism, we see that Ism is able to retain usually all of bagging's accuracy gain on the test set. This means that, for the provided examples, Ism can give exactly the decisions needed to label these examples as the bagging ensemble does. Note that these decisions might not necessarily be sufficient to classify unseen examples exactly as the ensemble does.

Table 3. Average relative model size compared to J48 and CMM(p) of the different models

	CMM(p)	CMM(up)	Bagging	ISM_t	ISM_td	ISM_tu	ISM_tdu
J48	1.82	3.08	25.73	3.28	2.4	3.36	2.5

4.2 Comprehensibility

To compare the comprehensibility of different models usually the complexity of the model is considered. As all the models we are dealing with consist of decision trees, we can here define the complexity of a model to be the number of nodes occurring in the tree[2]. Assuming a single tree learned with J48 provides a comprehensible model, we compare the complexity of the new models to that of J48. Table 3 reports the average relative model size of a model X to J48 ($nb\_nodes(X)/nb\_nodes(\mathrm{J48})$).

As could be expected, the model size of a bagged ensemble (of 25 iterations) is about 25 times larger than that of a single model. ISM_td is able to reduce this to 2.4 times. Provided we are dealing with binary trees, this means that in a tree built with ISM_td on average only 1.2 extra tests are occurring on a path from the root to a leaf compared to a tree built by J48. In general, models induced by ISM_t are slightly larger than those of ISM_td. Since the information gain is completely estimated from the ensembles, in ISM_t splits might occur where all training examples go in one branch. In ISM_td, tests that give rise to such splits will get an information gain of zero as the probability that examples end up in one of the branches is computed on the training set. Additionally, the more (labeled or unlabeled) data is provided, the larger the models induced by ISM can become. The complexity of CMM with pruning is still smaller than that of ISM_td. Overall we assume that, provided an end-user is able to interpret a tree induced by J48, he should be able to interpret a tree induced by ISM as well.

4.3 Stability

Semantic stability is often measured by estimating the probability that models generated by the same learner on different training sets will give the same prediction to a randomly selected instance. Following the method described by Turney [15], for each data set 1000 unlabeled examples were generated from a uniform distribution and predictions of models obtained on the different folds of the data in cross-validations were compared to each other. The exact formula of the stability S of a certain learner X is then given below:

$$S(X) = \frac{1}{1000}\sum_{i=0}^{1000}[\frac{2}{(n-1)n}\sum_{k=1}^{n}\sum_{l=0}^{k-1}\delta(X_k(i),X_l(i))] \tag{5}$$

[2] Note that in case of an ensemble this is still an overly optimistic measure for comprehensibility because to understand the decisions it is not sufficient to look at all tests used in the trees but also how the trees are combined.

Table 4. Average stability of the different models

J48	Bagging	Ism_t	Ism_td	Cmm(up)	Cmm(p)	Ism_tu	Ism_tdu
73.74%	80.73%	74.89%	74.51%	76.08%	76.63%	75.64%	75.26%

where n is the number of folds, $X_k(i)$ is the prediction for example i from the model learned by learner X on fold k and $\delta(L, M)$ is 1 if L equals M else 0.

Table 4 presents the average stability of the different algorithms over all data sets. As can be seen, Ism is able to increase the stability only with about 1% over J48. Providing Ism with more information about the complete data distribution (by offering extra unlabeled data[3]) results in a further small increase of the stability. Bagging on the other hand, led to an improvement of 7% and Cmm 2.8%. While Ism retains more of the accuracy gain of bagging than Cmm, Cmm seems to preserve more of the stability gain.

5 Conclusions and Future Work

In this paper we presented a method to learn a decision tree that approximates the decisions made by an ensemble of decision trees. The tree is constructed without the need of generating artificial data. Instead, heuristics are computed for each of the possible tests by predicting class distributions for these tests using the ensemble. We show that we indeed obtain a tree that is able to retain some of the accuracy (and less of the stability) gains of bagging, while providing a model that keeps most of the comprehensibility of a single model directly learned on the data. Moreover the method can easily benefit from a transductive learning setting, where unlabeled instances are already available in the training phase. The supplied examples will be predicted the same as the ensemble by the new tree. Our method is also clearly competitive with an existing approach Cmm, which makes use of artificially generated data. In general, our method outperforms Cmm with respect to accuracy, while Cmm has a higher stability. Since for Cmm postpruning was very beneficial, both in terms of accuracy and complexity, this should be considered as future work for our method as well. To ensure comprehensibility of the obtained model one might even want to impose user defined size constraints on the model. The effect on the accuracy of constraining the size of the models further should be investigated. We would also like to assess our method using other ensemble learners, such as random forests and boosting. Furthermore this method should be extensively evaluated in the relational case. First experiments, reported in Van Assche et al. [16], already show promising results. Last, we would also like to extend the method for regression. This means a heuristic based on variance should be estimated from the ensemble instead of entropy. As variance is an additive measure, the variance for a certain test can be estimated based on the variances in the ensemble leaves that are consistent with the test.

[3] This is the unlabeled test data from the train-test folds, not the unlabeled data generated to compute the stability, if so, the stability would be the same as bagging.

Acknowledgements

Anneleen Van Assche is supported by the Institute for the Promotion of Innovation by Science and Technology in Flanders (I.W.T.-Vlaanderen). Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen).

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105, 1999.
2. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
4. M. W. Craven. *Extracting Comprehensible Models from Trained Neural Networks*. PhD thesis, University of Wisconsin, Madison, 2003.
5. I. de Castro Dutra, D. Page, V. Costa, and J. Shavlik. An empirical evaluation of bagging in inductive logic programming. In *Proc. of the 12th International Conference on Inductive Logic Programming*, pages 48–65, 2002.
6. T. Dietterich. Ensemble methods in machine learning. In *Proc. of the 1th International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.
7. P. Domingos. Knowledge discovery via multiple models. *Intelligent Data Analysis*, 2:187–202, 1998.
8. C. Ferri, J. Hernández-Orallo, and M. Ramrez-Quintana. From Ensemble Methods to Comprehensible Models. In *Proc. of 5th International Conference on Discovery Science (DS 2002)*, pages 165–177, 2002.
9. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In L. Saitta, editor, *Proc. of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.
10. S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.
11. S. Hoche and S. Wrobel. Relational learning using constrained confidence-rated boosting. In C. Rouveirol and M. Sebag, editors, *Proc. of the 11th International Conference on Inductive Logic Programming*, pages 51–64. Springer-Verlag, 2001.
12. J. Quinlan. Boosting first-order learning. In *Algorithmic Learning Theory, 7th International Workshop (ALT '96)*, 1996.
13. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
14. G. Ridgeway, D. Madigan, and J. Richardson, T. adn O'Kane. Interpretable boosted naive bayes classification. In *Proc. of the 4th International Conference on Knowledge Discovery in Databases*, pages 101–104. AAAI Press, 1998.
15. P. Turney. Bias and quantification of stability. *Machine Learning*, 20:23–33, 1995.
16. A. Van Assche and H. Blockeel. Seeing the forest through the trees: Learning a comprehensible model from a first order ensemble. In *Proc. of the 17th International Conference on Inductive Logic Programming*, to appear, 2007.
17. A. Van Assche, C. Vens, H. Blockeel, and S. Džeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 64(1-3):149–182, 2006.
18. I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005. 2nd Edition.
19. Z. Zhou, Y. Jiang, and S. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Communications*, 16(1):3–15, 2003.