Re-designing Cost-Sensitive Decision Tree Learning [⋆]

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

DSIC, UPV, Camí de Vera s/n, 46022 Valencia, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

Abstract. In this work we present several innovations in the generation of comprehensible models, by the use of a newly devised structure for the learning of multiple decision trees. In particular, we are able to obtain a set of models (a hypothesis ensemble) from which we can construct a locally combined model or select the best one (the archetype of the ensemble). We obtain significant improvements on several senses of "cost". First, new splitting criteria and evaluation measures are devised to minimise the misclassification cost. Secondly, the use of a multiple decision tree structure can enhance the quality of models over classical greedy decision trees. Thirdly, different parts of each single model are shared in the structure proposed, so dramatically reducing resource requirements, which are usually very high in other combination methods. According to our experiments, the technique proposed obtains the best results (in terms of accuracy and other cost-sensitive measures) of the current machine learning techniques that are able to generate comprehensible models.

Keywords: Decision tree learning, machine learning, data mining, cost-sensitive learning, comprehensibility, ensemble methods.

1 Introduction

The widespread use of machine learning techniques in real-world and business applications, such as data and web mining, knowledge discovery, personalisation tools, is demanding new machine learning methods which should be more comprehensible, more scalable, able to cope with huge volumes of data, and, in general, more sensitive to the application context. New machine learning methods such as hybrid methods, ensemble methods and support vector machines aim at obtaining more precise models than other classical methods. However, in this pursuit, most of them have neglected one or more of the other requirements of current machine learning applications: comprehensibility, expressiveness and context sensitivity. On the other hand, decision tree learning, rule learning and ILP methods have always stuck to the premise of obtaining comprehensible methods. However, they have sacrificed accuracy or efficiency. Ensemble methods have been used to improve the quality of decision tree learning and ILP techniques [20, 21]. Although the improvement has been significant, the comprehensibility of the resulting models is lost and, moreover, the cost in resources needed to store and obtain the set of hypotheses (ensemble) is extremely high. Finally, many evaluation metrics, such as accuracy or mean squared error are quite myopic in the sense that they do not take into account that not every error has the same cost. Depending on the application (especially in decision support systems), a misclassification or regression error can have more serious consequences than the reverse error.

In this paper we present a full re-design of decision tree learning that is based on a multi-tree structure, which can contain an exponential number of possible hypotheses with linear resources. From this *shared* ensemble a single hypothesis or a combination can be extracted. The way in which the multi-tree structure is created and in which solutions are sought allows a convenient handling of computational resources, so reducing the computational costs associated with other ensemble methods. The construction and selection of hypotheses are now based on cost-sensitive criteria, instead of classical accuracy (or error) measures, producing models that behave well within the widest range of contexts as possible (variable cost matrices). Finally, new methods for extracting

⋆ This work has been partially supported by CICYT under grant TIC2001-2705-C03-01, Generalitat Valenciana under grant GV00-092-14, and Acción Integrada Hispano-Italiana HI2000-0161.

comprehensible models are presented, in order to reduce the costs of model interpretation and dissemination.

The paper is organised as follows. First, in section 2, we discuss the kinds of costs that are hampering the potential benefits of machine learning and how these costs can be reduced. Section 3 describes a new decision tree structure, dubbed multitree, upon which the search and extraction methods are built. The first practical use of such a structure is described in section 4, where the exponential number of trees can be combined locally, using new fusion methods. Section 5 addresses the cost of interpretation of ensemble methods by selecting the single representative solution (archetype) that is most similar to the combined solution, so giving a comprehensible and highly accurate model. Misclassification costs are addressed in section 6, and new cost-sensitive measures are introduced, which replace accuracy or error-based evaluation. Section 7 presents several experiments concerning the previous features and section 8 discusses some unexpected applications of the archetype method for "capturing" comprehensible models from incomprehensible ones. Finally, the last section presents the conclusions and proposes some future work.

2 Costs Associated with Machine Learning

The potential benefits of machine learning are usually hampered by some costs associated with current techniques. Obviously, there are some costs which are previous or external to machine learning, such as data cleansing and data transformation. But many others are intrinsic to machine learning algorithms. In particular, we can classify these hindrances or costs into two categories.

- Generation Costs: these are mainly *computational costs*, i.e., computer resource consumption. Many machine learning algorithms are not scalable to large volumes of data, owing either to memory limitations or to time restraints (or both). Due to the intrinsic complexity of learning, the idea is to design machine learning methods that make an effective use of resources or, in other words, that give better solutions for increasing provided resources (anytime algorithm).
- Application Costs: even when the learning stage has been quick and easy, and the models are accurate *in average*, this does not mean that the models that have been obtained can be used seamlessly and confidently. First, a model can be highly accurate for frequent cases but extremely inaccurate for infrequent, critical situations. In many contexts, such as diagnosis or fault detection, the errors in these circumstances (e.g. not detecting a fault when there it is) usually have much higher costs than errors in the reverse common situation (e.g. detecting a fault when there it is not). Secondly, even accurate models which take into account that some errors are more serious than others can be useless if the purpose is to obtain some new knowledge from the data. The resulting model may be incomprehensible, either because it is not expressed in the form of rules or the number of rules is too high. In these circumstances, the interpretation of results is fraught with significant costs or it may even be impossible.

It must be said that special attention has been put on the first kind of costs in the machine learning community, especially with the ever-increasing sizes of datasets usual in data mining applications. However, measures to alleviate the first kind of costs have been sometimes accompanied by a worsening of the second kind of costs.

Moreover, the race between new machine learning algorithms has usually been set up in terms of accuracy, i.e., a more accurate learner *ought to be* a better learner. This kind of competition has neglected many of the costs we have just described, mainly computational costs (e.g. the high resource consumption of ensemble methods) and interpretation costs (e.g. the incomprehensibility of neural networks and support vector machines). However, there has been little success of new algorithms that find a good compromise in all the relevant aspects that influence the benefits of machine learning.

In what follows, we show the integration of different issues of some successful and modern techniques in machine learning, such as ensemble methods and ROC analysis, and its combination with some new proposals we have recently introduced (multi-trees, archetyping). This amalgamation of old and new techniques do allow the generation of comprehensible and cost-sensitive models that can be obtained with an appropriate use of computational resources.

3 Multi-tree Structure

The construction of decision trees is usually performed in two different steps [19]:

- **Tree Construction:** In this phase, the whole decision tree is constructed. The process is driven by a splitting criterion that selects the best split. The selected split is applied to generate new branches, and the rest of splits are discarded. The algorithm stops when the examples that fall into a branch belong to the same class.
- **Pruning**. This phase consists in the removal of parts of the tree in order to avoid overfitting.

Thus, decision trees are built in a eager way, which allows the quick construction of a model. However, it may produce bad models because of bad decisions. In [8] we have defined a new structure in which the rejected splits are not removed, but stored as *suspended* nodes. The further exploration of these nodes after the first solution has been built allows the extraction of new models from this structure. For this reason we call it decision multi-tree, rather than decision tree. Since each new model is obtained by extending the existing structure of the multi-tree, these models share their common parts. A decision multi-tree can also be seen as an AND/OR tree [15, 17], if one consider the split nodes as OR-nodes, and the nodes generated by an exploited OR-node as AND-nodes.

For the construction of a decision multi-tree it is necessary to specify two new criteria apart from the two required for the construction of a single decision tree.

- **Suspended node selection:** To populate a multi-tree, we need to specify a criterion that selects one of the suspended nodes. In [9] we presented and evaluated some criteria, such as 'topmost', 'bottom', or 'random'. Our experimental results showed that 'random' is a trade-off between speed and quality.
- **Selection of a model**: Since a multi-tree represents a set of models, we may want to select one or n comprehensible models (decision trees) according to a selection criterion (we usually use a criterion based on Occam's razor, i.e. the solution with fewer rules).

Figure 1 shows a decision multi-tree. OR-nodes are represented with an arc and leaves are represented by rectangles. Three different models are exhibited in this multi-tree since two suspended nodes have been exploited.

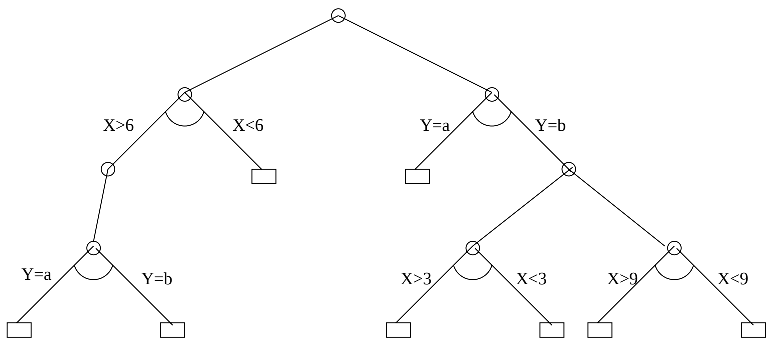

Fig. 1. Example of the multi-tree structure.

The decision multi-tree approach presents some interesting features. First, the number of solutions grows exponentially w.r.t. the number of suspended OR-nodes exploited. Secondly, the solutions share some of their parts. The quantity of the shared nodes depends on the depth of the suspended OR-node which has been exploited.

4 Shared Ensembles Combination

It has been shown that the combination of a set of classifiers is a technique that improves the accuracy of simple classifiers. Some combination methods are boosting, bagging, randomisation, stacking and windowing [20][3]. However, the amount of memory required to store the hypotheses can make ensemble methods hard to deploy in applications. One way to partially overcome this limitation could be to share the common parts of the components of the ensemble. In [5] we have presented a new ensemble method in which the set of hypotheses to be combined share their common parts by using the multi-tree structure we have described in section 3. Next, we summarise how the combination can be performed *locally* in this structure.

We consider a *class vector* $(v_{k,1}, \ldots, v_{k,C})$ for each node k, where C is the number of classes. For leaf nodes, $v_{k,j}$ are the training cases of class j that have fallen into the leaf k. For internal nodes we have to propagate upwards the class vectors. This is done for each new unlabelled example we want to make a prediction for. For the OR-nodes, the vector is the one of the child where the example falls (since an example can only fall through one of its children). For the AND-nodes, we do a fusion of the vectors of its children. Following [11], we have considered several fusion strategies that convert m class vectors into one combined vector Ω_j:

- **sum**: $\Omega_j = \sum_{k=1}^{m} v_{k,j}$
- **arithmetic mean**: $\Omega_j = \sum_{k=1}^{m} \frac{v_{k,j}}{m}$
- **product:** $\Omega_j = \prod_{k=1}^{m} v_{k,j}$
- **geometric mean:** $\Omega_j = \sqrt[m]{\prod_{k=1}^{m} v_{k,j}}$
- **maximum:** $\Omega_j = \mathtt{max}_k(v_{k,j})$
- **minimum:** $\Omega_j = \mathtt{min}_k(v_{k,j})$

In addition, we have devised some transformations to be done to the original vectors at the leaves before its propagation:

- **good loser** : $v'_{k,j}(x) = \sum_j v_{k,j}(x)$ if $j = \mathtt{argmax}(v_{k,j}(x))$ and 0 otherwise
- **bad loser:** $v'_{k,j}(x) = v_{k,j}(x)$ if $j = \mathtt{argmax}(v_{k,j}(x))$ and 0 otherwise.
- **majority:** $v'_{k,j}(x) = 1$ if $j = \mathtt{argmax}(v_{k,j}(x))$ and 0 otherwise.
- **difference**: $v'_{k,j}(x) = v_{k,j}(x) - \sum_{i \neq j} v_{k,j}(x)$

We will use the vector and fusion methods 'difference' and 'maximum', because, according to our experiments, this pair has given the best results.

This makes up a local way of combining hypotheses, which turns out to be a more precise and efficient way of combinining shared ensembles. Note that a global (topmost) combination would be unfeasible, due to the exponential number of solutions.

5 Archetyping Shared Ensembles

Although accuracy is significantly increased with ensemble methods, the main drawback is that comprehensibility of a single model is lost. In [8] we have addressed how to reduce from a combination of hypotheses (shared ensamble) to one single hypothesis without losing too much accuracy w.r.t. the combined hypothesis. This single hypothesis is called an *archetype* or *representative* of the ensemble. The idea is to select the single hypothesis that is most similar to the combined one according to a suitable measure of similarity. In this way, the selected model exhibits in general a higher accuracy than the model selected by using classical methods for selecting one hypothesis, such as that one with the smallest size or with the lowest expected error.

In order to do this, the hypotheses are compared with the combined hypothesis, which is used as an oracle. We have studied several measures of similarity which are based on metrics defined for a pair of hypothesis [12]. The metrics use a matrix that summarises which is the class predicted by a single hypothesis and the one predicted by the combined hypothesis. More formally, given two classifiers h_a and h_b, and an unlabelled dataset with n examples with C classes, we can construct a $C \times C$ contingency or confusion matrix $M_{i,j}$ that contains the number of examples e such that $h_a(e) = i$ and $h_b(e) = j$.

In many situations, it is possible that a single hypothesis is the most similar one to the combined hypothesis *with respect to the training set*, but it is not the most similar one in general (*with respect to other datasets*). Therefore, it is suitable or even necessary to evaluate similarity with respect to an external (and desirably large) reference dataset. In many cases, however, we cannot reserve part of the training set for this, or it could be counterproductive.

The idea is then to use all the training set for constructing the hypotheses and a *random* dataset for selecting one of them. For this reason, we have considered the following measures which extend the metrics in [12] for dealing with classification problems which involve more than two classes and for using unlabelled datasets for the estimation of similarities:

– θ **measure**: It is just based on the idea of reckoning the probability that both classifiers agree:

$$\theta = \sum_{i=1}^{C} \frac{M_{i,i}}{n}$$

Its value is between 0 and 1. An inverse measure, known as discrepancy, is also used by [12].

– κ **measure**: The previous metric has the problem that when one class is much more common than the others or there are only two classes, this measure is highly affected by the fact that some predictions may match just by chance. Following [14], we define the Kappa measure, originally introduced as the Kappa statistic (κ) [2]. This is just a proper normalisation based on the probability that two classifiers agree by chance:

$$\theta_2 = \sum_{i=1}^{C} (\sum_{j=1}^{C} \frac{M_{i,j}}{n} \cdot \sum_{j=1}^{C} \frac{M_{j,i}}{n})$$

As a result, the Kappa statistic is defined as:

$$\kappa = \frac{\theta - \theta_2}{1 - \theta_2}$$

Its value is usually between 0 and 1, although a value lower than 0 is possible, meaning that the two classifiers agree less than two random classifiers.

In a shared ensemble, the similarity of each hypothesis with respect to the combined hypothesis would be unfeasible to compute, because there would be an exponential number of comparisons. Hence, we are interested in measuring the similarity *for each node* with respect to the combined solution, taking into account just the examples of the invented dataset that fall into a node.

The idea is, in general, that once the multi-tree is constructed, we use its combination to predict the classes for the previously unlabelled invented dataset. Given an example e from the unlabelled invented dataset, this example will fall into different OR-nodes and finally into different leaves, giving different class vectors. Then, the invented dataset is labelled by voting these predictions.

Next, we calculate a contingency matrix M (initialised to 0) for each node (internal or leaf): for each example in the labelled invented dataset, we increment by 1 the cell $M_{a,b}$ of each leaf where the example falls, with a being the predicted class by the leaf and b being the predicted class by the combination. When all the examples have been evaluated and the matrices in the leaf nodes have been assigned, then we propagate the matrices upwards as follows:

– For the contingency matrix M of AND-nodes we accumulate the contingency matrix of their m children nodes: $(M_1 + M_2 + \cdots + M_m)$.
– For the contingency matrix M of OR-nodes, the node of their children with greater Kappa (or other similarity measure) is selected and its matrix is propagated upwards. The selected node is marked.

This ultimately generates the hypothesis that is most similar to the combined hypothesis, using a particular invented dataset and a given similarity measure.

Our experimental evaluation of the *archetype* method has shown that it is very dependent on the measure of similarity used, with κ being the best metric. With this metric we obtain satisfactory results; the archetype hypothesis obtains higher accuracy than the first single hypothesis with the lowest number of rules (Occam solution), and it is approximately in the middle between the latter and the combined solution.

6 Learning Cost-sensitive Decision Trees

Traditionally, classification accuracy (or error), i.e., the percentage of instances that are correctly classified (respectively incorrectly classified) has been used as a measure of the quality of classifiers. However, in many situations, not every misclassification has the same consequences, and problem-dependent misclassification costs have to be taken into account. However, it is common that the cost parameters are not known at training time. In this case, Receiver Operating Characteristic (ROC) analysis can be applied [18, 22]. ROC analysis provides tools to distinguish classifiers that are optimal under some class and cost distributions from classifiers that are always sub-optimal, and to select the optimal classifier once the cost parameters are known.

There are several methods to take costs into account when learning decision trees. If costs are known at training time, the training algorithm could be made cost-sensitive, e.g. by incorporating costs in the splitting criterion. However, it has been shown that such cost-sensitive techniques do not lead to trees with lower costs [1, 4] and that cost-sensitive class labelling is more effective [1]. Given an example with 2 classes, if we assume that costs are unknown at training time, each of the 2^n possible labellings of the n leaves of a given decision tree establishes a classifier, and we can use ROC analysis to determine the optimal labellings among them. However, this set of classifiers has special properties (e.g., for any classifier there is another one making opposite predictions) which allows a more direct computation of the optimal labellings. In [7], we proved that there are n+1 of these, which are determined by a simple ordering on the leaves of the tree.

Thus, from a cost-sensitive perspective it makes sense to view a decision tree as an unlabelled tree with an ordering on the leaves. Furthermore, this suggests the use of the area under the ROC curve (AUC), obtained by plotting the n+1 optimal labellings in ROC space, to evaluate the quality of a decision tree (or any other partitioning of instance space) instead of accuracy.

A natural question is then whether existing decision tree algorithms which aim at optimising the accuracy of a single labelling also lead to good AUC values, or whether we can do better by adapting the algorithm. Using the optimal labelling set, AUC of the leaves under a split can be computed efficiently. In particular, given several possible splits for growing the tree, where each split consists of a set of new leaves, we can compute the ordering of these leaves and calculate the corresponding ROC curve. Consequently, we can define a new splitting criterion based on the AUC values. A detailed description of this new splitting criterion, known as AUCsplit, can be found in [7]. Experimental results are shown next.

7 Experiments

In this section we present an experimental evaluation of the ideas presented in the previous sections, as implemented in the SMILES system [10]. SMILES is a multi-purpose machine learning system which includes a multiple decision tree learner with the previous and many other features.

For the experiments, we have used GainRatio [19] as splitting criterion (unless otherwise stated) and we have chosen a random method [9] for populating the shared ensemble (after a solution is found, one suspended OR-node is woken at random). Pruning is not enabled unless stated.

The experiments were performed in a Pentium III-800Mhz with 180MB of memory running Linux 2.4.2. We have used several datasets from the UCI dataset repository [16]. Table 1 shows the dataset name, the size in number of examples, the number of classes and the number of nominal and numerical attributes.

Since there are many sources of randomness, we have performed the experiments by averaging 10 results of a 10-fold cross-validation.

7.1 Comparison with other ensemble methods

Figure 2 presents a comparison of accuracy between our method (shared combination, difference + maximum, using *multi-tree*), *boosting* and *bagging* (all of them without pruning), depending on

#	Dataset	Size	Classes	Nom.Attr.	Num.Attr.
1	Balance-scale	625	3	0	4
2	Cars	1728	4	5	0
3	Dermatology	358	6	33	1
4	Ecoli	336	8	0	7
5	Iris	150	3	0	4
6	House-votes	435	2	16	0
7	Monks1	566	2	6	0
8	Monks2	601	2	6	0
9	Monks3	554	2	6	0
10	New-thyroid	215	3	0	5
11	Post-operative	87	3	7	1
12	Soybean-small	35	4	35	0
13	Tae	151	3	2	3
14	Tic-tac	958	2	8	0
15	Wine	178	3	0	13

Table 1. Information about datasets used in the experiments.

the number of iterations. We have employed the Weka[1] implementation of these two ensemble methods.

Although initially our method obtains lower results with few iterations, with a higher number of iterations it surpasses the other systems. Probably the slow increase of accuracy in the multi-tree method is due to the random selection of the OR-nodes to be explored.

Nevertheless, the major advantage of the method is appreciated by looking at the consumption of resources. Figure 3 shows the average training time depending on the number of iterations (1-300) for the three methods. Note that the time increase of *bagging* is linear, as expected. *Boosting* behaves better with high values because the algorithm implemented in Weka trickily stops the learning if it does not detect a significant increase of accuracy. Finally, SMILES presents a sub-linear increase of required time due to the sharing of common components of the multi-tree structure.

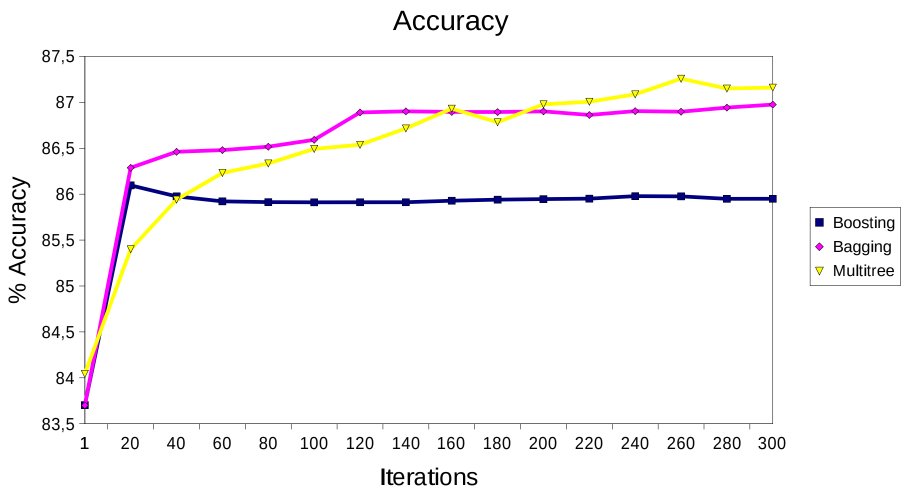

Fig. 2. Accuracy comparison between ensemble methods.

7.2 Archetype classifier

An experimental evaluation of the archetyoe method is presented in Table 2. In this table we show the accuracy of the first single solution, and the accuracy of the combination, the archetype solution and the Occam solution for multi-trees created by exploring 10, 100, and 1000 alternative OR-nodes.

The results confirm the usefulness of the archetype proposal: by just exploring 10 OR-nodes the archetype solution surpasses the first solution and the Occam solution. This difference is increased as long as the multi-tree is populated. This is mainly due to the improvement of the accuracy of the combined solution and the fact that the archetype hypothesis can actually get close to it.

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`

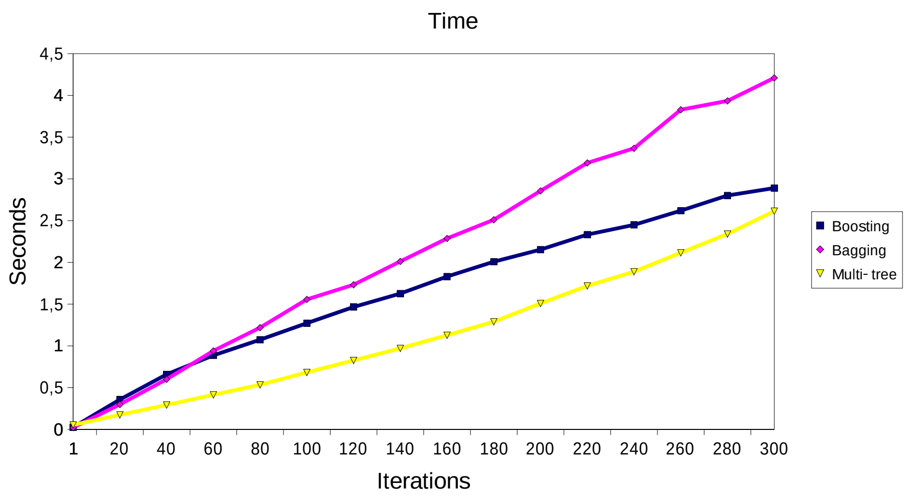

Fig. 3. Time comparison between ensemble methods.

	1	10			100			1000		
#	1st	Comb	Arc	Occ	Comb	Arc	Occ	Comb	Arc	Occ
1	76.82	77.92	77.18	76.81	83.11	80.08	76.74	88.02	83.52	76.77
2	89.01	89.59	89.13	89.03	91.05	89.65	89.08	91.57	90.02	89.11
3	89.80	91.29	90.63	90.09	93.57	90.57	90.20	94.06	91.31	90.69
4	77.55	79.06	77.61	77.79	79.85	79.36	78.36	80.73	77.73	77.48
5	93.63	94.86	94.19	93.93	95.95	94.44	93.63	95.72	94.12	93.93
6	94.67	94.73	93.67	94.73	94.60	93.13	94.40	95.60	94.13	94.80
7	92.25	96.18	95.98	96.47	100.00	100.00	100.00	100.00	100.00	100.00
8	74.83	74.85	74.32	73.75	77.35	76.10	72.53	82.33	82.05	70.43
9	97.55	97.75	97.67	97.60	97.53	97.58	97.55	97.71	97.71	97.62
10	92.62	93.19	92.57	92.57	92.57	92.76	92.95	90.67	92.62	93.67
11	60.88	63.38	61.63	60.00	66.13	63.88	62.25	68.50	65.75	62.13
12	97.25	97.00	98.00	97.50	96.50	96.50	96.75	95.00	93.25	96.25
13	62.93	64.40	63.93	61.93	65.53	62.93	62.07	65.20	62.00	61.13
14	78.22	79.03	78.11	78.27	82.68	78.19	78.58	84.62	79.80	79.47
15	93.12	93.12	92.82	92.94	92.59	92.06	93.24	91.53	92.35	94.41
gmeans	83.87	84.94	84.29	83.93	86.56	85.00	84.28	87.47	85.67	84.19

Table 2. Influence of the size of the multi-tree on the archetype.

The Occam solution does not seem to be improved by larger multi-trees. Nevertheless, the Occam hypothesis can also be regarded as a way to obtain more and more compact solutions without losing accuracy.

7.3 AUC criterion evaluation

For the evaluation of the AUC criterion we employed 25 datasets extracted from the UCI repository [16]. All of them have two classes, either originally or by selecting one of the classes and joining all the other classes.

Table 3 lists the accuracy of the best single labelling and the AUC values of the whole set of optimal labellings for Gain Ratio and AUCSplit, this time both with pruning. The first thing that can be observed is that the differences in accuracy are small, but the differences in terms of AUC are more relevant.

Since means of different datasets are illustrative but not reliable we compare dataset by dataset to see whether one method is better than the other. The 'Better?' column represents if AUCsplit behaves better ($\surd$) or worse (x) than Gain Ratio. These marks are only shown when the differences are significant according to the t-test with level of confidence 0.1. This gives 8 wins, 13 ties and 4 loses for accuracies and 11 wins, 11 ties and 3 loses for AUC. The experiments have been extracted from [7].

	GainRatio		AUCSplit		Better?	
Dataset	Acc.	AUC	Acc.	AUC	Acc.	AUC
Monks1	90.7±6.6	83.6±11.8	96.5±3.9	94.3±6.7	√	√
Monks2	57.7±6.5	61.1±7.9	56.0±6.2	56.7±8.0	x	x
Monks3	97.6±7.8	97.4±8.5	99.1±1.1	99.1±1.4	√	√
Tic-tac	78.9±4.6	79.8±7.2	77.6±4.7	76.9±6.5	x	x
House-votes	95.8±2.6	95.2±3.1	95.8±2.6	95.2±3.1		
Agaricus	1±0	1±0	1±0	1±0		
Breast-wdbc	92.5±4.1	91.5±6.1	92.9±3.7	94.7±4.6		√
Breast-wpbc	72.1±10.2	61.3±16.9	69.5±10.6	59.3±16.2	x	
Ionosphere	92.0±4.7	90.4±7.0	89.6±5.0	89.7±6.7	x	
Liver	62.6±8.8	64.2±10.6	64.0±9.0	65.8±10.1		
Pima	73.3±5.7	76.6±6.9	72.5±5.1	76.7±6.0		
Chess-kr-vs-kp	99.1±2.3	99.5±1.6	99.2±0.6	99.5±0.6		
Sonar	68.2±10.2	67.4±11.9	71.0±10.4	73.6±11.0	√	√
Breast-Cancer	95.4±2.5	96.3±2.5	96.2±2.5	97.6±2.1	√	√
Hepatitis	86.4±14.2	85.1±17.9	83.4±14.0	63.5±22.3		x
Thyroid-hypo	98.0±10.9	84.6±13.1	98.6±0.8	94.8±5.6	√	√
Thyroid-sick-eu	95.2±1.4	92.6±3.5	96.7±1.2	95.1±3.1	√	√
tae [0]	71.4±12.4	61.5±20.8	68.9±11.6	59.8±21.3		
cars [unacc]	95.0±1.8	98.2±0.9	94.8±1.9	98.1±1.0		
nursery [nr]	1±0	1±0	1±0	1±0		
pendigits [0]	99.6±0.3	99.6±0.5	99.6±0.2	99.4±0.6		
page-blocks [0]	96.8±0.9	93.3±4.7	96.8±0.2	95.1±6.9		√
yeast [ERL]	70.4±3.9	72.2±4.9	71.1±3.6	73.3±4.0		√
letter [a]	99.5±0.2	98.9±1.4	99.5±0.1	99.3±0.7	√	√
optdigits [0]	98.9±1.8	94.2±19.4	99.5±0.3	98.5±1.8	√	√
M.	87.49	85.78	87.55	86.24		

Table 3. Accuracy and AUC for Gain Ratio and AUCSplit.

8 Supplementary Applications

In previous subsections we have talked about comprehensible languages that become incomprehensible because of the use of ensemble methods and combined hypotheses. However, there are other non-ensemble machine learning methods that are originally incomprehensible, e.g. neural networks [13] or support vector machines [6]. In some situations they obtain hypotheses with higher accuracies but with the drawback of being a black box.

The idea is that the oracle we have referred to in section 5 does not need to be an internal combined hypothesis but it can be any external source, such as a neural network. Therefore, this could be regarded as a new method to "convert" incomprehensible neural networks (or other models) to comprehensible decision trees (with possibly a slight loss in accuracy). This approach is much more flexible than any other rule extraction method because it can be applied to any blackbox source and any comprehensible destination. Figure 4 shows the described process where a neural network is learned (e.g. by using the WEKA system). In parallel, the SMILES system learns a multi-tree. The neural network is used to label an invented dataset and the final labelled dataset is input to SMILES to select the single hypothesis from the multi-tree that is most similar to the neural network classifier.

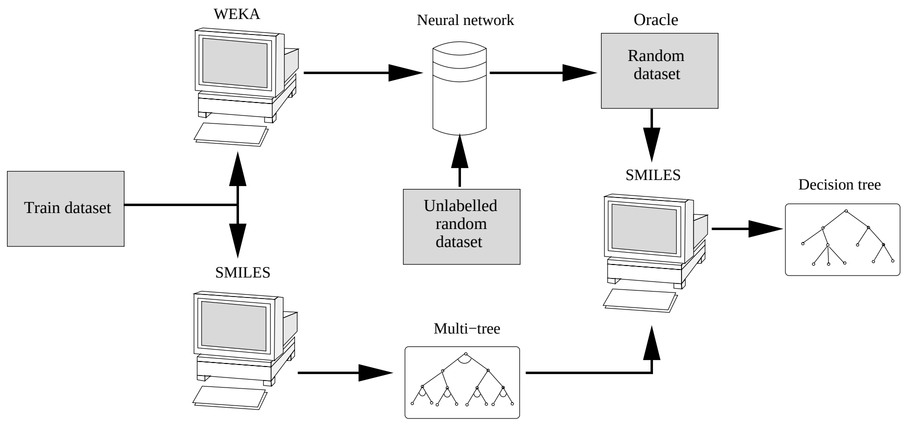

Fig. 4. Representation of the selection of a decision tree by an external oracle

9 Conclusions and Future Work

In this paper we have presented a re-design of decision tree learning that aims at reducing several costs associated with the application of machine learning techniques. We have presented a new multi-tree structure, new combination and fusion methods, new evaluation metrics and splitting criteria based on the Area Under the ROC Curve, a new method for extracting the archetype from an ensemble and its applications to translate incomprehensible models into comprehensible ones. As a consequence, the costs associated with machine learning methods, such as computational (space and time) costs, error costs, incomprehensibility costs, are highly reduced by our approach.

As future work we are primarily focussing on an extension of expressiveness of our framework. We have been designing new partitions to tackle constructor terms (tags, lists, sets, etc.) useful in many applications (web mining, xml mining, relational data mining, etc.). This would make it possible for a homogeneous input of examples and background knowledge and output of models in XML format. As a long-term work, we plan to extend decision tree learning with higher-order features, in order to better exploit background knowledge.

References

1. J. P. Bradford, C. Kunz, R. Kohavi, and C. Brunk. Pruning decision trees with misclassification costs. *Lecture Notes in Computer Science*, 1398, 1998.
2. J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psych. Meas.*, 20:37–46, 1960.
3. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139–157, 2000.
4. C. Drummond and R.C. Holte. Explicitly representing expected cost: An alternative to roc representation. In *Knowledge Discovery and Data Mining*, pages 198–207, 2000.
5. V. Estruch, C. Ferri, J. Hernández, and M.J. Ramírez. Shared Ensembles using Multi-trees. In *the 8th Ib. Conf. on A. I., Iberamia'02.* (Submitted), 2002.
6. Theodoros Evgeniou and Massimiliano Pontil. Support vector machines: Theory and applications. *Lecture Notes in Computer Science*, 2049, 2001.
7. C. Ferri, P. Flach, and J. Hernández. Learning Decision Trees using the Area Under the ROC Curve. In *Int. Conf. on Machine Learning, ICML'02.* (To appear), 2002.
8. C. Ferri, J. Hernández, and M.J. Ramírez. Induction of Decision Multi-trees using Levin Search. In *Int. Conf. on Computational Science, ICCS'02*, LNCS, 2002.
9. C. Ferri, J. Hernández, and M.J. Ramírez. Learning multiple and different hypotheses. Technical report, D.S.I.C., Universitat Politècnica de València, 2002.
10. C. Ferri, J. Hernández, and M.J. Ramírez. SMILES system, a multi-purpose learning system. `http://www.dsic.upv.es/~flip/smiles/`, 2002.
11. Ludmila I. Kuncheva. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.
12. Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Submitted to Machine Learning, 2002.
13. Clifford Lau, editor. *Neural Networks: Theoretical Foundations and Analysis.* IEEE, 1992.
14. Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *14th Int. Conf. on Machine Learning*, pages 211–218. Morgan Kaufmann, 1997.
15. N.J. Nilsson. *Artificial Intelligence: a new synthesis.* Morgan Kaufmann, 1998.
16. University of California. UCI Machine Learning Repository Content Summary. `http://www.ics.uci.edu/~mlearn/MLSummary.html`.
17. J. Pearl. *Heuristics: Intelligence search strategies for computer problem solving.* Addison Wesley, 1985.
18. Foster Provost and Tom Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 43. AAAI Press, 1997.
19. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
20. J. R. Quinlan. Bagging, Boosting, and C4.5. In *Proc. of the 13th Nat. Conf. on A.I. and the 8th Innovative Applications of A.I. Conf.*, pages 725–730. AAAI/MIT Press, 1996.
21. Ross Quinlan. Relational learning and boosting. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 292–306. Springer-Verlag, September 2001.
22. J.A. Swets, R.M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283(4), 2000.