From Ensemble Methods To Comprehensible Models*

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

DSIC, UPV, Camino de Vera s/n, 46020 Valencia, Spain.
{cferri,jorallo,mramirez}@dsic.upv.es

Abstract. Ensemble methods improve accuracy by combining the predictions of a set of different hypotheses. However, there are two important shortcomings associated with ensemble methods. Huge amounts of memory are required to store a set of multiple hypotheses and, more importantly, comprehensibility of a single hypothesis is lost. In this work, we devise a new method to extract one single solution from a hypothesis ensemble without using extra data, based on two main ideas: the selected solution must be similar, semantically, to the combined solution, and this similarity is evaluated through the use of a random dataset. We have implemented the method using shared ensembles, because it allows for an exponential number of potential base hypotheses. We include several experiments showing that the new method selects a single hypothesis with an accuracy which is reasonably close to the combined hypothesis.

Keywords: Ensemble Methods, Decision Trees, Comprehensibility in Machine Learning, Classifier Similarity, Randomisation.

1 Introduction

Comprehensibility has been the major advantage that has been advocated for supporting some machine learning methods such as decision tree learning, rule learners or ILP. One major feature of discovery is that it gives insight from the models, properties and theories that can be obtained. A model that is not comprehensible may be useful to obtain good predictions, but it cannot provide knowledge about how predictions are made.

With the goal of improving model accuracy, there has been an increasing interest in constructing ensemble methods that combine several hypotheses [4]. The effectiveness of combination is further increased the more diverse and numerous the set of hypotheses is [10]. Decision tree learning (either propositional or relational) is especially benefited by ensemble methods [18, 19]. Well-known techniques for generating and combining hypotheses are boosting [9, 18], bagging [1, 18], randomisation [5], stacking [22] and windowing [17].

* 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-Alemana HA2001-0059.

Although ensemble methods significantly increase accuracy, they have some drawbacks, mainly the loss of comprehensibility of the model and the large amount of memory required to store the hypotheses [13].

Recent proposals have shown that memory requirements can be considerably reduced (in [16], a method called miniboosting reduces the ensemble to just three hypotheses, with 40% less of the improvement that would be obtained by a 10-trial AdaBoost). Nonetheless, the comprehensibility of the resulting combined hypothesis is not improved. A combined hypothesis is usually a voting of many hypotheses and it is usually treated as a black box, giving no insight at all.

However, one major goal of the methods used in discovery science is comprehensibility. The question is how to reduce to one single hypothesis from the combination of m hypotheses without losing too much accuracy with respect to the combined hypothesis. Instead of using classical methods for selecting one hypothesis, such as the hypothesis with the lowest expected error, or the one with the smallest size (Occam's razor), we will select the single hypothesis that is most similar to the combined hypothesis. This single hypothesis will be called an *archetype* or *representative* of the ensemble and can be seen as an 'explanation' of the ensemble.

To do this, the main idea is to consider the combination as an oracle that would allow us to measure the similarity of each single hypothesis with respect to this oracle. More precisely, for a hypothesis or solution h and an unlabelled example e, let us define $h(e)$ as the class or label assigned to e by h. Consider an ensemble of solutions $\mathcal{E} = h_1, h_2, \cdots h_m$ and a method of combination χ. By $\Sigma_{\chi,\mathcal{E}}$ we denote the combined solution formed by using the method χ on $\mathcal{E}$. Thus, $\Sigma_{\chi,\mathcal{E}}(e)$ is the class assigned to e by the combined solution. Now, we can use $\Sigma_{\chi,\mathcal{E}}$ as an oracle, which, generally, gives better results than any single hypothesis [4]. The question is to select a single hypothesis h_i from $\mathcal{E}$ such that h_i is the most similar (semantically) to the oracle $\Sigma_{\chi,\mathcal{E}}$.

This rationale is easy to understand following the representation used in a statistical justification for the construction of good ensembles presented by Dietterich in [4]. A learning algorithm is employed to find different hypotheses $\{h_1, h_2, \cdots, h_m\}$ in the hypothesis space or language $\mathcal{H}$. By constructing an ensemble out of all these classifiers, the algorithm can "average" their votes and reduce the risk of choosing a wrong classifier. Figure 1 depicts this situation. The outer curve denotes the hypothesis space $\mathcal{H}$. The inner curve denotes the set of hypotheses that give a reasonably good accuracy on the training data and hence could be generated by the algorithm. The point labelled by F is the true hypothesis.

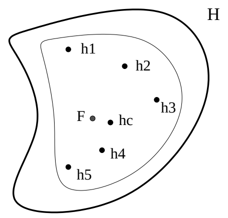

Fig. 1. Representation of an ensemble of hypotheses

If an ensemble h_c is constructed by combining the accurate hypotheses, h_c is a good approximation to F. However, h_c is an ensemble, which means that it needs to store $\{h_1, h_2, \cdots, h_5\}$ and it is not comprehensible. For this reason, we are interested in selecting the single hypothesis from $\{h_1, h_2, \cdots, h_m\}$ that would be closest to the combination h_c. Following the previous rationale, this single hypothesis would be close to F. In the situation described in Figure 1, we would select h_4 as the *archetype* or *representative* of the ensemble.

A final question, also pointed out by [4], is that a statistical problem arises when the amount of training data available is too small compared to the size of the hypothesis space $\mathcal{H}$. The selection of a good archetype would not be possible if a sufficient amount of data is not available for comparing the hypotheses. The reserve of part of the training data is generally not a good option because it would yield a smaller training dataset and the ensemble would have a lower quality. This problem has a peculiar but simple solution: the generation of random unlabelled datasets.

Although the technique presented in this work is applicable to many kinds of ensemble methods, we will illustrate it with *shared* ensembles, because the number of hypotheses, in this kind of structure, grows exponentially wrt. the number of iterations. Therefore, there is a much bigger population which the representative can be extracted from.

The paper is organised as follows. First, in section 2, we discuss the use of a similarity measure and we adapt several similarity metrics we will use. Section 3 explains how artificial datasets can be employed to estimate the similarity between every classifier and their combination. Section 4 presents the notion of shared ensemble, its advantages for our goals and how it can be adapted for the selection of the most similar hypothesis with respect to the combination. A thorough experimental evaluation is included in Section 5. Finally, the last section presents the conclusions and proposes some future work.

2 Hypothesis Similarity Metrics

As we have stated, our proposal is to select the single hypothesis which is most similar to the combined one. Consequently, we have to introduce different measures of hypothesis similarity. These metrics and an additional dataset will allow the estimation of a value of similarity between two hypotheses. In the following, we will restrict our discussion to classification problems.

Several measures of hypothesis similarity (or diversity) have been considered in the literature with the aim of obtaining an ensemble with high diversity [12]. However, some of these are defined for a set of hypotheses and others for a pair of hypotheses. We are interested in these "pairwise diversity measures", since we want to compare a single hypothesis with an oracle. However, not all of these measures can be applied here. First, the approach presented by [12] requires the correct class to be known. The additional dataset should be labelled, which means that part of the training set should be reserved for the estimation of similarities. Secondly, some other metrics are only applicable to two classes. As a result, in what follows, we describe the pairwise metrics that can be estimated by using an unlabelled dataset and that can be used for problems with more than two classes.

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 $h_b(e) = j$. With this matrix, we can define the following similarity metrics:

- θ **measure**: It is just based on the idea of determining the probability of both classifiers agreeing:

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

  Its value is between 0 and 1. An inverse measure, known as discrepancy is also considered 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 [13], we define the Kappa measure, which was originally introduced as the Kappa statistic (κ) [3]. 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 agree.
- Q **measure**: The Q measure is defined as follows [12]:

$$Q = \frac{\prod_{i=1}^{C} M_{i,i} - \prod_{i=1,j=1,i \neq j}^{C} M_{i,j}}{\prod_{i=1}^{C} M_{i,i} + \prod_{i=1,j=1,i \neq j}^{C} M_{i,j}}$$

  This value varies between -1 and 1. Note that this measure may have problems if any component of M is 0. Thus it is convenient to apply smoothing in M to compute the measure. We will add 1 to every cell.

Obviously, the greater the reference dataset is, all of the previous metrics give a better estimate of similarity. In our case, and since the previous measures use the contingency matrix, we can have huge reference datasets available: random invented datasets.

3 Random Invented Datasets

In many situations, a single hypothesis may be the one which is the most similar to the combined hypothesis *with respect to the training set*, however it may not be the most similar one in general (*with respect to other datasets*). In some cases, e.g. if we do not use pruning, then all the hypotheses (and hence the combined solution) may have 100% accuracy with respect to the training set, and all the hypotheses are equally "good". 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 then is to use the entire training set to construct the hypotheses and to use a *random* dataset to select one of them. In this work, we consider that the examples in the training set are equations of the form $f(\cdots) = c$, where f is a function symbol and c is the class of the term $f(\cdots)$. Given a function f with a arguments, an unlabelled random example is any instance of the term $f(X_1, X_2, \cdots, X_a)$, i.e., any term of the form $f(v_1, v_2, \cdots, v_a)$ obtained by replacing every attribute X_i by values v_i from the attribute domain (attribute type). Note that an unlabelled random example is not an equation (a full example) because we include no information about the correct class.

We will use the following technique to generate each random unlabelled example: each attribute X_i of a new example is obtained as the value v_i in a different example $f(v_1, \ldots, v_i, \ldots, v_a)$ selected from the training set by using a uniform distribution. This procedure of generating instances assumes that all the attributes are independent, and just maintains the probabilities of appearance of the different values observed in each attribute of the training dataset.

4 Shared Ensembles

A multi-tree is a data structure that permits the learning of ensembles of trees that share part of their branches. These are called "shared ensembles". In the particular case of trees, a multi-tree can be based on an AND/OR organisation, where some alternative splits are also explored. Note that a multi-tree is not a forest [10], because a multi-tree shares the common parts of different trees, whereas a forest is just a collection of trees.

In a previous work [6], we presented an algorithm for the induction of multi-trees which is able to obtain several hypotheses, either by looking for the best one or by combining them in order to improve accuracy. To do this, once a node has been selected to be split (an AND-node) the possible splits below (OR-nodes) are evaluated. The best one, according to the splitting criterion is selected and the rest are suspended and stored. After is completed the first solution, when a new solution is required, one of the suspended nodes is chosen and 'woken', and the tree construction follows under this node. This way, the search space is an AND/OR tree [14] which is traversed, thus producing an increasing number of solutions as the execution time increases. In [7], we presented several methods for growing the multi-tree structure. Since each new solution is built by completing a different alternative OR-node branch, our method differs from other approaches such as the boosting or bagging methods [1, 9, 18] which would induce a new decision tree for each solution.

Note that in a multi-tree structure there is an exponential number of possible hypotheses with respect to the number of alternative OR-nodes explored. Consequently, although the use of multi-trees for combining hypotheses is more complex, it is more powerful because it allows us to combine many more hypotheses using the same resources. Other previous works have explored the entire the search space of the AND/OR tree to make the combination [2], inspired by Context Tree Weighting (CTW) models [20], whereas we only explore a subset of the best trees.

4.1 Shared Ensemble Combination

Given several classifiers that assign a probability to each prediction (also known as soft classifiers) there are several combination methods or fusion strategies that can be applied. Let us denote by $p_k(c_j|x)$ an estimate of the posterior probability that classifier k assigns class c_j for example x.

If we consider all the estimates equally reliable we can define several fusion strategies: majority vote, sum or arithmetic mean, product or geometric mean, maximum, minimum and median. Some works have studied which strategy is best. In particular, [11] concludes that, for two-class problems, minimum and maximum are the best strategies, followed by average (arithmetic mean).

In decision tree learning, the $p_k(c_j|x)$ depend on the leaf node where each x falls. More precisely, these probabilities depend on the *proportion* of training examples of each class that have fallen into each node during training. The reliability of each node usually depends on the *cardinality* of the node.

Let us define a *class vector* $v_{k,j}(x)$ as the vector of training cases that fall in each node k for each class j. For leaf nodes the values would be the training cases of each class that have fallen into the leaf. To propagate upwards these vectors to internal nodes, we must clarify how to propagate through AND and OR nodes. This is done for each new unlabelled example we want to make a prediction for. For the AND-nodes, the answer is clear: an example can only fall through an AND-node. Hence, the vector would be the one of the child where the example falls. OR-nodes, however, must do a fusion whenever different alternative vectors occur. This is an important difference in shared ensembles: fusion points are distributed all over the multi-tree structure.

We have implemented several fusion strategies. Nonetheless, it is not the goal of this paper to evaluate different methods for combining hypotheses but to select a single hypothesis. Thus, for the sake of simplicity, in this paper we will only use the *maximum* strategy because it obtains the best performance, according to our own experiments and those of [11].

4.2 Selecting an Archetype from a Shared Ensemble

In a shared ensemble, we are not interested (because it would be unfeasible) to compute the similarity of each hypothesis with respect to the combined hypothesis, because there would be an exponential number of comparisons. What we are interested in is a measure of similarity for each node with respect to the combined solution, taking into account only the examples of the invented dataset that fall into a node.

The general idea is, 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 in the way explained in the previous subsection.

After this step, we can calculate a contingency matrix for each node, in the following way. For each node (internal or leaf), we have a $C \times C$ contingency matrix called M, initialised to 0, where C is the number of classes. For each example in the labelled invented dataset, we increment the cell $M_{a,b}$ of each leaf where the example falls by 1, 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.

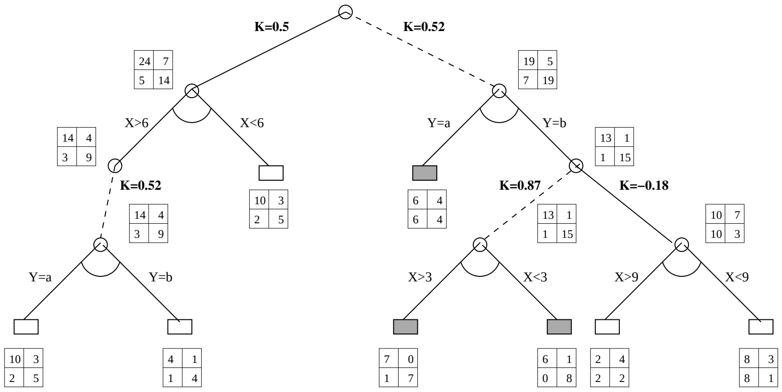

Fig. 2. Selection of a single decision tree from the multi-tree structure.

Figure 2 shows the selection of one hypothesis from a multi-tree according to the contingency matrix. The AND-nodes are represented with an arc. The leaves are represented by rectangles. First, we fill the matrices of the leaves. Then, we propagate these upwards as has been detailed previously. Finally, when we reach the top of the tree, it is straightforward to extract the solution by simply descending the multi-tree by the marked nodes. In the figure, the marked nodes are represented by the dashed lines, and the leaves of the selected hypothesis are shadowed.

Therefore, we can summarise the approach in five different steps:

1. **Multi-tree generation**: The first step consists in the generation of a multi-tree from a training dataset. There are some criteria which affect the quality of the multi-tree: the splitting criterion, the pruning method, and the criterion for the selection of the suspended node to be woken.
2. **Invented dataset**: In this phase, an unlabelled invented dataset is created, by a random dataset.
3. **Multi-tree combination**: The invented dataset is labelled by the combination of the shared ensemble. A method of combination of hypotheses can be specified.
4. **Calculation and propagation of contingency matrices**: A contingency matrix is assigned to each node of the multi-tree, using the labelled invented dataset and a similarity metric.
5. **Selection of a solution**: An archetype hypothesis is extracted from the multi-tree by descending the multi-tree through the marked nodes.

5 Experiments

In this section, we present an experimental evaluation of our approach, as it is implemented in the SMILES system [8]. SMILES is a multi-purpose machine learning system which includes (among many other features) the implementation of a multi-tree learner. For the experiments, we used GainRatio [17] as splitting criterion. We chose a random method [7] for populating the shared ensemble (after a solution is found, a suspended OR-node is woken at random) and we used the *maximum* strategy for combination.

We used several datasets from the UCI dataset repository [15]. Table 1 shows the dataset name, the size in number of examples, the number of classes, the nominal and numerical attributes.

#	Dataset	Size	Classes	Nom.Attr.	Num.Attr.
1	monks1	566	2	6	0
2	monks2	601	2	6	0
3	monks3	554	2	6	0
4	tic-tac	958	2	8	0
5	house-votes	435	2	16	0
6	post-operative	87	3	7	1
7	balance-scale	625	3	0	4
8	soybean-small	35	4	35	0
9	dermatology	358	6	33	1
10	cars	1728	4	5	0
11	tae	151	3	2	3
12	new-thyroid	215	3	0	5
13	ecoli	336	8	0	7

Table 1. Information about datasets used in the experiments.

Since there are many sources of randomness, we have performed the experiments by averaging 10 results of a 10-fold cross-validation. This makes a total of 100 runs (each one with a different multi-tree construction, random dataset and hypothesis selection process) for each pair of method and dataset.

In the experiments, we will use the following notation:

- *First Solution*: this is the solution given by just one hypothesis (the first hypothesis that is obtained). This is similar to C4.5 [17].
- *Combined Solution*: this is the solution given by combining the results of the ensemble (in our case, the multi-tree, as described in the previous section).
- *Archetype Solution*: this is the single solution which is most similar to the combined solution.
- *Occam Solution*: this is the single solution with the lowest number of rules, i.e., the shortest solution.

It is not our purpose to evaluate the improvement of the *Combined Solution* over the *First Solution* using shared ensembles. We have done that in previous works [7]. We have not included the results using post-pruning because it does not improve the performance of any of the four kinds of solutions.

Our goal is to show that a significant gain can be obtained from the *First Solution* to the *Archetype* and *Occam* methods as long as the size of the ensemble increases. Another question to be answered is to determine which method to extract a single solution from an ensemble is better: *Archetype* or *Occam*.

5.1 Evaluating Similarity Metrics

Table 2 shows the accuracy for each pair composed of a dataset and a method and the geometric means for each method. The methods studied are *First*, *Combined* and *Archetype*. The latter uses three different similarity metrics κ, θ and Q. The multi-tree has been generated exploring 100 suspended OR-nodes.

#	1st	Comb	Arc. κ	Arc. θ	Arc. Q
1	92.3	100	100	100	100
2	74.8	77.4	76.1	76.2	75.8
3	97.5	97.5	97.6	97.6	97.6
4	78.2	82.7	78.2	78.3	78.5
5	93.6	96.0	94.4	93.9	94.2
6	60.9	66.3	63.8	64.3	61.9
7	76.8	83.1	80.1	80.1	79.8
8	97.3	96.5	96.5	91.0	47.0
9	89.8	93.6	90.6	89.9	74.3
10	89.0	91.0	89.6	89.6	89.3
11	62.9	64.5	61.9	62.9	49.8
12	92.6	92.6	92.8	92.9	91.4
13	77.5	79.9	79.4	78.9	76.7
gmeans	82.41	85.45	83.78	83.45	76.24

Table 2. Comparison between measures of similarity.

As expected, hypothesis combination improves the accuracy w.r.t. the first single tree. The use of the *Archetype* method also obtains good results. On the other hand, the results show that the *Archetype* method is very dependent on the measure of similarity used: κ seems to be the best metric and Q the worst (it even obtains lower accuracy than the first single hypothesis).

5.2 Influence of the Size of the Invented Dataset

Similarity is approximated through the use of an invented dataset. Let us study the influence of its size, varying from 10 to 100,000 examples. The similarity

		10	100	1000	10000	100000
#	Comb	Arc	Arc	Arc	Arc	Arc
1	99.8	72.3	93.3	99.8	100	99.9
2	77.3	64.6	61.0	75.2	76.1	76.2
3	97.6	82.9	94.5	97.6	97.6	97.6
4	82.9	65.9	70.3	78.0	78.2	78.6
5	95.8	73.7	92.4	94.4	94.4	93.8
6	67.5	69.1	63.6	63.9	63.8	63.5
7	83.0	62.5	75.4	79.4	80.1	79.9
8	95.0	68.8	93.3	95.0	96.5	96.5
9	93.6	45.6	84.7	90.5	90.6	89.9
10	91.0	71.0	75.4	88.1	89.6	89.8
11	63.7	44.3	54.3	59.1	61.9	61.2
12	92.5	73.8	89.3	91.3	92.8	92.6
13	80.0	46.8	73.9	77.9	79.4	79.0
gmeans	85.36	63.57	77.40	82.88	83.78	83.57

Table 3. Influence of the size of the invented dataset.

metric and the size of the multi-tree are fixed to κ and 100 alternative opened OR-trees, respectively.

Table 3 shows that in order to obtain a good archetype hypothesis, the similarity metric has to be computed as accurately as possible. Although it depends on the dataset, a size of 10,000 invented examples seems to be sufficient.

5.3 Influence of the Size of the Ensemble

The effect of the size of the multi-tree is evaluated in Table 4. 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[1] alternative OR-nodes. We also include the geometric average number of solutions in the multi-tree (*#Sol)*. Note that with 100 OR-nodes, we obtain millions of solutions with much less required memory than 100 non-shared hypotheses.

The results are quite encouraging: by simply 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 in the accuracy of the combined solution and the fact that the archetype hypothesis can actually get close to it. 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.

6 Conclusions

This work has presented a novel method for extracting a single solution from an ensemble of solutions without removing training data for validation. The most closely related work is Quinlan's miniboosting [16]. However, Quinlan's method can be considered an ensemble method which generates three trees, followed by a merging stage where a single but quite complex tree could be obtained. Moreover, as he recognises, "although it is (usually) possible to construct a

[1] The experiments for datasets 9 and 13 have been performed exploring only 300 and 500 alternative OR-nodes, respectively.

	1	10				100				1000			
#	1st	Comb	Arc	Occ	#Sol	Comb	Arc	Occ	#Sol	Comb	Arc	Occ	#Sol
1	92.3	96.1	96.0	96.5	107	100	100	100	8.7×10^{8}	100	100	100	1.6×10^{19}
2	74.8	74.9	74.3	74.3	148	77.4	76.1	72.5	2.6×10^{10}	82.3	82.1	70.4	3.2×10^{20}
3	97.5	97.7	97.7	97.6	46	97.5	97.6	97.5	80×10^{4}	97.7	97.7	97.6	7.1×10^{14}
4	78.2	79.0	78.1	78.3	257	82.7	78.2	78.6	2.7×10^{12}	84.6	79.8	79.5	3.1×10^{38}
5	93.6	94.9	94.2	93.9	63	96.0	94.4	93.6	26×10^{5}	95.7	94.1	93.9	5.6×10^{11}
6	60.9	63.8	61.8	60.0	55	66.3	63.8	62.3	59674	68.5	65.9	62.1	2.1×10^{9}
7	76.8	77.9	77.2	76.8	131	83.1	80.1	76.7	3.4×10^{8}	88.0	83.5	76.8	1.2×10^{18}
8	97.3	97.0	98.0	97.5	23	96.5	96.5	96.8	38737	95.0	93.3	96.3	1.8×10^{18}
9	89.8	91.3	90.6	90.1	92	93.6	90.6	90.2	3.3×10^{7}	93.8	91.1	90.8	1.2×10^{10}
10	89.0	89.6	89.1	89.0	151	91.0	89.6	89.1	1.7×10^{9}	91.6	90.0	89.1	2.8×10^{24}
11	62.9	62.5	62.3	61.9	97	64.5	61.9	62.1	1.5×10^{6}	64.5	60.9	61.1	4.6×10^{14}
12	92.6	93.2	92.6	92.6	26	92.6	92.8	93.0	3392	90.7	92.6	93.7	6.1×10^{7}
13	77.5	79.1	77.6	77.8	57	79.9	79.4	78.4	1134750	80.3	78.2	77.0	3.8×10^{8}
gm.	82.41	83.49	82.85	82.55	78.31	85.45	83.78	82.91	4.3×10^{7}	86.44	84.49	82.65	6.2×10^{14}

Table 4. Influence of the size of the multi-tree.

single merged decision tree that induces the same partition as a small ensemble, the tree is so large that it conveys no insight. This is a pity, as insight was the prime motivation for producing a single tree".

We overcome the previous problem by producing a single tree that is based on a selection, using the combination as an oracle. Consequently, the result is a single comprehensible solution, an archetype or representative of the ensemble. As we have shown, the single solution obtained by our method is not 100% equivalent to the combination, but in general it gets reasonably close. On the other hand, it is clear that a similar technique could be used for regression models, using, e.g., minimum squared error as a discrepancy (similarity) metric.

From a more general point of view, ensemble methods have been used as an argument in favor of the Epicurus criterion (all consistent models should be retained) and against Occam's razor, because complex combined hypotheses usually obtain better results than the simplest solution [21]. A counterargument may be that the combination is usually expressed outside the hypothesis language. With our work, we have shown that even inside the hypothesis language, the shortest solution is not the best one.

With regard to future work, a mixture of the archetyping method and Occam's razor could also be investigated. Another idea is that the oracle 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 models (with possibly a slight loss in accuracy), which could also be seen as an 'explanation' of the original model.

Finally, it is important to clarify that an archetype solution cannot be obtained without an ensemble, and the quality of the representative would depend on the number of individual hypotheses in the ensemble. Note that this number is exponentially increased by the use of shared ensembles, in particular our multi-tree structure, without requiring a huge amount of memory. Nonetheless, a quite interesting open work would be to study specific methods to generate the ensemble (in our case, to construct the multi-tree) or to investigate the combination method that would produce better oracles for the selection of the archetype.

Acknowledgements

We would like to thank the anonymous reviewers for suggesting the idea of using the archetype as an explanation of the ensemble.

References

1. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
2. J.G. Cleary and L.E. Trigg. Experiences with ob1, an optimal bayes decision tree learner. Technical report, Department of Computer Science, Univ. of Waikato, New Zealand, 1998.
3. J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Meas.*, 20:37–46, 1960.
4. T. G Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.
5. 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.
6. 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.
7. C. Ferri, J. Hernández, and M.J. Ramírez. Learning multiple and different hypotheses. Technical report, Department of Computer Science, Universitat Politécnica de Valéncia, 2002.
8. C. Ferri, J. Hernández, and M.J. Ramírez. SMILES system, a multi-purpose learning system. `http://www.dsic.upv.es/~flip/smiles/`, 2002.
9. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *the 13th Int. Conf. on Machine Learning (ICML'1996)*, pages 148–156, 1996.
10. Tim Kam Ho. C4.5 decision forests. In *Proc. of 14th Intl. Conf. on Pattern Recognition, Brisbane, Australia*, pages 545–549, 1998.
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. Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *14th Int. Conf. on Machine Learning*, pages 211–218. Morgan Kaufmann, 1997.
14. N.J. Nilsson. *Artificial Intelligence: a new synthesis.* Morgan Kaufmann, 1998.
15. University of California. UCI Machine Learning Repository Content Summary. `http://www.ics.uci.edu/~mlearn/MLSummary.html`.
16. J. Quinlan. Miniboosting decision trees. Submitted to JAIR, 1998.
17. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
18. 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.
19. Ross Quinlan. Relational learning and boosting. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 292–306. Springer-Verlag, September 2001.
20. P. Volf and F. Willems. Context maximizing: Finding mdl decision trees. In *Symposium on Information Theory in the Benelux, Vol.15*, pages 192–200, 1994.
21. Geoffrey I. Webb. Further experimental evidence against the utility of Occam's razor. *Journal of Artificial Intelligence Research*, 4:397–417, 1996.
22. David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.