Delegating Classifiers

César Ferri CFERRI@DSIC.UPV.ES
Dep. de Sist. Informàtics i Computació, Univ. Politècnica de València, Spain.

Peter Flach PETER.FLACH@BRISTOL.AC.UK
Department of Computer Science, University of Bristol, UK.

José Hernández-Orallo JORALLO@DSIC.UPV.ES
Dep. de Sist. Informàtics i Computació, Univ. Politècnica de València, Spain.

Abstract

A sensible use of classifiers must be based on the estimated reliability of their predictions. A cautious classifier would delegate the difficult or uncertain predictions to other, possibly more specialised, classifiers. In this paper we analyse and develop this idea of delegating classifiers in a systematic way. First, we design a two-step scenario where a first classifier chooses which examples to classify and delegates the difficult examples to train a second classifier. Secondly, we present an iterated scenario involving an arbitrary number of chained classifiers. We compare these scenarios to classical ensemble methods, such as bagging and boosting. We show experimentally that our approach is not far behind these methods in terms of accuracy, but with several advantages: (i) improved efficiency, since each classifier learns from fewer examples than the previous one; (ii) improved comprehensibility, since each classification derives from a single classifier; and (iii) the possibility to simplify the overall multi-classifier by removing the parts that lead to delegation.

1. Introduction and Motivation

Many recent approaches in machine learning have benefited from the idea that the predictions of a committee or ensemble of models will be usually better than the predictions of one single model. The errors of one can be counteracted by the hits of the other models. This collaborative view of learning can occur without interaction between the learning agents, known as *ensemble learning*, or with interaction during the learning stage, known as *co-learning*. In this paper we explore the novel approach of *delegation*, which can be summarised by the motto: let others do the things that you cannot do well. We introduce the notion of a *cautious classifier* as one that only classifies the examples for which its predictions have high confidence, leaving the rejected examples for another classifier. We demonstrate in this paper that the overall performance of such a hierarchical tandem of classifiers is usually better than a single classifier. By delegating rather than combining models, we avoid some of the problems of ensemble methods, in particular the loss of comprehensibility and the use of excessive computational resources.

From a general point of view, the idea of delegation leads to several questions, around two main issues. First, we have to determine a threshold or decision rule to decide when to apply the first classifier and when to delegate to the second one. Secondly, and more crucially, we have to determine good techniques to generate classifiers that perform better than the first one for the examples that the first one has delegated. With these two main issues in mind, we propose a simple method, with the following features. (1) The decision whether an example has to be tackled by the first classifier or by the second classifier is made by the first classifier itself, by using its own estimated reliability. Because of this, we will use a good ranker (e.g., a good probability estimator) as a first classifier. (2) The second classifier is specialised on the examples for which the first classifier behaves worst by training this second classifier solely with the examples rejected by the first classifier. As we will see in subsequent sections, these issues and the way in which we deal with them still allow a range of alternative approaches, many of them explored in this paper.

We also explore iterated delegating classifiers, which suggests a relationship with other ensemble or combination methods, such as boosting (Freund & Schapire, 1996) and

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the authors.

stacking (Wolpert, 1992). Boosting assigns higher weight to incorrectly classified examples and lower weight to the examples which are classified correctly for each iteration. Delegation, on the other hand, removes the examples which are classified with high confidence and leaves the examples which are classified with lower confidence for subsequent iterations. Stacking builds a second-stage meta-classifier which decides which base classifier (from an independent ensemble of classifiers) to use. Other multiple classifier methods, such as cascading (Gama & Brazdil, 2000), and, specially, local cascading, generate new attributes from the class probability estimations given by the base classifiers or by previous decision tree splits. In contrast, delegation produces models which are completely and exclusively defined in terms of the original attributes and class. Arbitrating (Ortega et al., 2001) and grading (Seewald & Fürnkranz, 2001) are also related to delegation, but both learn external referees to assess the probability of error of each classifier from the pool of base classifiers, and their areas of expertise. No new attributes are generated. Moreover, there is only one base classifier and this one decides which examples to accept and which to reject.

The closest idea to delegation comes from the general separate-and-conquer technique and, specifically, a variant introduced by the PART algorithm (Frank & Witten, 1998), which learns a decision tree, selects the branch with largest coverage, removes the rest of the tree and retrains a second tree with the remaining examples. The process continues until all the examples are covered. Delegation is independent from the base classifier used and it is based on the idea of a confidence threshold, which in the case of decision trees could select several branches with large coverage.

Delegation is hence a serial (not parallel or hierarchical), transferring (no combination), attribute-preserving, self-refereeing, multi-classifier method. The advantages of the delegation approach are manifold. First, since each classifier is keeping part of the examples, the next classifier has fewer examples for training and, hence, the process can be much more efficient than other ensemble methods. Secondly, the resulting overall classifier is not a combination of classifiers, but a decision list; if we use decision trees as base classifiers, the overall classifier is a decision tree, and its decisions can be traced and understood. Thirdly, since some parts of the models will not be used for any example at all, we can, in some cases, simplify the models; e.g., in the case of decision trees, we could prune a subtree if all its leaves lead to delegation.

The rest of the paper is organised as follows. Section 2 defines the notion of cautious classifiers, how they can be constructed from a soft classifier and how they are used to perform delegation. In Section 3 we analyse delegating classifiers as a generalised separate-and-conquer method. Section 4 starts the experimental evaluation with an analysis of the behaviour for a delegation tandem of just two classifiers. Section 5 discusses the round rebound technique, which bounces part of the delegated examples back to the first classifier, an iterated scenario, and the general algorithm. The approach is extensively evaluated for several configurations and compared to classical ensemble methods. Section 6 discusses the overall results. Section 7 closes the paper with a summary and future work.

2. Cautious Classifiers and Delegation

In many application areas, a classifier that abstains from making a prediction when it is not sure of being able to make the right decision is preferable over a greedy classifier that always makes a classification. While under 0-1 loss accuracy and error are complementary, this is not the case when the classifier has the ability to abstain. We define a *cautious classifier* as a classifier that gives predictions for the subset of inputs for which it is more confident (that may still be right or wrong) but abstains for the rest of its inputs. In other words, a cautious classifier is a partial function.

Any soft classifier, that is, a classifier that estimates the class probabilities or the reliability of each prediction, can be converted to a cautious classifier. We use the following definitions. For a classifier f we consider the associated functions $f_{CLASS}(e)$, $f_{CONF}(e)$, and $f_{PROB_c}(e)$ (for each class c from a total of C classes). The function $f_{CLASS}(e)$ returns the class assigned by classifier f to example e, the function $f_{CONF}(e)$ returns the *confidence* (i.e., an estimate of the reliability) of the prediction given by classifier f to example e, and $f_{PROB_c}(e)$ returns the probability of class c for example e. Unless stated otherwise, we assume that $f_{CLASS}(e) = argmax_c\ f_{PROB_c}(e)$ and $f_{CONF}(e) = max_c\{f_{PROB_c}(e)\}$.

Given these definitions, a cautious classifier f can be obtained from a soft classifier using a confidence threshold.

Decision Rule for a Cautious Classifier with threshold τ:
IF $f_{CONF}(e) > \tau$ THEN PREDICT $f_{CLASS}(e)$
ELSE ABSTAIN

A soft classifier will be converted into a good cautious classifier if the reliabilities are well estimated, as achieved by, for instance, a good class probability estimator, or, for binary problems, a good ranker. Note that for two-class problems, a cautious classifier gives predictions for the reliable positives and the reliable negatives, abstaining for the rest. This represents an *abstention window* (see Figure 1).

The notion of a cautious classifier is an interesting concept in itself, but in this paper we are concerned with the design of good complete classifiers. It is the idea of completing these cautious classifiers that leads to the concept of *delegation*. If a cautious classifier $f^{(1)}$ decides that it is not

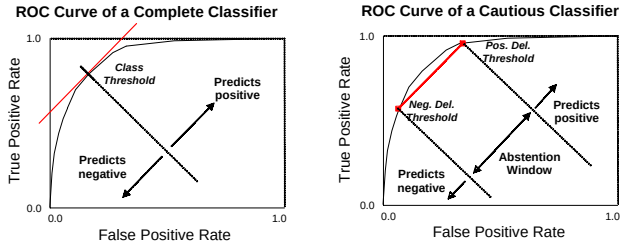

Figure 1. Left: a ROC curve of a complete classifier with a single threshold separating positives from negatives. Right: a ROC curve with two thresholds, representing an abstention window.

competent to classify an example with enough confidence, but wants to complete the work, then it can delegate the example to another classifier. If we have this second classifier, denoted by $f^{(2)}$, and a confidence threshold τ, then the *delegating decision rule* is as follows.

Decision Rule for a Delegating Classifier with threshold τ:
IF $f^{(1)}_{CONF}(e) > \tau$ THEN PREDICT $f^{(1)}_{CLASS}(e)$
ELSE PREDICT $f^{(2)}_{CLASS}(e)$

A key issue is to establish a good way for obtaining classifier $f^{(2)}$. A natural way to obtain classifier $f^{(2)}$ is to train it only on the training examples for which $f^{(1)}$ has low confidence. In this way, the second classifier will be specialised for these examples. More formally, if we have a training set Tr, a soft classifier f and a confidence threshold τ, then we just divide this set into two data sets $Tr^{>}_f = \{e \in Tr : f_{CONF}(e) > \tau\}$ and $Tr^{\leq}_f = Tr - Tr^{>}_f$. Informally, we will refer to $Tr^{>}_f$ as the "retained" or "high-confidence" examples and $Tr^{\leq}_f$ as the "delegated" or "low-confidence" examples.

The same threshold is used for training and for prediction (the delegating decision rule). The question arises how to determine this threshold. One approach we consider in this paper is that a classifier retains a fixed percentage of the examples. For instance, we may stipulate that the first classifier should retain 60% of the most highly ranked examples, delegating the rest to the second classifier. More formally, given a fraction ρ, a classifier f and a training set Tr, we can obtain the threshold τ in the following way:

$$\tau = max\{t : |\{e \in Tr : f_{CONF}(e) > t\}| \geq \rho \times |Tr|\}$$

That is, τ is the greatest threshold such that at least a proportion ρ of the training examples have higher confidence. It does not ensure a proportion of exactly ρ, because there may be many examples with the same confidence, but it returns an upper approximation of this proportion. The method is called *Global Absolute Percentage*.

We also introduce an alternative decision rule for delegating classifiers, in order to handle imbalanced data sets. In this case, we have a different threshold τ_c for each class c.

Decision Rule for a Delegating Classifier with stratified thresholds $\tau_1, \tau_2, ..., \tau_C$:
IF $f^{(1)}_{CONF}(e) > \tau_c$ THEN PREDICT $f^{(1)}_{CLASS}(e)$
ELSE PREDICT $f^{(2)}_{CLASS}(e)$
where $c = f^{(1)}_{CLASS}(e)$

If we denote by Tr_c the examples in Tr of class c, we can obtain each threshold as follows:

$$\tau_c = max\{t : |\{e \in Tr_c : f_{PROB_c}(e) > t\}| \geq \rho \times |Tr_c|\}$$

The retained examples in this case are: $Tr^{>}_f = \{e \in Tr : c = f_{CLASS}(e) \wedge f_{CONF}(e) > \tau_c\}$. This method is called *Stratified Absolute Percentage*.

The above outlines well-defined techniques for training and testing delegating classifiers. In particular, the predicted class and associated confidence for each test instance is obtained from one of the base classifiers. Computing the accuracy of the overall delegating accuracy is thus straightforward. In addition, we use the Area Under the ROC Curve (AUC). AUC is not only suitable in contexts with variable misclassification costs or class distributions (Provost & Fawcett, 2001), it also provides an estimate of a soft classifier's ranking performance, as it is equivalent to the Whitney-Mann-Wilcoxon sum of ranks test. For multi-class problems we employ the approximation presented in (Hand & Till, 2001), which averages the AUC of all 1-vs-1 ROC curves. It may happen that the first classifier retains all the examples of one class and hence the second classifier has fewer classes than the first one. To calculate AUC in this case, we simply assume that the second classifier assigns 0 probability for non-delegated classes.

3. Delegation as a Generalisation of Separate-and-Conquer Methods

In this section we put delegation in a general machine learning context and discuss its relation with separate-and-conquer methods such as the sequential covering algorithm for learning sets of rules. Clearly, the behaviour of a tandem of delegating classifiers depends on the machine learning method used for the base classifiers. Here, we analyse the use of fence-and-fill methods (i.e., methods that partition the instance space into regions). In that case, the first classifier will retain the bigger, relatively purer areas of the instance space and delegate the smaller, less pure areas to the second classifier. Interestingly, since the $Tr^{>}_f$ examples are removed for the second classifier, the removed areas can "clear the space" so that the remaining small areas may be joined into bigger areas for the second classifier. This will be particularly useful in the presence of real patterns that were obscured by the patterns extracted by the first classifier. Figure 2 illustrates the process with two decision tree learners. Here, we assume some form of smoothing (e.g., Laplace correction), so that the confidences associated with

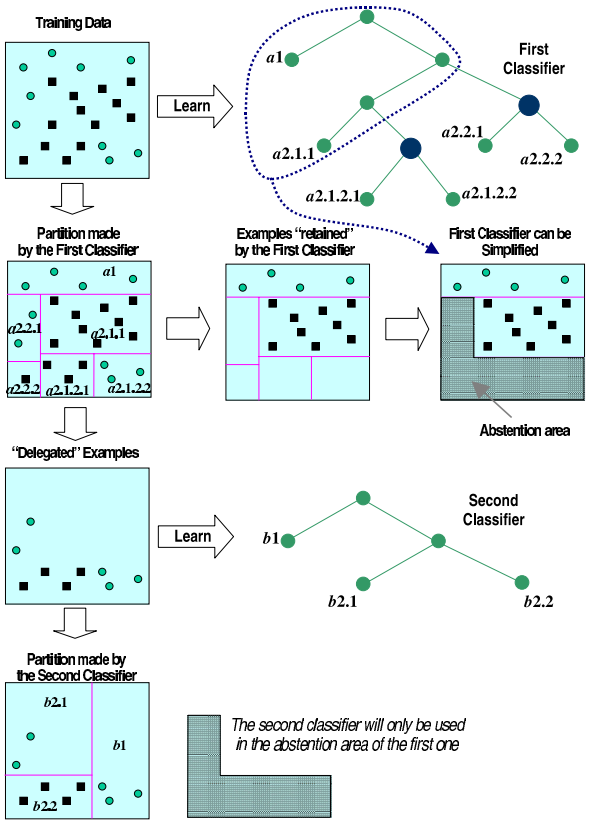

Figure 2. The delegation process using decision trees.

the smallest leaves drop below the threshold and thus these areas are delegated to the second classifier.

In this particular example we can simplify the first classifier, replacing the subtrees that lead to only low-confidence leaves with "delegation nodes" that lead into the second tree. On the meta-level, the entire delegating classifier is again a decision tree. In fact, we can change the tree into a graph by combining all delegation nodes into a single node, so that we only represent the second tree once (Figure 3). This process, which we refer to as *grafting*, is a very natural way to learn decision graphs.

There is an analogy between delegating classifiers and separate-and-conquer rule learners such as CN2 (Clark & Niblett, 1989) and FOIL (Quinlan, 1990). With the separate-and-conquer approach, a single rule is learned in each iteration and the area it covers is removed from the training set. Delegating classifiers perform a similar strategy but work at the meta-level, learning a full classifier in each step. This entails several important differences. First, the area covered in each iteration contains examples of multiple classes. Secondly, the area can have a much more sophisticated shape than the axis-parallel hyper-rectangles covered by a single rule. Thirdly, the reliability estimates over the area are in general not constant. Consequently, delegation, especially under the iterative scenario explored

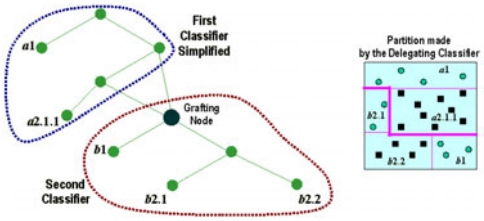

Figure 3. Grafting delegating classifiers.

in Section 5, can be seen as a powerful generalisation of separate-and-conquer methods. This connection highlights that the potential applications and kinds of problems where delegation can work well will be those where separate-and-conquer methods are applicable (see e.g. (Fürnkranz, 1999)). The use of a complete classifier in each iteration makes delegation a much more powerful method.

In the next sections we experimentally validate the method in three different scenarios: the two-stage scenario considered above, a round-rebound scenario where the second classifier can delegate examples back to the first, and an iterative scenario with a variable number of base classifiers.

4. Two-stage Scenario and Experiments

The two-stage scenario was introduced in Section 2. In this section we experimentally analyse the behaviour of this type of delegating classifiers. Table 1 lists the 22 datasets from the UCI dataset repository (Blake & Merz, 1998) we used. We applied 20×5-fold cross-validation instead of a more usual 10×10-fold cross-validation because for computing the AUC we need examples of every class and some datasets have small minority classes. For a set of datasets we use arithmetic means unless stated otherwise.

In the experiments, we employ Probability Estimation Trees (PETs), which estimate the probability of class membership for every class. A trained decision tree can be easily adapted to be a probability estimator by using the absolute class frequencies of each leaf in the tree. However, the probability estimates obtained by PETs can be poor in comparison with other probability estimators. In order to obtain better estimates, we employ the improvements presented in (Ferri et al., 2003), using the SMILES system. These improvements involve a new splitting criterion and a new smoothing method (mbranch smooth), both devoted to improve the AUC of the learned trees. Pruning is not enabled, since it is not beneficial for increasing the estimated probabilities (Domingos & Provost, 2003).

Table 2 shows a mean increase of about one point in accuracy with respect to a single tree, obtained by just retraining a second tree with around 50% of the initial dataset. According to t-tests there are 8 significant wins and 1 loss in accuracy, and 7 wins and 1 loss in AUC. It can be seen (see columns $AcT^{\geq}1$ and $AcT \geq 2$) that the second classifier

Table 1. Datasets used in the experiments: number of examples, number of classes, number of nominal and numeric attributes.

#	DATASETS	SIZE	C	NOM.	NUM.
1	BALANCE SCALE	625	3	0	4
2	BREAST CANCER WDBC	699	2	0	9
3	BREAST CANCER WIS.	569	2	1	30
4	CONTRA. METHOD (CMC)	1473	3	7	2
5	DERMATOLOGY	366	6	33	1
6	HAYES-ROTH	160	3	4	0
7	HEART DISEASE	920	5	8	5
8	HOUSE CONG. VOTING	435	2	16	0
9	IRIS PLAN	150	3	0	4
10	MONK'S1	566	2	6	0
11	MONK'S2	601	2	6	0
12	MONK'S3	554	2	6	0
13	NEW THYROID	215	3	0	5
14	SEGMENTATION	2310	7	0	19
15	TEACHING A. EVAL.	151	3	2	3
16	TIC-TAC-TOE	958	2	8	0
17	WINE RECOGNITION	178	3	0	13
18	SPECT	267	2	22	0
19	CARS	1728	4	6	0
20	OPTDIGITS	5620	10	0	64
21	SPAM	4601	2	0	57
22	THYROID SICK EU	3163	2	19	6

has higher accuracy than the first one precisely on the low-confidence examples delegated by the first classifier. This is the key to the overall improvement in accuracy achieved by the delegating classifier.

To analyse the effect of the first classifier's ranking performance, we considered four configurations for the first classifier: pruning and no smoothing, no pruning and no smoothing, no pruning and Laplace smoothing as used in (Domingos & Provost, 2003), and no pruning and m-branch improvements presented in (Ferri et al., 2003). Table 3 gives mean results (over 22 datasets) of these four configurations for the global absolute percentage method (50%). As we can see, it is the ranking performance rather than the accuracy of the first classifier which is crucial for accuracy and particularly AUC of the delegating classifier.

Table 4 demonstrates that accuracy is not very sensitive to the percentage of examples retained, although it seems that around 50% is a good compromise. With lower percentages most of the work is left to the second classifier, which is then very similar to the first one and not specialised sufficiently to improve the results. A high retention percentage lowers the influence of the second classifier and may also lead to overfitting. The method appears to be robust in the sense that, with several configurations, mean accuracy is never worse than for a single classifier. The AUC of the stratified method is worse than for the global method. This may be due to the fact that the non-stratified method usually levels the proportion of classes for the second sample since examples of the majority class are usually better ranked than the examples of the minority class. Hence, the second classifier can pay more attention to minority classes.

5. Round Rebound and Iterative Scenarios

In this section we consider variants of the two-stage delegation scenario. The first scenario we consider is that the second classifier delegates its low-confidence examples back to the first; we call this *round rebound*. The rationale is that if an example is rejected by both the first and the second classifier, it would be best classified by the first rather than the second because the first classifier is more general and potentially less overfitting. This leads to the following decision rule.

Table 2. Experimental results for two-stage delegating classifiers: accuracy of the first tree on all test examples (Ac1st), accuracy of the whole delegating classifier with $\rho = 50\%$ using global absolute percentage (AcDel), AUC of the first tree (AUC1st) and the whole delegating classifier (AUCDel), percentage of examples retained by the first tree (%Retd), accuracy of first ($\text{AcT}^{\geq}1$) and second ($\text{AcT}^{\geq}2$) tree on the delegated test set. Symbols $\circ$, $\bullet$ show a statistically significant improvement or degradation in accuracy and AUC according to the t-test with 99% significance of the whole delegating classifier wrt. the first tree.

#	AC1ST	ACDEL	AUC1ST	AUCDEL	%RETD	$\text{ACT}^{\geq}1$	$\text{ACT}^{\geq}2$
1	78.40	78.94	82.26	84.73$\circ$	51.48	61.88	62.92
2	93.12	93.10	97.37	87.17	65.60	83.19	83.37
3	93.55	93.66	97.89	97.92	54.07	87.14	87.37
4	46.80	46.60	69.64	66.51$\bullet$	50.08	33.73	33.30
5	90.30	92.52$\circ$	98.87	99.11	52.93	82.24	86.88
6	71.71	77.35$\circ$	91.44	92.95	52.14	49.15	60.25
7	49.10	48.94	68.27	67.56$\circ$	50.19	26.59	26.32
8	94.26	93.72	98.33	98.16	59.91	86.13	84.45
9	94.14	93.55	98.39	98.22	64.88	90.49	87.82
10	96.97	97.59	98.08	98.40	51.14	95.24	96.31
11	76.60	80.38$\circ$	78.17	84.25$\circ$	50.29	68.43	75.10
12	97.52	96.97$\bullet$	98.90	98.77	52.46	96.14	95.06
13	93.36	93.74	98.20	97.96	54.34	88.26	88.89
14	96.14	96.13	99.71	99.71	54.42	92.23	92.17
15	60.70	60.57	77.18	75.04	51.02	56.70	56.46
16	76.73	78.85$\circ$	85.77	88.32$\circ$	50.50	61.83	65.84
17	92.49	92.57	97.58	97.62	57.41	88.59	88.69
18	76.60	78.32	77.64	77.36	51.64	62.23	65.36
19	89.87	94.72$\circ$	95.79	98.94$\circ$	55.42	76.96	87.99
20	90.54	91.10$\circ$	99.49	99.50	51.78	81.86	83.01
21	92.38	93.05$\circ$	97.29	97.54$\circ$	51.42	85.82	87.22
22	90.67	91.72$\circ$	91.00	91.72$\circ$	51.45	81.20	83.35
MEAN	83.72	84.73	90.78	91.31	53.84	74.37	76.28
GEOM	82.14	83.14	90.16	90.64	53.68	71.09	73.07

Table 3. Influence of good probability estimation: comparison of accuracy and AUC of the first decision tree (Single) with accuracy and AUC of the overall delegating classifier (Del50%).

	Pr NoSmooth	NoPr NoSmooth	NoPr Laplace	NoPr Mbranch
Single Acc	83.88	83.72	83.72	83.72
Single AUC	86.46	87.16	90.18	90.78
Del50% Acc	84.01	83.81	84.77	84.73
Del50% AUC	85.93	87.16	90.89	91.31

Decision Rule for a Round Rebound Delegating Classifier with thresholds $\tau^{(1)}$, $\tau^{(2)}$:
IF $f^{(1)}_{CONF}(e) > \tau^{(1)}$ THEN PREDICT $f^{(1)}_{CLASS}(e)$
ELSE IF $f^{(2)}_{CONF}(e) > \tau^{(2)}$ THEN PREDICT $f^{(2)}_{CLASS}(e)$
ELSE PREDICT $f^{(1)}_{CLASS}(e)$

For the second threshold $\tau^{(2)}$, we can again select a percentage of the complete training set (Absolute Percentage), or we can select a percentage of the examples delegated by the first classifier (*Relative Percentage*):

$$\tau^{(2)} = max\{t : |\{e \in Tr^{\leq}_{f^{(1)}} : f^{(2)}_{CONF}(e) > t\}| > \rho \times |Tr^{\leq}_{f^{(1)}}|\}$$

The difference is that, given a percentage of 40%, using the absolute percentage method both classifiers retain ap-

Table 4. Influence of delegation percentage and methods for setting the thresholds: global absolute percentage (GAP) and stratified absolute percentage (SAP).

	None	20%	33%	45%	50%	55%	67%	80%
GAP Acc	83.72	84.23	84.13	84.67	84.73	84.72	84.61	84.60
GAP AUC	90.78	91.02	91.14	91.27	91.31	91.24	91.04	90.92
SAP Acc	84.73	84.29	84.42	84.37	84.34	84.32	84.48	84.32
SAP AUC	91.31	90.79	90.79	90.65	90.61	90.47	90.26	89.85

proximately 40% of all examples, while with the relative percentage method, from the 60% delegated to the second classifier, 24% would be retained.

We now have four different methods of determining the thresholds for the first and second classifiers: global absolute percentage, stratified absolute percentage, global relative percentage and stratified relative percentage. Table 5 shows the mean accuracy and AUC results (over the 22 datasets) of the round rebound scenario with the four different methods and several retention percentages. Comparing these results with previous results, the mean accuracy is slightly better (from 84.73 to 85.04 in the best case, with 10 significant wins and 0 losses) and for AUC it almost stays constant (from 91.31 to 91.33, with the same 7 wins and 1 loss). While this is a small improvement, it should be noted that it is achieved without additional learning cost.

The second variation we consider is an iteration of several delegating classifiers with the following decision rule.

Decision Rule for a Delegating Classifier
with thresholds $\tau^{(1)}$, $\tau^{(2)}$, ..., $\tau^{(n)}$:
IF $f^{(1)}_{CONF}(e) > \tau^{(1)}$ THEN PREDICT $f^{(1)}_{CLASS}(e)$
ELSE IF $f^{(2)}_{CONF}(e) > \tau^{(2)}$ THEN PREDICT $f^{(2)}_{CLASS}(e)$
...
ELSE IF $f^{(n-1)}_{CONF}(e) > \tau^{(n-1)}$ THEN PREDICT $f^{(n-1)}_{CLASS}(e)$
ELSE PREDICT $f^{(n)}_{CLASS}(e)$

It is again possible to include the round rebound; however, if the number of examples in the last iteration are few the effect of rebound will be negligible, so we will not use it in the iterative scenario.

We thus arrive at the following general algorithm for learning a delegating classifier. The algorithm has several parameters, some of them already explained (the method and percentage for obtaining the threshold), but also the maximum number of iterations, which is especially important when using the relative methods of obtaining the threshold.

Procedure Learn_Delegating_Classifier(Train)
$Tr^{(1)} \leftarrow Train$, $i \leftarrow 0$
do
$i \leftarrow i+1$
LEARN $f^{(i)}$ with $Tr^{(i)}$
Obtain $\tau^{(i)}$ // with any of the 4 possible methods
$Tr^{>}_{f^{(i)}} := \{e \in Tr^{(i)} : f^{(i)}_{CONF}(e) > \tau^{(i)}\}$
$Tr^{\leq}_{f^{(i)}} := \{e \in Tr^{(i)} : f^{(i)}_{CONF}(e) \leq \tau^{(i)}\}$
$Tr^{(i+1)} \leftarrow Tr^{\leq}_{f^{(i)}}$
until $Tr^{>}_{f^{(i)}} = \emptyset$ or $i > maxIterations$

Table 6 shows the mean results (accuracy, AUC and number of iterations) for different retention percentages for the absolute methods, and Table 7 shows similar results for the relative methods (stratified not shown since results are worse). These tables demonstrate that the results improve with the number of iterations. This is easy to do and to control with the absolute methods, although since the thresholds are upper approximations, a percentage of 1% in the global absolute does not translate into 100 iterations but only a mean of 31.31. The important thing about the absolute methods, apart from the fact that the results are better, is that there seems to be no saturation point; whereas with the relative method we have to find a good combination of the relative proportion of examples retained and the maximum number of iterations. In the absolute case, just reducing the percentage increments the number of iterations and the results. With respect to the two-stage scenario, now the best improvement in accuracy is considerable (from 83.72 to 85.93) and the improvement in AUC begins to be important (from 90.78 to 91.82).

Finally, we compare delegating (global absolute percentage of 1% and 2%) with two ensemble methods: boosting and bagging. For this last experiment, we use Weka (Witten & Frank, 1999), using J4.8 for all the experiments (Laplace smoothing enabled), with pruning only enabled for boosting. In this case, in order to use the same base algorithm for the comparison, we use an implementation of our delegation algorithm in Weka, without the AUC improvements (no m-branch smoothing and no special splitting criteria). The results of delegation are hence slightly different to those shown in previous tables, but are consistent with them. We again used 20×5-fold cross-validation.

Table 5. Round rebound two-stage scenario results for different retention percentages and methods to determine the thresholds.

		33%	45%	50%	55%	67%
GLOBAL ABSOLUTE	ACC	84.68	85.04	84.97	84.95	84.87
GLOBAL ABSOLUTE	AUC	91.20	91.33	91.33	91.26	91.11
STRATIFIED ABSOLUTE	ACC	84.44	84.55	84.40	84.39	84.53
STRATIFIED ABSOLUTE	AUC	91.05	90.77	90.64	90.51	90.27
GLOBAL RELATIVE	ACC	84.41	84.59	84.63	84.66	84.77
GLOBAL RELATIVE	AUC	91.19	91.16	91.20	91.11	90.96
STRATIFIED RELATIVE	ACC	84.37	84.43	84.44	84.50	84.50
STRATIFIED RELATIVE	AUC	91.06	90.92	90.84	90.71	90.31

Table 6. Results for the iterative scenario with absolute methods and different retention percentages.

	50%	33%	20%	10%	5%	2%	1%
GLOBAL ACC	84.73	85.20	85.33	85.64	85.82	85.85	85.93
GLOBAL AUC	91.31	91.40	91.43	91.61	91.75	91.82	91.82
GLOBAL #IT	2.00	3.16	4.68	7.74	12,50	21.64	31.31
STRAT. ACC	84.34	84.75	85.09	85.30	85.42	85.53	85.58
STRAT. AUC	90.61	90.70	90.74	91.02	91.25	91.44	91.51
STRAT. #IT	2.00	3.06	4.33	6.70	9.83	15.30	18.24

Table 7. Results for the iterative scenario with relative methods, different retention percentages and different number of iterations.

MAXITERATIONS:	2	3	5	10	20	50
GLOBAL 50% ACC	84.73	85.42	85.49	85.48	85.47	85.47
GLOBAL 50% AUC	91.31	91.50	91.51	91.50	91.50	91.50
GLOBAL 50% #IT	2.00	2.99	4.85	7.06	7.11	7.11
GLOBAL 33% ACC	84.13	85.13	85.57	85.66	85.65	85.65
GLOBAL 33% AUC	91.14	91.56	91.71	91.78	91.71	91.71
GLOBAL 33% #IT	2.00	3.00	4.95	8.74	10.52	10.52
GLOBAL 10% ACC	84.08	84.56	84.92	85.50	85.83	85.85
GLOBAL 10% AUC	91.01	91.24	91.29	91.65	91.80	91.79
GLOBAL 10% #IT	2.00	3.00	4.98	9.52	16.83	25.52

Table 8. Comparison between delegation and ensemble methods. Results of J48 from Weka, delegating 2%, bagging and boosting with 10 iterations, delegating 1%, and bagging and boosting with 20 iterations. Significance tests are with respect to J48.

#	J48	DEL2%	BAG10	BOOS10	DEL1%	BAG20	BOO20
1	78.84	81.61 ◦	81.38 ◦	78.82	81.77 ◦	81.61 ◦	77.73 •
2	94.13	94.63	95.93 ◦	96.14 ◦	94.56	96.02 ◦	96.37 ◦
3	93.42	94.19 ◦	95.04 ◦	96.00 ◦	94.17 ◦	95.25 ◦	96.23 ◦
4	49.18	47.69 •	51.39 ◦	50.18 ◦	48.09 •	51.52 ◦	50.21 ◦
5	93.87	93.89	96.12 ◦	96.43 ◦	93.24	96.16 ◦	96.60 ◦
6	73.22	79.63 ◦	75.66 ◦	77.19 ◦	79.53 ◦	76.47 ◦	77.25 ◦
7	51.74	51.41	55.92 ◦	54.41 ◦	51.93	56.09 ◦	54.80 ◦
8	95.34	95.23	96.06 ◦	95.11	94.94	96.28 ◦	95.09
9	94.43	94.47	94.33	94.03	94.03	94.20	94.13
10	93.60	99.97 ◦	99.43 ◦	98.93 ◦	99.97 ◦	99.77 ◦	98.93 ◦
11	61.81	84.28 ◦	65.00 ◦	73.84 ◦	84.73 ◦	65.81 ◦	75.47 ◦
12	98.63	98.79 ◦	98.76 ◦	97.90 •	98.81 ◦	98.82 ◦	97.89 •
13	91.98	92.81 ◦	93.53 ◦	94.53 ◦	92.72	93.58 ◦	94.70 ◦
14	96.55	96.59	97.30 ◦	98.05 ◦	96.55	97.38 ◦	98.31 ◦
15	53.81	54.60	56.31 ◦	59.87 ◦	54.80	57.68 ◦	60.24 ◦
16	78.88	86.59 ◦	82.38 ◦	82.81 ◦	86.76 ◦	82.76 ◦	83.28 ◦
17	92.23	92.56	94.78 ◦	95.67 ◦	92.65	95.20 ◦	96.01 ◦
18	79.20	79.44	82.13 ◦	80.50 ◦	79.46	82.06 ◦	80.88 ◦
19	93.73	96.64 ◦	93.70	95.25 ◦	96.84 ◦	93.86	95.97 ◦
20	90.02	92.82 ◦	95.05 ◦	97.16 ◦	92.89 ◦	95.49 ◦	97.88 ◦
21	92.36	93.46 ◦	94.25 ◦	94.95 ◦	93.48 ◦	94.42 ◦	95.03 ◦
22	97.51	97.09 •	97.71 ◦	97.52	96.99 •	97.74 ◦	97.70 ◦
MEAN	83.84	86.29	86.01	86.60	86.31	86.28	86.85
GEOM	82.10	84.61	84.43	85.09	84.68	84.75	85.35

From the results in Table 8, delegation is not far behind other ensemble methods in terms of accuracy. The important point is that delegation requires much more modest resources, as at each iteration there is fewer data for training. For instance, delegation with global absolute percentage at 2% requires 21.64 iterations on the average and a total of 8933.67 examples handled ($Tr^{(1)} + Tr^{(2)} + .. + Tr^{(n)}$)). with an average dataset size of 1195.41 examples ($Tr^{(1)}$). This means with 20 iterations, the time complexity is only around 8 times higher. Similarly, for delegation at 1% there are 31.31 mean iterations and a total of 11923.40 examples handled. This means that the execution time is only around 10 times the time it takes to generate a single tree.

In Table 9 we show a pairwise comparison of the number of win/losses (t-test 99% significance) between the following algorithms: Delegating (2%), J48, Bagging (10 iterations), and Boosting (10 iterations). Although delegating improves significantly the performance of J48, it does not reach the level of Bagging or Boosting.

Table 9. Number of datasets where the algorithm of the column is significantly better than the algorithm of the row.

	Del 2%	J48	Bag10	Boos10
Del 2%	-	2	12	13
J48	11	-	20	17
Bag10	5	0	-	9
Boos10	7	1	7	-

6. Discussion

The experimental results in the previous sections demonstrate that delegation is a technique with a very good trade-off between the quality of models and the resources needed to obtain them. The efficiency of the method derives from its separate-and-conquer philosophy and the serial multi-classifier topology, resulting in a sub-linear increase of resources with respect to the number of classifiers.

We have also demonstrated a clear improvement in accuracy. A formal analysis is beyond the scope of this paper, particularly since the delegating technique is not a combination technique, hence we cannot justify the improvement from a reduction of variance as with other ensemble methods. Here, we discuss a number of factors that seem to play an important role. First and foremost, since classifiers decide themselves which examples to retain and which to delegate, the quality of the reliability estimation of each delegating classifiers is crucial. This is why we used unpruned decision trees optimised for probability estimation. Secondly, the idea of iteratively removing patterns (from easier to more difficult patterns, see), common to separate-and-conquer methods, is likely to assist in obtaining good accuracy. Thirdly, class distribution is modified by the non-stratified threshold, usually balancing classes as the process evolves, which also intuitively seems beneficial. Fourthly, there is the reduction of training sets as a result of delegation. A smaller training dataset means better specialisation but may also lead to overfitting. However, the results are improved by the iterated scenario, at least if retention percentages are small, showing that overfitting may be compensated by better specialisation or by other factors. Finally, the kind of base classifier is also relevant. Divide-and-conquer methods, such as decision trees, particularly benefit from delegation because there will be several delegating nodes which can be joined into a graft node, as discussed in section 3. We would argue that the good behaviour of the round rebound configuration demonstrates some of these factors. Although the second classifier generally improves classification accuracy on the delegated sample, there is also a small subsample which is better bounced back to the original classifier, thus avoiding overfitting.

Although AUC is also increased, the results are less conclusive. One explanation could be found in the fact that the base probability estimation trees we are using are really specialised for AUC, and hence improving their performance is difficult. Combining the predictions of the delegating classifiers, or selecting the classifiers globally instead of hierarchically could improve the results in AUC. Another reason might be that some classes could have no delegated examples, and this may affect the overall AUC.

The experimental results are inconclusive as to whether delegating works better with large datasets than smaller

ones, but there does seem to be a preference for larger datasets. This and the better efficiency of delegating versus other combination techniques could justify the following *modus operandi*: (1) If comprehensibility and efficiency are crucial, use delegated decision trees with a no-rebound two-stage scenario. (2) If only efficiency is crucial, use the two-stage scenario with round-rebound or the iterated scenario (possibly employing efficient classifiers first and possibly more computation-intensive classifiers for subsequent models). (3) If neither comprehensibility nor efficiency are crucial, try other classical ensemble methods. Finally, regarding robustness, we have seen that for all the schemes discussed in this paper, delegation only degrades performance for at most 2 of the 22 datasets.

7. Conclusions and Future Work

The key idea of a delegating classifier is that it only makes predictions with a minimum level of confidence and delegates the prediction to another classifier otherwise. We have argued that delegation is a fundamental notion in machine learning, and we have analysed the similarities and differences with other general techniques such as separate-and-conquer rule learning and ensemble methods. We have investigated the performance of different configurations of delegating classifiers. In general, the results show that delegating classifiers can significantly improve the accuracy of the predictions, without some of the disadvantages of other multiclassifiers. In particular, we do not combine the predictions of classifiers: each instance is classified by a single classifier. This does not degrade the comprehensibility of the model as ensemble methods do. Secondly, the resulting multi-classifier can often be simplified. For instance, delegating decision trees can be simplified to decision graphs. Finally, our approach is considerably more efficient than classical ensemble methods, because each subsequent classifier is learned using fewer examples than the previous one.

As future work, we will investigate the use of different methods for the base classifiers at each stage. For example, the first classifier can be an efficient classifier such as Naive Bayes, while further down the delegation chain we use more data-intensive methods. We also plan to investigate combination of the predictions of the base classifiers, e.g., by weighting their prediction depending on the confidence and iteration. We would lose comprehensibility and the possibility of simplifying the partial solutions but we would maintain the same efficiency, and we could possibly come even closer to the accuracy of boosting or bagging. Another idea to pursue, when using decision trees, is a safe pre-pruning, which would detect when a node is not leading to leaves with confidence greater than the threshold. This would increase efficiency. We also plan to investigate the applicability of delegation beyond classification. Most of the presented ideas can be applied to regression and clustering, provided each learned model has a good estimated reliability for predictions or cluster membership. Finally, for multi-class problems, we can use the first classifier to determine the most likely classes for an example, and then use several binary classifiers in the second stage. This would be specially interesting for support vector machines.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and pointers to related work. This work has been supported by Generalitat Valenciana.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–283.

Domingos, P., & Provost, F. (2003). Tree induction for probability-based ranking. *Mach. Learn.*, *52*, 199–216.

Ferri, C., Flach, P., & Hernández, J. (2003). Improving the AUC of Probabilistic Estimation Trees. *Proc. of the 14th European Conf. on Machine Learning* (pp. 121–132).

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proc. of the 15th Int. Conf. on Machine Learning (ICML-98)* (pp. 144–151).

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proc. 13th Int. Conf. on Machine Learning* (pp. 148–146). Morgan Kaufmann.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13*, 3–54.

Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning*, *41*, 315–343.

Hand, D., & Till, R. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*, 171–186.

Ortega, J., Koppel, M., & Argamon, S. (2001). Arbitrating among competing classifiers using learned referees. *Knowledge and Information Systems*, *3*, 470–490.

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Mach. Learn.*, *42*, 203–231.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning*, *5*, 239–266.

Seewald, A. K., & Fürnkranz, J. (2001). An evaluation of grading classifiers. *Advances in Intelligent Data Analysis: Proc. of the 4th Int. Conf. (IDA-01)* (pp. 115–124).

Witten, I. H., & Frank, E. (1999). *Data mining: Practical machine learning tools and techniques with java implementations*. Morgan Kaufmann Publishers.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.