Combining Classifiers with Meta Decision Trees

Ljupčo Todorovski, Sašo Džeroski
Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia

Abstract

The paper introduces meta decision trees (MDTs), a novel method for combining multiple classifiers. Instead of giving a prediction, MDT leaves specify which classifier should be used to obtain a prediction. We present an algorithm for learning MDTs based on the C4.5 algorithm for learning ordinary decision trees (ODTs). An extensive experimental evaluation of the new algorithm is performed on twenty-one data sets, combining classifiers generated by five learning algorithms: two algorithms for learning decision trees, a rule learning algorithm, a nearest neighbor algorithm and a naive Bayes algorithm. In terms of performance, stacking with MDTs combines classifiers better than voting and stacking with ODTs. In addition, the MDTs are much more concise than the ODTs and are thus a step towards comprehensible combination of multiple classifiers. MDTs also perform better than several other approaches to stacking.

1 Introduction

The task of constructing ensembles of classifiers [8] can be broken down into two sub-tasks. We first have to *generate* a diverse set of base-level classifiers. Once the base-level classifiers have been generated, the issue of how to *combine* their predictions arises.

Several approaches to *generating* base-level classifiers are possible. One approach is to generate classifiers by applying different learning algorithms (with heterogeneous model representations) to a single data set (see, e.g., Merz [14]). Another possibility is to apply a single learning algorithm with different parameters settings to a single data set. Finally, methods like bagging [5] and boosting [9] generate multiple classifies by applying a single learning algorithm to different versions of a given data set. Two different methods for manipulating the data set are used: random sampling with replacement (also called bootstrap sampling) in bagging and re-weighting of the misclassified training examples in boosting.

Techniques for *combining* the predictions obtained from the multiple base-level classifiers can be clustered in three combining frameworks: voting (used in bagging and boosting), stacked generalization or stacking [22] and cascading [10]. In voting, each base-level classifier gives a vote for its prediction. The prediction receiving the most votes is the final prediction. In stacking, a learning algorithm is used to learn how to combine the predictions of the base-level classifiers. The induced meta-level classifier is then used to obtain the final prediction from the predictions of the base-level classifiers. Cascading is an iterative process of combining classifiers: at each iteration, the training data set is extended with the predictions obtained in the previous iteration.

The work presented here focuses on combining the predictions of base-level classifiers induced by applying different learning algorithms to a single data set. It adopts the stacking framework, where we have to learn how to combine the base-level classifiers. To this end, it introduces the notion of meta decision trees (MDTs), proposes an algorithm for learning them and evaluates MDTs in comparison to other methods for combining classifiers.

Meta decision trees (MDTs) are a novel method for combining multiple classifiers. The difference between meta and ordinary decision trees (ODTs) is that MDT leaves specify which base-level classifier should be used, instead of predicting the class value directly. The attributes used by MDTs are derived from the class probability distributions predicted by the base-level classifiers for a given example. We have developed MLC4.5, a modification of C4.5 [17], for inducing meta decision trees. MDTs and MLC4.5 are described in Section 3.

The performance of MDTs is evaluated on a collection of twenty-one data sets. MDTs are used to combine classifiers generated by five base-level learning algorithms: two tree-learning algorithms C4.5 [17] and LTree [11], the rule-learning algorithm CN2 [7], the k-nearest neighbor (k-NN) algorithm [20] and a modification of the naive Bayes algorithm [12]. In the experiments, we compare the performance of stacking with MDTs to the performance of stacking with ODTs. We also compare MDTs with two voting schemes and two other stacking approaches. Finally, we compare MDTs to boosting and bagging of decision trees as state of the art methods for constructing ensembles of classifiers.

Section 4 reports on the experimental methodology and results. The experimental results are analyzed and discussed in Section 5. The presented work is put in the context of previous work on combining multiple classifiers in Section 6. Section 7 presents conclusions based on the empirical evaluation along with directions for further work.

2 Combining Multiple Classifiers

In this paper, we focus on combining multiple classifiers generated by using different learning algorithms on a single data set. In the first phase, depicted on the left hand side of Figure 1, a set $\mathcal{C} = \{C_1, C_2, \ldots C_N\}$ of N base-level classifiers is generated by applying the learning algorithms $\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_N$ to a single training data set L.

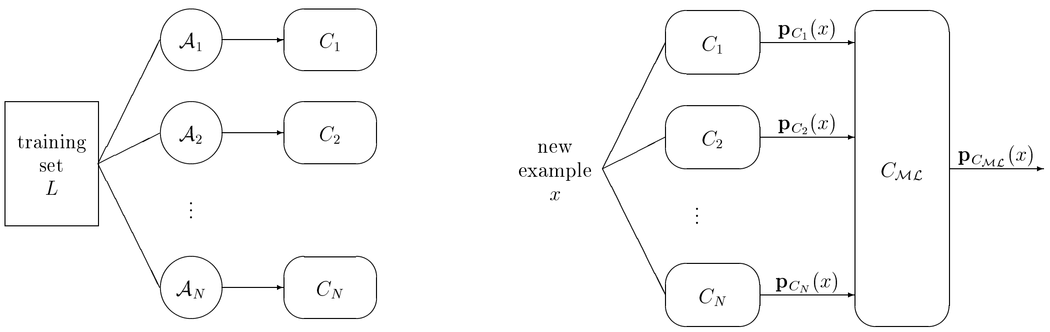

Figure 1: Constructing and using an ensemble of classifiers. Left hand side: generation of the base-level classifiers $\mathcal{C} = \{C_1, C_2, \ldots C_N\}$ by applying N different learning algorithms to a single training data set L. Right hand side: classification of a new example x using the base-level classifiers in $\mathcal{C}$ and the combining method $C_{\mathcal{ML}}$.

We assume that each base-level classifier from $\mathcal{C}$ predicts a probability distribution over the possible class values. Thus, the prediction of the base-level classifier C when applied to example x is a probability distribution vector:

$$\mathbf{p}_C(x) = (p_C(c_1|x), p_C(c_2|x), \ldots p_C(c_k|x)),$$

where $\{c_1, c_2, \ldots c_k\}$ is a set of possible class values and $p_C(c_i|x)$ denotes the probability that example x belongs to class c_i as estimated (and predicted) by classifier C. The class c_j with the highest class probability $p_C(c_j|x)$ is predicted by classifier C.

The classification of a new example x at the meta-level is depicted on the right hand side of Figure 1. First, the N predictions $\{\mathbf{p}_{C_1}(x), \mathbf{p}_{C_2}(x), \ldots \mathbf{p}_{C_N}(x)\}$ of the base-level classifiers in $\mathcal{C}$ on x are generated. The obtained predictions are then combined using the combining method $C_{\mathcal{ML}}$. Different combining methods are used in different combining frameworks. In the following subsections, the combining frameworks of voting and stacking are presented.

2.1 Voting

In the voting framework for combining classifiers, the predictions of the base-level classifiers are combined according to a static voting scheme, which does not change with training data set L. The voting scheme remains the same for all different training sets and sets of learning algorithms (or base-level classifiers).

The simplest voting scheme is the plurality vote. According to this voting scheme, each base-level classifier casts a vote for its prediction. The example is classified in the class that collects the most votes.

There is a refinement of the plurality vote algorithm for the case where class probability distributions are predicted by the base-level classifiers [8]. Let $\mathbf{p}_C(x)$ be the class probability distribution predicted by the base-level classifier C on example x. The probability distribution vectors returned by the base-level classifiers can be summed to obtain the class probability distribution of the meta-level voting classifier:

$$\mathbf{p}_{C_{\mathcal{ML}}}(x) = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \mathbf{p}_C(x)$$

The predicted class for x is the class c_j with the highest class probability $p_{C_{\mathcal{ML}}}(c_j|x)$.

2.2 Stacking

In contrast to voting, where $C_{\mathcal{ML}}$ is static, stacking induces a combiner method $C_{\mathcal{ML}}$ from the meta-level training set, based on L, in addition to the base-level classifiers. $C_{\mathcal{ML}}$ is induced by using a learning algorithm at the meta-level: the meta-level examples are constructed from the predictions of the base-level classifiers on the examples in L and the correct classifications of the latter. As the combiner is intended to combine the predictions of the base-level classifiers (induced on the entire training set L) for unseen examples, special care has to be taken when constructing the meta-level data set. To this end, the cross-validation procedure presented in Table 1 is applied.

Table 1: The algorithm for building the meta-level data set used to induce the combiner $C_{\mathcal{ML}}$ in the stacking framework.

```
function build_combiner(L, {A_1, A_2, ... A_N}, m, A_ML)
    {L_1, L_2, ... L_m} = stratified_partition(L, m)
    L_ML = {}
    for i = 1 to m do
        for j = 1 to N do
            Let C_{j,i} be the classifier obtained by applying A_j to L \ L_i
            Let CV_{j,i} be class values predicted by C_{j,i} on examples in L_i
            Let CD_{j,i} be class distributions predicted by C_{j,i} on examples in L_i
            MLA_{j,i} = meta_level_attributes(CV_{j,i}, CD_{j,i}, L_i)
        endfor
        L_ML = L_ML ∪ join_{j=1}^{N} MLA_{j,i}
    endfor
    Apply A_ML to L_ML in order to induce the combiner C_ML
    return C_ML
endfunction
```

First, the training set L is partitioned into m disjoint sets $L_1, L_2, \ldots L_m$ of (almost) equal size. The partitioning is stratified in the sense that each set L_i roughly preserves the class probability distribution in L. In order to obtain the base-level predictions on unseen examples, the learning algorithm $\mathcal{A}_j$ is used to train base-level classifier $C_{j,i}$ on the training set $L \setminus L_i$. The trained classifier $C_{j,i}$ is then used to obtain the predictions for examples in L_i. The predictions of the base-level classifiers include the predicted class values $CV_{j,i}$ as well as the class probability distributions $CD_{j,i}$. These are then used to calculate the meta-level attributes for the examples in L_i.

The meta-level attributes are calculated for all N learning algorithms and joined together into set of meta-level examples. This set is one of the m parts of the meta-level data set $L_{\mathcal{ML}}$. Repeating this procedure m times (once for each set L_i), we obtain the whole meta-level data set. Finally, the learning algorithm $\mathcal{A}_{\mathcal{ML}}$ is applied to it in order to induce the combiner $C_{\mathcal{ML}}$.

The framework for combining multiple classifiers used in this paper is based on the combiner methodology described in [6] and the stacking framework of [22]. In these two approaches, only the class values predicted by the base-level classifiers are used as meta-level attributes. Therefore, the `meta_level_attributes` procedure used in these frameworks is trivial: it returns only the class values predicted by the base-level classifiers. In our approach, we use the class probability distributions predicted by the base-level classifiers in addition to the predicted class values for calculating the set of meta-level attributes. The meta-level attributes used in our study are discussed in detail in Section 3.

3 Meta Decision Trees

In this section, we first introduce meta decision trees (MDTs). We then discuss the possible sets of meta-level attributes used to induce MDTs. Finally, we present an algorithm for inducing MDTs, named MLC4.5.

3.1 What are Meta Decision Trees

The structure of a meta decision tree is identical to the structure of an ordinary decision tree. A decision (inner) node specifies a test to be carried out on a single attribute value and each outcome of the test has its own branch leading to the appropriate subtree. In a leaf node, a MDT predicts which classifier is to be used for classification of an example, instead of predicting the class value of the example directly (as an ODT would do).

The difference between ordinary and meta decision trees is illustrated with the example presented in Tables 2 and 3. First, the predictions of the base-level classifiers (Table 2a) are obtained on the given data set. These include predicted class probability distributions as well as class values themselves. In the meta-level data set M (Table 2b), the meta-level attributes C_1 and C_2 are the class value predictions of two base-level classifiers C_1 and C_2 for a given example. The two additional meta-level attributes $Conf_1$ and $Conf_2$

measure the confidence of the predictions of C_1 and C_2 for a given example. The highest class probability, predicted by a base-level classifier, is used as a measure of its prediction confidence.

Table 2: Building the meta-level data set. a) Predictions of the base-level classifiers. b) The meta-level data set M.

a) Predictions of the base-level classifiers

	Classifier C_1			Classifier C_2		
Base-level attributes (x)	$p_{C_1}(0\|x)$	$p_{C_1}(1\|x)$	pred.	$p_{C_2}(0\|x)$	$p_{C_2}(1\|x)$	pred.
...	0.875	0.125	0	0.875	0.125	0
...	0.875	0.125	0	0.25	0.75	1
...	0.125	0.875	1	0.75	0.25	0
...	0.125	0.875	1	0.125	0.875	1
...	0.625	0.375	0	0.75	0.25	0
...	0.625	0.375	0	0.125	0.875	1
...	0.375	0.625	1	0.75	0.25	0
...	0.375	0.625	1	0.125	0.875	1

b) Meta-level data set M

$Conf_1$	C_1	$Conf_2$	C_2	true class
0.875	0	0.875	0	0
0.875	0	0.75	1	0
0.875	1	0.75	0	1
0.875	1	0.875	1	1
0.625	0	0.75	0	0
0.625	0	0.875	1	1
0.625	1	0.75	0	0
0.625	1	0.875	1	1

The meta decision tree induced using the meta-level data set M is given in Table 3a). The MDT is interpreted as follows: if the confidence $Conf_1$ of the base-level classifier C_1 is high, then C_1 should be used for classifying the example, otherwise the base-level classifier C_2 should be used. The ordinary decision tree induced using the same meta-level data set M (given in Table 3b) is much less comprehensible, despite the fact that it reflects the same rule for choosing among the base-level predictions. Note that both the MDT and the ODT need the predictions of the base-level classifiers in order to make their own predictions.

Table 3: The difference between ordinary and meta decision trees. a) The meta decision tree induced from the meta-level data set M (by MLC4.5). b) The ODT induced from the same meta-level data set M (by C4.5). c) The MDT written as a logic program.

a) The MDT induced from M

```
Conf1 <= 0.625:  C2
Conf1 > 0.625:  C1
```

b) The ODT induced from M

```
C1 = 0:
|    Conf1 > 0.625
|    Conf1 <= 0.625:
|    |    C2 = 0:  0
|    |    C2 = 1:  1
C1 = 1:
|    Conf1 > 0.625:  1
|    Conf1 <= 0.625:
|    |    C2 = 0:  0
|    |    C2 = 1:  1
```

c) The MDT written as a logic program

combine($Conf_1$, C_1, $Conf_2$, C_2 , C) :- $Conf_1 \leq 0.625$, $C = C_2$.
combine($Conf_1$, C_1, $Conf_2$, C_2 , C) :- $Conf_1 > 0.625$, $C = C_1$.

The comprehensibility of the MDT from Table 3a) is entirely due to the extended expressiveness of the MDT leaves. Both the MDT and the ODT in Table 3a) and b) are induced from the propositional data set M. While the ODT induced from M is purely propositional, the MDT is not. A (first order) logic program equivalent to the MDT is presented in Table 3c). The predicate combine($Conf_1$, C_1, $Conf_2$, C_2, C) is used to combine the predictions of the base-level classifiers C_1 and C_2 into class C according to the values of the attributes (variables) $Conf_1$ and $Conf_2$. Each clause of the program corresponds to one leaf node of the MDT and includes a non-propositional class value assignment ($C = C_2$ in the first and $C = C_1$ in the second clause). In the propositional framework, the only possible assignments are $C = 0$ and $C = 1$, one for each class value.

There is another way of interpreting meta decision trees. A meta decision tree selects an appropriate classifier for a given example in the domain. Consider the subset of examples falling in one leaf of the MDT. It identifies a subset of the data where one of the base-level classifiers performs better than the others. Thus, the MDT identify subsets that are *relative* areas of expertise of the base-level classifiers. An area of expertise of a base-level classifier is relative in the sense that its predictive performance in that area is better as compared to the performances of the other base-level classifiers. This is different from an area of expertise of an individual base-level classifier [15], which is a subset of the data where the predictions of a single base-level classifier are correct.

Note that in the process of inducing meta decision trees two types of attributes are used. *Ordinary* attributes are used in the decision (inner) nodes of the MDT (e.g., attributes $Conf_1$ and $Conf_2$ in the example meta-level data set M). The role of these attributes is identical to the role of attributes used for inducing ordinary decision trees. *Class* attributes (e.g., C_1 and C_2 in M), on the other hand, are used in the leaf nodes only. Each base-level classifier has its class attribute: the values of this attribute are the predictions of the base-level classifier. Thus, the class attribute assigned to the leaf node of the MDT decides which base-level classifier will be used for prediction. When inducing ODTs for combining classifiers, the class attributes are used in the same way as ordinary attributes.

The partitioning of the data set into relative areas of expertise is based on the values of the ordinary meta-level attributes used to induce MDTs. In existing studies about areas of expertise of individual classifiers [15], the original base-level attributes from the domain at hand are used. We use a different set of ordinary attributes for inducing MDTs. These are properties of the class probability distributions predicted by the base-level classifiers and reflect the certainty and confidence of the predictions. However, the original base-level attributes can be also used to induce MDTs. Details about each of the two sets of meta-level attributes are given in the following subsection.

3.2 Meta-Level Attributes

As meta-level attributes, we calculate the properties of the class probability (CDP) distributions predicted by the base-level classifiers that reflect the certainty and confidence of the predictions.

First, $\texttt{maxprob}(x, C)$ is the highest class probability (i.e. the probability of the predicted class) predicted by the base-level classifier C for example x:

$$\texttt{maxprob}(x, C) = \max_{i=1}^{k} p_C(c_i|x).$$

Next, $\texttt{entropy}(x, C)$ is the entropy of the class probability distribution predicted by the classifier C for example x:

$$\texttt{entropy}(x, C) = -\sum_{i=1}^{k} p_C(c_i|x) \cdot \log_2 p_C(c_i|x).$$

Finally, $\texttt{weight}(x, C)$ is the fraction of the training examples used by the classifier C to estimate the class distribution for example x. For decision trees, it is the weight of the examples in the leaf node used to classify the example. For rules, it is the weight of the examples covered by the rule(s) which has been used to classify the example. This property has not been used for the nearest neighbor and naive Bayes classifiers, as it does not apply to them in a straightforward fashion.

The entropy and the maximum probability of a probability distribution reflect the certainty of the classifier in the predicted class value. If the probability distribution returned is highly spread, the maximum probability will be low and the entropy will be high, indicating that the classifier is not very certain in its prediction of the class value. On the other hand, if the probability distribution returned is highly focused, the maximum probability is high and the entropy low, thus indicating that the classifier is certain in the predicted class value. The weight quantifies how reliable is the predicted class probability distribution. Intuitively, the weight corresponds to the number of training examples used to estimate the probability distribution: the higher the weight, the more reliable the estimate.

An example MDT, induced on the image domain from the UCI repository [3], is given in Table 4. The leaf denoted by an asterisk `(*)` specifies that the C4.5 classifier is to be used to classify an example, if (1) the maximum probability in the class probability distribution predicted by k-NN is smaller than 0.77; (2) the fraction of the examples in the leaf of the C4.5 tree used for prediction is larger than 0.4% of all the examples in the training set; and (3) the entropy of the class distribution predicted by C4.5 is less then 0.14. In sum, if the k-NN classifier is not very confident in its prediction (1) and the C4.5 classifier is very

Table 4: A meta decision tree induced on the image domain using class distribution properties as ordinary attributes.

```
knn_maxprob > 0.77147:  KNN
knn_maxprob <= 0.77147:
|      c45_weight <= 0.00385:  KNN
|      c45_weight > 0.00385:
|      |      c45_entropy <= 0.14144:  C4.5 (*)
|      |      c45_entropy > 0.14144:  LTREE
```

confident in its prediction (3 and 2), the leaf recommends using the C4.5 prediction; this is consistent with common-sense knowledge in the domain of classifier combination.

Another set of ordinary attributes used for inducing meta decision trees is the set of original domain (base-level) attributes (BLA). In this case, the relative areas of expertise of the base-level classifiers are described in terms of the original domain attributes as in the example MDT in Table 5.

Table 5: A meta decision tree induced on the image domain using base-level attributes as ordinary attributes.

```
short-line-density-5 <= 0:
|      short-line-density-2 <= 0:  KNN
|      short-line-density-2 > 0:  LTREE
short-line-density-5 > 0:
|      short-line-density-5 <= 0.111111:  LTREE
|      short-line-density-5 > 0.111111:  C45 (*)
```

The leaf denoted by an asterisk `(*)` in Table 5 specifies that C4.5 should be used to classify examples with `short-line-density-5` values larger than 0.11. MDTs based on the base-level ordinary attributes can provide new insight into the applicability of the base-level classifiers to the domain of use. However, only a human expert from the domain of use can interpret an MDT induced using these attributes. It cannot be interpreted directly from the point of view of classifier combination.

Note here another important property of MDTs induced using the CDP set of meta-level attributes. They are domain independent in the sense that the same language for expressing meta decision trees is used in all domains once we fix the set of base-level classifiers to be used. This means that a MDT induced on one domain can be used in any other domain for combining the same set of base-level classifiers (although it may not perform very well). In part, this is due to the fact that the CDP set of meta-level attributes is domain independent. It depends only on the set of base-level classifiers $\mathcal{C}$. However, an ODT built from the same set of meta-level attributes is still domain dependent for two reasons. First, it uses tests on the class values predicted by the base-level classifiers (e.g., the tests `C1 = 0` / `C1 = 1` in the root node of the ODT from Table 3b). Second, an ODT predicts the class value itself, which is clearly domain dependent.

In sum, there are three reasons for the domain independence of MDTs: (1) the CDP set of meta-level attributes; (2) not using class attributes in the decision (inner) nodes and (3) predicting the base-level classifier to be used instead of predicting the class value itself.

3.3 MLC4.5 - a Modification of C4.5 for Learning MDTs

In this subsection, we present MLC4.5[1], an algorithm for learning MDTs based on Quinlan's C4.5 [17] system for inducing ODTs. MLC4.5 takes as input a meta-level data set as generated by the algorithm in Table 1. Note that this data set consists of ordinary and class attributes. There are four differences between MLC4.5 and C4.5:

1. Only ordinary attributes are used in internal nodes;
2. Assignments of the form $C = C_i$ (where C_i is a class attribute) are made by MLC4.5 in leaf nodes, as opposed to assignments of the form $C = c_i$ (where c_i is a class value);
3. The goodness-of-split for internal nodes is calculated differently (as described below);
4. MLC4.5 does not post-prune the induced MDTs.

The rest of the MLC4.5 algorithm is identical to the original C4.5 algorithm. Below we describe C4.5's and MLC4.5's measures for selecting attributes in internal nodes.

[1] A patch that can be used to transform the source code of C4.5 into MLC4.5 is available at `http://ai.ijs.si/bernard/mdts/`

C4.5 is a greedy divide and conquer algorithm for building classification trees [17]. At each step, the best split according to the gain (or gain ratio) criterion is chosen from the set of all possible splits for all attributes. According to this criterion, the split is chosen that maximizes the decrease of the impurity of the subsets obtained after the split as compared to the impurity of the current subset of examples. The impurity criterion is based on the entropy of the class probability distribution of the examples in the current subset S of training examples:

$$\mathit{info}(S) = -\sum_{i=1}^{k} p(c_i, S) \cdot \log_2 p(c_i, S)$$

where $p(c_i, S)$ denotes the relative frequency of examples in S that belong to class c_i. The gain criterion selects the split that maximizes the decrement of the *info* measure.

In MLC4.5, we are interested in the accuracies of each of the base-level classifiers C from $\mathcal{C}$ on the examples in S, i.e., the proportion of examples in S that have a class equal to the class attribute C. The newly introduced measure, used in MLC4.5, is defined as:

$$\mathit{info}_{\mathcal{ML}}(S) = 1 - \max_{C \in \mathcal{C}} \mathtt{accuracy}(C, S),$$

where $\mathtt{accuracy}(C, S)$ denotes the relative frequency of examples in S that are correctly classified by the base-level classifier C. The vector of accuracies does not have probability distribution properties (its elements do not sum to 1), so the entropy can not be calculated. This is the reason for replacing the entropy based measure with an accuracy based one.

As in C4.5, the splitting process stops when at least one of the following two criteria is satisfied: (1) the accuracy of one of the classifiers on a current subset is 100% (leading to $\mathit{info}_{\mathcal{ML}}(S) = 0$); or (2) a user defined minimal number of examples is achieved in the current subset. In each case, a leaf node is being constructed. The classifier with the maximal accuracy is being predicted by a leaf node of a MDT.

In order to compare MDTs with ODTs in a principled fashion, we also developed a intermediate version of C4.5 (called AC4.5) that induces ODTs using the accuracy based $\mathit{info}_{\mathcal{A}}$ measure:

$$\mathit{info}_{\mathcal{A}}(S) = 1 - \max_{i=1}^{k} p(c_i, S).$$

4 Experimental Methodology and Results

The main goal of the experiments we performed was to evaluate the performance of meta decision trees, especially in comparison to other methods for combining classifiers, such as voting and stacking with ordinary decision trees, as well as other methods for constructing ensembles of classifiers, such as boosting and bagging. We also investigate the use of different meta-level attributes in MDTs. We performed experiments on a collection of twenty-one data sets from the UCI Repository of Machine Learning Databases and Domain Theories [3]. These data sets have been widely used in other comparative studies.

In the remainder of this section, we first describe how classification error rates were estimated and compared. We then list all the base-level and meta-level learning algorithms used in this study. Finally, we describe a measure of the diversity of the base-level classifiers that we use in comparing the performance of meta-level learning algorithms.

4.1 Estimating and Comparing Classification Error Rates

In all the experiments presented here, classification errors are estimated using 10-fold stratified cross validation. Cross validation is repeated ten times using a different random reordering of the examples in the data set. The same set of re-orderings have been used for all experiments.

For pair-wise comparison of classification algorithms, we calculated the relative improvement and the paired t-test, as described below. In order to evaluate the accuracy improvement achieved in a given domain by using classifier C_1 as compared to using classifier C_2, we calculate the relative improvement: $1 - \mathtt{error}(C_1)/\mathtt{error}(C_2)$. In the analysis presented in Section 5, we compare the performance of meta decision trees induced using CDP as ordinary meta-level attributes to other approaches: C_1 will thus refer to combiners induced by MLC4.5 using CDP. The average relative improvement across all domains is calculated using the geometric mean of error reduction in individual domains: $1 - \mathtt{geometric\_mean}(\mathtt{error}(C_1)/\mathtt{error}(C_2))$.

The classification errors of C_1 and C_2 averaged over the ten runs of 10-fold cross validation are compared for each data set ($\texttt{error}(C_1)$ and $\texttt{error}(C_2)$ refer to these averages). The statistical significance of the difference in performance is tested using the paired t-test (exactly the same folds are used for C_1 and C_2) with significance level of 95%: $+/-$ to the right of a figure in the tables with results means that the classifier C_1 is significantly better/worse than C_2.

Another aspect of tree induction performance is the simplicity of the induced decision trees. In the experiments presented here, we used the size of the decision trees, measured as the number of (internal and leaf) nodes, as a measure of simplicity: the smaller the tree, the simpler it is.

4.2 Base-Level Algorithms

Five learning algorithms have been used in the base-level experiments: two tree-learning algorithms C4.5 [17] and LTree [11], the rule-learning algorithm CN2 [7], the k-nearest neighbor (k-NN) algorithm [20] and a modification of the naive Bayes algorithm [12].

All algorithms have been used with their default parameters' settings. The output of each base-level classifier for each example in the test set consist of at least two components: the predicted class and the class probability distribution. All the base-level algorithms used in this study calculate the class probability distribution for classified examples, but two of them (k-NN and naive Bayes) do not calculate the weight of the examples used for classification (see Section 3). The code of the other three of them (C4.5, CN2 and LTree) was adapted to output the class probability distribution as well as the weight of the examples used for classification.

The classification errors of the base-level classifiers on the twenty-one data sets are presented in Table 7 in Appendix A. The smallest overall classification error is achieved using linear discriminant trees induced with LTree. However, on different data sets, different base-level classifiers achieve the smallest classification error.

4.3 Meta-Level Algorithms

At the meta-level, we evaluate the performances of eleven different algorithms for constructing ensembles of classifiers (listed below). Nine of these make use of exactly the same set of five base-level classifiers induced by the five algorithms from the previous section. In brief, two perform stacking with ODTs, using the algorithms C4.5 and AC4.5 (see previous section); three perform stacking with MDTs using the algorithm MLC4.5 and three different sets of meta-level attributes (CDP, BLA, CDP+BLA); two are voting schemes; Select-Best chooses the best base-level classifier, and SCANN performs stacking with nearest neighbor after analyzing dependencies among the base-level classifiers. In addition, boosting and bagging of decision trees are considered, which create larger ensembles (200 trees).

C-C4.5 uses ordinary decision trees induced with C4.5 for combining base-level classifiers.

C-AC4.5 uses ODTs induced with AC4.5 for combining base-level classifiers.

MDT-CDP uses meta decision trees induced with MLC4.5 on a set of class distribution properties (CDP) as meta-level attributes.

MDT-BLA uses MDTs induced with MLC4.5 on a set of base-level attributes (BLA) as meta-level attributes.

MDT-CDP+BLA uses MDTs induced with MLC4.5 on a union of two alternative sets of meta-level attributes (CDP and BLA).

P-VOTE is a simple plurality vote algorithm (see Section 2.1).

CD-VOTE is a refinement of the plurality vote algorithm for the case where class probability distributions are predicted by the base-level classifiers (see Section 2.1).

Select-Best selects the base-level classifier that performs best on the training set (as estimated by 10-fold stratified cross-validation). This is equivalent to building a single leaf MDT.

SCANN [14] performs **S**tacking with **C**orrespodence **A**nalysis and **N**earest **N**eighbours. Correspondence analysis is used to deal with the highly correlated predictions of the base-level classifiers: SCANN transforms the original set of potentially highly correlated meta-level attributes (i.e., predictions of the base-level classifiers), into a new (smaller) set of uncorrelated meta-level attributes. A nearest neighbor classifier is then used for classification with the new set of meta-level attributes.

Boosting of decision trees. Two hundred iterations were used for boosting. Decision trees were induced using J48[2], (C4.5) with default parameters' settings for pre- and post-pruning. The WEKA [21] data mining suite implements the AdaBoost [9] boosting method with re-weighting of the training examples.

Bagging of decision trees. Two hundred iterations (decision trees), were used for bagging, using J4.8 with default settings.

A detailed report on the performance of the above methods can be found in Appendix A. Their classification errors can be found in Table 9. The sizes of (ordinary and meta) decision trees induced with different meta-level combining algorithms are given in Table 8. Finally, a comparison of the classification errors of each method to those of MDT-CDP (in terms of average relative accuracy improvement and number of significant wins and losses) is given in Table 10. A summary of this detailed report is given in Table 6.

4.4 Diversity of Base-Level Classifiers

Empirical studies performed in [1, 2] show that the classification error of meta-level learning methods as well as the improvement of accuracy achieved using them is highly correlated to the degree of diversity of the predictions of the base-level classifiers. The measure of the diversity of two classifiers used in these studies is error correlation. The smaller the error correlation, the greater the diversity of the base-level classifiers.

[2]The experiments with bagging and boosting have been performed using the WEKA data mining suite [21] which includes J48, a Java re-implementation of C4.5. The differences between the J48 results and the C4.5 results are negligible: an average of 0.01% with a maximum relative difference of 4%.

Error correlation is defined by [1, 2] as the probability that both classifiers make the same error. This definition of error correlation is not "normalized": its maximum value is the lower of the two classification errors. An alternative definition of error correlation, proposed in [11], is used in this paper. Error correlation is defined as the conditional probability that both classifiers make the same error, given that one of them makes an error:

$$\hat{\phi}(C_i, C_j) = p(C_i(x) = C_j(x) | C_i(x) \neq c(x) \vee C_j(x) \neq c(x)),$$

where $C_i(x)$ and $C_j(x)$ are predictions of classifiers C_i and C_j for a given example x and $c(x)$ is the true class of x. The error correlation for a set of multiple classifiers $\mathcal{C}$ is defined as the average of the pairwise error correlations:

$$\hat{\phi}(\mathcal{C}) = \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{C_i \in \mathcal{C}} \sum_{C_j \neq C_i} \hat{\phi}(C_i, C_j).$$

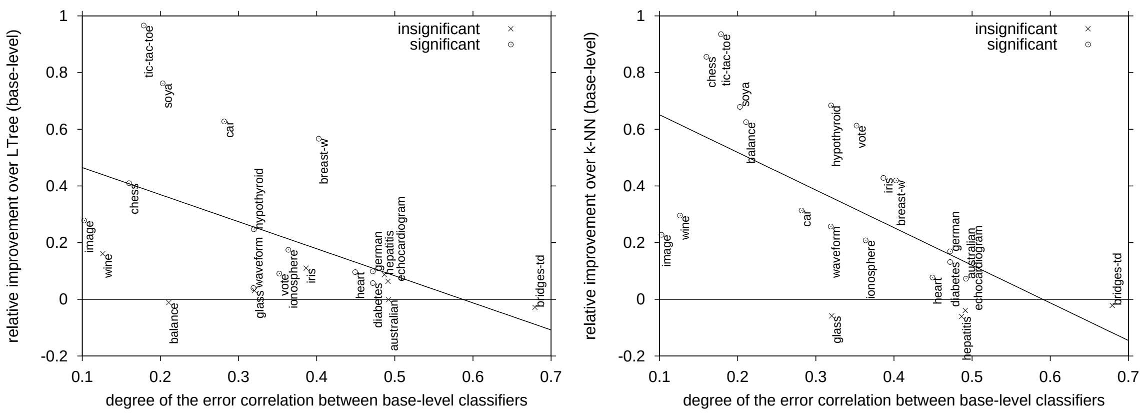

Figure 2: Relative accuracy improvement achieved with MDTs when compared to two base-level classifiers (LTree and k-NN) in dependence of the error correlation among the five base-level classifiers.

The graphs in Figure 2 confirm these results for meta decision trees. The relative accuracy improvement achieved with MDTs over LTree and k-NN (two base-level classifiers with highest overall accuracies) decreases as the error correlation of the base-level classifiers increases. The linear regression line interpolated through the points confirms this trend, which shows that performance improvement achieved with MDTs is correlated to the diversity of errors of the base-level classifiers.

Table 6: Summary performance of the meta-level learning algorithms as compared to MDTs induced using class distribution properties (MDT-CDP) as meta-level attributes: average relative accuracy improvement (in %), number of significant wins/losses and average tree size (where applicable). Details in Tables 10 and 8 in Appendix A.

Meta-level algorithm	Ave. rel. acc. imp. (in %)	Wins/losses	Ave. tree size
C-C4.5	-4.09	15/2	46.01
C-AC4.5	7.90	12/1	95.48
MDT-CDP	0.00	0/0	2.79
MDT-BLA	14.47	8/0	66.94
MDT-CDP+BLA	13.91	8/0	73.73
P-VOTE	22.33	10/3	NA
CD-VOTE	19.64	10/5	NA
Select-Best	7.81	5/2	1
SCANN	13.95	9/6	NA
Boosting	9.36	13/6	NA
Bagging	22.26	10/4	NA

5 Analysis of the Experimental Results

The results of the experiments are summarized in Table 6. In brief, the following main conclusions that can be drawn from these results:

1. The properties of the class probability distributions predicted by the base-level classifiers (CDP) are better meta-level attributes for inducing MDTs than the base-level attributes (BLA). Using BLA in addition to CDP worsens the performance.

2. Meta decision trees (MDTs) induced using CDP outperform ordinary decision trees and voting for combining classifiers.

3. MDTs perform slightly better than the SCANN and Select-Best methods.

4. The performance improvement achieved with MDTs is correlated to the diversity of errors of the base-level classifiers: the higher the diversity, the better the relative performance as compared to other methods.

5. Using MDTs to combine classifiers induced with different learning algorithms outperforms ensemble learning methods based on bagging and boosting of decision trees.

Below we look into these claims and the experimental results in some more detail.

5.1 MDTs with Different Sets of Meta-Level Attributes

We analyzed the dependence of MDTs performance on the set of ordinary meta-level attributes used to induce them. We used three sets of attributes: the properties of the class distributions predicted by the base-level classifiers (CDP), the original (base-level) domain attributes (BLA) and their union (CDP+BLA).

The average relative improvement achieved by MDTs induced using CDP over MDTs induced using BLA and CDP+BLA is about 14% with CDP being significantly better in 8 domains (see Table 6 and Table 10 in Appendix A). MDTs induced using CDP are about 25 times smaller on average than MDTs induced using BLA and CDP+BLA (see Table 8). These results show that the CDP set of meta-level attributes is better than the BLA set. Furthermore, using BLA in addition to CDP decreases performance. In the remainder of this Section, we only consider MDTs induced using CDP.

An analysis of the results for ordinary decision trees induced using CDP, BLA and CDP+BLA (only the results for CDP are actually presented in the paper) shows that the claim holds also for ODTs. This result is especially important because it highlights the importance of using the class probability distributions predicted by the base-level classifiers for identifying their (relative) areas of expertise. So far, base-level attributes from the original domain have typically been used to identify the areas of expertise of base-level classifiers.

5.2 Meta Decision Trees vs. Ordinary Decision Trees

To compare combining classifiers with MDTs and ODTs, we first look at the relative improvement of using MLC4.5 over C-C4.5 (see Table 6, column C-C4.5 of Table 10 in Appendix A and left hand side of Figure 3).

MLC4.5 performs significantly better in 15 and significantly worse in 2 data sets. There is a 4% overall decrease of accuracy (this is a geometric mean), but this is entirely due to the result in the tic-tac-toe domain, where all combining methods perform very well. If we exclude the tic-tac-toe domain, a 7% overall relative increase is obtained. We can thus say that MLC4.5 performs slightly better in terms of accuracy. However, the MDTs are much smaller, the size reduction factor being over 16 (see Table 8), despite the fact that ODTs induced with C4.5 are post-pruned and MDTs are not.

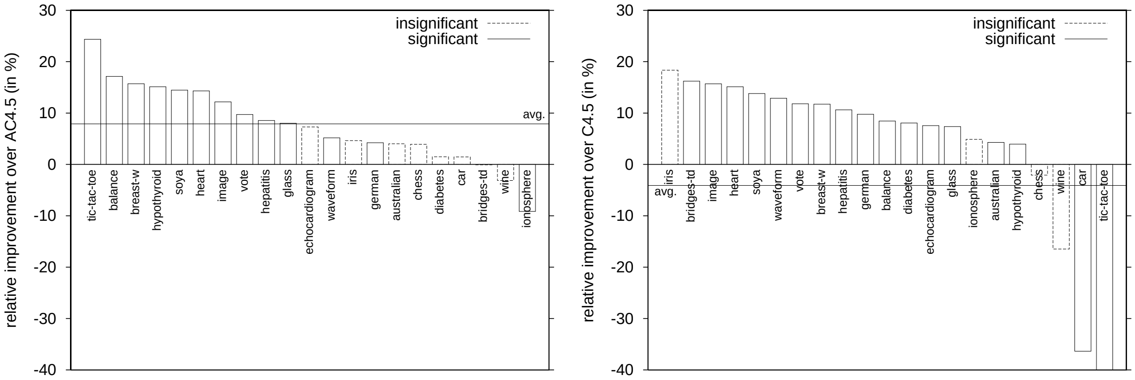

Figure 3: Relative improvement of the accuracy when using MDTs induced with MLC4.5 when compared to the accuracy of ODTs induced with AC4.5 and C4.5.

To get a clearer picture of the performance differences due to the extended expressive power of MDT leaves (as compared to ODT leaves), we compare MLC4.5 and C-AC4.5 (see Table 6, column C-AC4.5 in Table 10 and right hand side of Figure 3). Both MLC4.5 and AC4.5 use the same learning algorithm. The only difference between them is the types of trees they induce: MLC4.5 induces meta decision trees and AC4.5 induces ordinary ones. The comparison clearly shows that MDTs outperform ODTs for combining classifiers. The overall relative accuracy improvement is about 8% and MLC4.5 is significantly better than C-AC4.5 in 12 out of 21 data sets and is significantly worse in only one (ionosphere). Consider also the graph on the right hand side of Figure 3. MDTs perform better than ODTs in all but two domains, with the performance gains being much larger than the losses.

Furthermore, the MDTs are, on average, more than 34 times smaller than the ODTs induced with AC4.5 (see Table 8). The reduction of the tree size improves the comprehensibility of meta decision trees. For example, we were able to interpret and comment on the MDT in Table 4.

In sum, meta decision trees performs better than ordinary decision trees for combining classifiers: MDTs are more accurate and much more concise. The comparison of MLC4.5 and AC4.5 shows that the performance improvement is due to the extended expressive power of MDT leaves.

5.3 Meta Decision Trees vs. Voting

Combining classifiers with MDTs is significantly better than plurality vote in 10 domains and significantly worse in 6. However, the significant improvements are much higher than the significant drops of accuracy, giving an overall accuracy improvement of 22%. Since CD-VOTE performs slightly better than plurality vote, a smaller overall improvement of 20% is achieved with MDTs. MLC4.5 is significantly better in 10 data sets and significantly worse in 5. These results show that MDTs outperform the voting schemes for combining classifiers (see Table 6 and Table 10 in Appendix A).

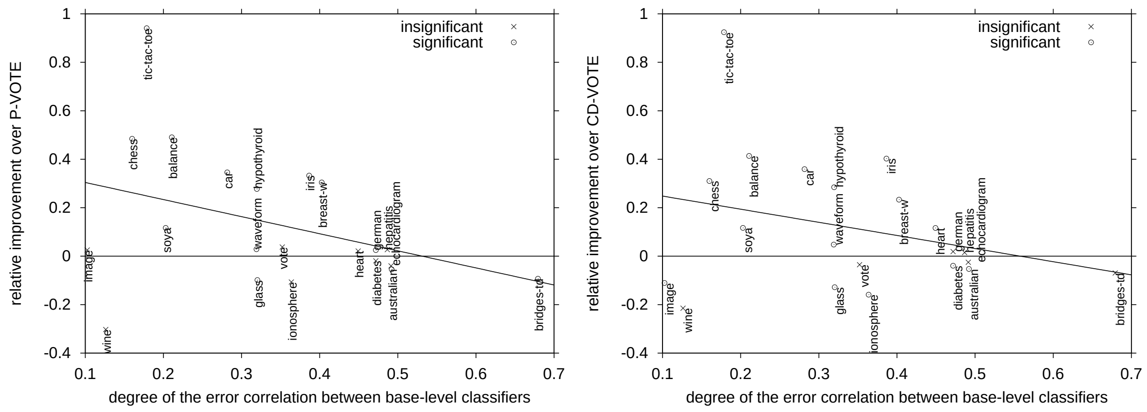

Figure 4: Relative improvement of the accuracy of MDTs over two voting schemes in dependence of the degree of error correlation between the base-level classifiers.

We explored the dependence of accuracy improvement by MDTs over voting on the diversity of the base-level classifiers. The graphs in Figure 4 show that MDTs can make better use of the diversity of errors of the base-level classifiers than the voting schemes. Namely, for the domains with low error correlation (and therefore higher diversity) of the base-level classifiers, the relative improvement of MDTs over the voting schemes is higher. However, the slope of the linear regression line is smaller than the one for the improvement over the base-level classifiers. Still, the trend clearly shows that MDTs make better use of the error diversity of the base-level predictions than voting.

5.4 Meta Decision Trees vs. Select-Best

Combining classifiers with MDTs is significantly better than Select-Best in 5 domains and significantly worse in 2, giving an overall accuracy improvement of almost 8% (see Table 6 and Table 10 in Appendix A).

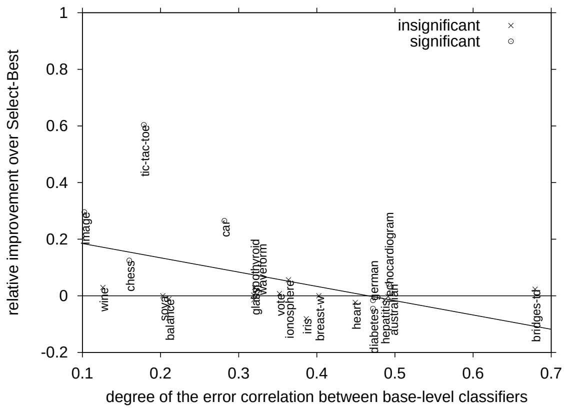

Figure 5: Relative improvement of the accuracy of MDTs over Select-Best method in dependence of the degree of error correlation between the base-level classifiers.

The results of the dependence analysis of accuracy improvement by MDTs over Select-Best on the diversity of the base-level classifiers is given in Figure 5. MDTs can make slightly better use of the diversity of errors of the base-level classifiers than Select-Best. The slope of the linear regression line is smaller than the one for the improvement over the voting methods.

5.5 Meta Decision Trees vs. SCANN

Combining classifiers with MDTs is significantly better than SCANN in 9 domains and significantly worse in 6 (see Table 6 and Table 10 in Appendix A). However, the significant improvements are much higher than the significant drops of accuracy, giving an overall accuracy improvement of almost 14%. These results show that MDTs outperform the SCANN method for combining classifiers.

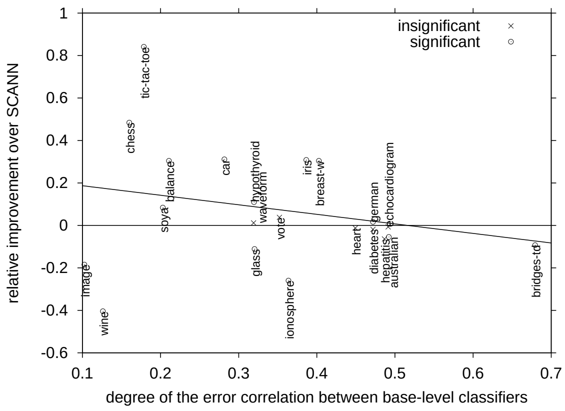

Figure 6: Relative improvement of the accuracy of MDTs over SCANN method in dependence of the degree of error correlation between the base-level classifiers.

We explored the dependence of accuracy improvement by MDTs over SCANN on the diversity of the base-level classifiers. The graph in Figure 6 shows that MDTs can make a slightly better use of the diversity of errors of the base-level classifiers than SCANN. The slope of the linear regression line is smaller than the one for the improvement over the voting methods.

5.6 Meta Decision Trees vs. Boosting and Bagging

Finally, we compare the performance of MDTs with the performance of two state of the art ensemble learning methods: bagging and boosting of decision trees.

MLC4.5 performs significantly better than boosting in 13 and significantly better than bagging in 10 out of the 21 data sets. MLC4.5 performed significantly worse than boosting in 6 domains and significantly worse than bagging in 4 domains only. The overall relative improvement of performance is 9% over boosting and 22% over bagging (see Table 6 and Table 10 in Appendix A).

It is clear that MDTs outperform bagging and boosting of decision trees. This comparison is not fair in the sense that MDTs use base-level classifiers induced by decision trees and four other learning methods, while boosting and bagging use only decision trees as a base-level learning algorithm. However, it does show that our approach to constructing ensembles of classifiers is competitive to existing state of the art approaches.

6 Related Work

An overview of methods for constructing ensembles of classifiers can be found in [8]. Several meta-level learning studies are closely related to our work.

Let us first mention the study of using SCANN [14] for combining base-level classifiers. As mentioned above, SCANN performs stacking by using correspondence analysis of the classifications of the base-level classifiers. The author shows that SCANN outperforms the plurality vote scheme, also in the case when the base-level classifiers are highly correlated. SCANN does not use any class probability distribution properties of the predictions by the base-level classifiers (although the possibility of extending the method in that direction is mentioned). Therefore, no comparison with the CD voting scheme is included in their study. Our study shows that MDTs using CDP attributes are slightly better than SCANN in terms of performance. Also, the concept induced by SCANN at the meta-level is not directly interpretable and can not be used for identifying the relative areas of expertise of the base-level classifiers.

In cascading, base-level classifiers are induced using the examples in a current node of the decision tree (or each step of the divide and conquer algorithm for building decision trees). New attributes, based on the class probability distributions predicted by the base-level classifiers, are generated and added to the set of the original attributes in the domain. The base-level classifiers used in this study are naive Bayes and Linear Discriminant. The integration of these two base-level classifiers within decision trees is much tighter than in our combining/stacking framework. The similarity to our approach is that class probability distributions are used.

A version of stacked generalization, using the class probability distributions predicted by the base-level classifiers, is implemented in the data mining suite WEKA [21]. However, class probability distributions are used there directly and not through their properties, such as maximal probability and entropy. This makes them domain dependent in the sense discussed in Section 3. The indirect use of class probability distributions through their properties makes MDTs domain independent.

Ordinary decision trees have already been used for combining multiple classifiers in [6]. However, the emphasis of their study is more on partitioning techniques for massive data sets and combining multiple classifiers trained on different subsets of massive data sets. Our study focuses on combining multiple classifiers generated on the same data set. Therefore, the obtained results are not directly comparable to theirs.

Combining classifiers by identifying their areas of expertise has already been explored in [15] and [13]. In their studies, a description of the area of expertise in the form of an ordinary decision tree, called arbiter, is induced for each individual base-level classifier. For a single data set, as many arbiters are needed as there are base-level classifiers. When combining multiple classifiers, a voting scheme is used to combine the decisions of the arbiters. However, a single MDT, identifying relative areas of expertise of all base-level classifiers at once is much more comprehensible. Another improvement presented in our study is the possibility to use the certainty and confidence of the base-level predictions for identifying the classifiers expertise areas and not only the original (base-level) attributes of the data set.

The present study is also related to our previous work on the topic of meta-level learning [18]. There we introduced an inductive logic programming [16] (ILP) framework for learning the relation between data set characteristics and the performance of different (base-level) classifiers. A more expressive (non-propositional) formulation is used to represent the meta-level examples (data set characteristics), e.g., properties of individual attributes.

The induced meta-level concepts are also non-propositional. While MDT leaves are more expressive than ODT leaves, the language of MDTs is still much less expressive than the language of logic programs used in ILP.

7 Conclusions and Further Work

We have presented a new technique for combining classifiers based on meta decision trees (MDTs). MDTs make the language of decision trees more suitable for combining classifiers: they select the most appropriate base-level classifier for a given example. Each leaf of the MDT represents a part of the data set, which is a relative area of expertise of the base-level classifier in that leaf. The relative areas of expertise can be identified on the basis of the values of the original (base-level) attributes (BLA) of the data set, but also on the basis of the properties of the class probability distributions (CDP) predicted by the base-level classifiers. The latter reflect the certainty and confidence of the class value predictions by the individual base-level classifiers.

The extensive empirical evaluation shows that MDTs induced from CDPs perform much better and are much more concise than MDTs induced from BLAs. Due to the extended expressiveness of MDT leaves, they also outperform ordinary decision trees (ODTs), both in terms of accuracy and conciseness. MDTs are usually so small that they can easily be interpreted: we regard this as a step towards a comprehensible model of combining classifiers by explicitly identifying their relative areas of expertise. In contrast, most existing work uses non-symbolic learning methods (e.g., neural networks) to combine classifiers [14].

MDTs can use the diversity of the base-level classifiers better than voting: they outperform voting schemes in terms of accuracy, especially in domains with high diversity of the errors made by the base-level classifiers. MDTs also perform slightly better than the SCANN method for combining classifiers and the Select-Best method, which simply takes the best single classifier. Finally, MDTs induced from CDPs perform better than boosting and bagging of decision trees and are thus competitive with state of the art methods for learning ensembles.

MDTs built by using CDPs are domain independent and are, in principle, transferable across domains once we fix the set of base-level learning algorithms. This in the sense that a MDT built on one data set can be used on any other data set (since it uses the same set of attributes). There are several potential benefits of the domain independence of MDTs. First, machine learning experts can use MDTs for domain independent analysis of relative areas of expertise of different base-level classifiers, without having knowledge about the particular domain of use. Furthermore, an MDT induced on one data set can be used for combining classifiers induced by the same set of base-level learning algorithms on other data sets. Finally, MDTs can be induced using data sets that contain examples originating from different domains.

Exploring the above options already gives us some topics for further work. Combining data from different domains for learning MDTs is an especially interesting avenue for further work that would bring together the present study with meta-level learning work on selecting appropriate classifiers for a given domain [4]. In this case, attributes describing individual data set properties can be added to the class distribution properties in the meta-level learning data set. Preliminary investigations along these lines have been already made [19].

There are several other obvious directions for further work. For ordinary decision trees, it is already known that post-pruning gives better results than pre-pruning. Preliminary experiments show that pre-pruning degrades the classification accuracy of MDTs. Thus, one of the priorities for further work is the development of a post-pruning method for meta decision trees and its implementation in MLC4.5.

An interesting aspect of our work is that we use class-distribution properties for meta-level learning. Most of the work on combining classifiers only uses the predicted classes and not the corresponding probability distributions. It would be interesting to use other learning algorithms (neural networks, Bayesian classification and SCANN) to combine classifiers based on the probability distributions returned by them. A comparison of combining classifiers using class predictions only vs. with class predictions along with class probability distributions would be also worthwhile.

The consistency of meta decision trees with common sense classifiers combination knowledge, as briefly discussed in Section 3, opens another question for further research. The process of inducing meta-level classifiers can be biased to produce only meta-level classifiers consistent with existing knowledge. This can be achieved using strong language bias within MLC4.5 or, probably more easily, within a framework of meta decision rules, where rule templates could be used.

Acknowledgments

The work reported was supported in part by the Slovenian Ministry of Education, Science and Sport and by the EU-funded project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (IST-1999-11495). We thank João Gama for many insightful and inspirational discussions about combining multiple classifiers. Many thanks to Marko Bohanec, Thomas Dietterich, Nada Lavrač and three anonymous reviewers for their comments on earlier versions of the manuscript.

References

[1] Ali, K. M. (1996) On explaining degree of error reduction due to combining multiple decision trees. *AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* [`http://www.cs.fit.edu/~imlm/imlm96/`].

[2] Ali, K. M. and Pazzani, M. J. (1996) Error reduction through learning multiple descriptions. *Machine Learning* 24: 173–202.

[3] Blake, C. L. and Merz, C. J. (1998) *UCI repository of machine learning databases.* [`http://www.ics.uci.edu/~mlearn/MLRepository.html`]. University of California, Irvine, Department of Information and Computer Sciences.

[4] Brazdil, P. B. and Henery, R. J. (1994) Analysis of Results. In Michie, D., Spiegelhalter, D. J., and Taylor, C. C., editors: *Machine learning, neural and statistical classification.* Ellis Horwood.

[5] Breiman, L. (1996) Bagging Predictors. *Machine Learning* 24(2): 123–140.

[6] Chan, P. K. and Stolfo, S. J. (1997) On the Accuracy of Meta-learning for Scalable Data Mining. *Journal of Intelligent Information Systems* 8(1): 5–28.

[7] Clark, P. and Boswell, R. (1991) Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*: 151–163. Springer-Verlag.

[8] Dietterich, T. G. (1997) Machine-Learning Research: Four Current Directions. *AI Magazine* 18(4): 97–136.

[9] Freund, Y. and Schapire, R. E. (1996) Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann.

[10] Gama, J. (1998) Combining Classifiers by Constructive Induction. In *Proceedings of the Ninth European Conference on Machine Learning.*

[11] Gama, J. (1999) Discriminant trees. In *Proceedings of the Sixteenth International Conference on Machine Learning*: 134-142. Morgan Kaufmann.

[12] Gama, J. (2000) A Linear-Bayes Classifier. Technical Report. Artificial Intelligence and Computer Science Laboratory, University of Porto.

[13] Koppel, M. and Engelson, S. P. (1996) Integrating multiple classifiers by finding their areas of expertise. *AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* [`http://www.cs.fit.edu/~imlm/imlm96/`].

[14] Merz, C. J. (1999) Using Correspondence Analysis to Combine Classifiers. *Machine Learning* 36(1/2): 33–58. Kluwer Academic Publishers.

[15] Ortega, J. (1996) Exploiting Multiple Existing Models and Learning Algorithms. *AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* [`http://www.cs.fit.edu/~imlm/imlm96/`].

[16] Quinlan, J. R. (1990) Learning logical definitions from relations. *Machine Learning 5*: 239–266.

[17] Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

[18] Todorovski, L. and Džeroski, S. (1999) Experiments in Meta-Level Learning with ILP. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*: 98–106. Springer-Verlag.

[19] Todorovski, L. and Džeroski, S. (2000) Combining two aspects of meta-learning with heterogeneous meta decision trees. In *Proceedings of the Fifth International Workshop on Multistrategy Learning*: 221–232.

[20] Wettschereck, D. (1994) *A study of distance-based machine learning algorithms.* PhD Thesis, Department of Computer Science, Oregon State University, Corvallis.

[21] Witten, I. H. and Frank, E. (1999) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann.

[22] Wolpert, D. (1992) Stacked Generalization. *Neural Networks* 5(2): 241–260.

A Tables with Experimental Results

Table 7: Classification errors (in %) of the base-level classifiers estimated using ten runs of 10-fold cross validation. The average values are calculated using arithmetic mean.

Data set	C4.5		CN2		k-NN		LTree		LinBayes	
australian	14.19	±0.49	17.41	±0.50	15.45	±0.52	14.32	±0.56	14.13	±0.24
balance	22.43	±0.57	19.95	±0.91	19.11	±1.43	7.08	±0.64	13.31	±0.33
breast-w	5.37	±0.26	6.43	±0.42	4.53	±0.21	6.07	±0.69	2.63	±0.08
bridges-td	14.91	±0.61	14.68	±1.11	14.83	±1.30	14.73	±0.29	12.10	±0.68
car	7.67	±0.85	5.01	±0.57	5.90	±0.76	10.87	±0.54	14.12	±0.28
chess	0.52	±0.03	1.58	±0.25	3.39	±0.30	0.83	±0.05	12.29	±0.48
diabetes	25.76	±1.27	25.89	±0.52	27.31	±0.71	25.16	±1.15	22.58	±0.39
echo	33.51	±2.37	33.46	±1.86	31.40	±2.90	34.84	±3.13	29.16	±1.45
german	28.64	±0.53	26.59	±0.59	30.02	±0.52	27.67	±0.71	24.45	±0.34
glass	32.43	±2.27	32.91	±1.73	29.80	±0.85	32.54	±2.56	36.94	±0.77
heart	23.01	±2.18	22.46	±1.40	17.52	±1.08	17.89	±0.74	15.11	±0.55
hepatitis	21.40	±1.32	19.59	±1.16	15.93	±1.34	18.50	±1.95	15.10	±0.71
hypo	0.72	±0.04	1.42	±0.25	2.31	±0.06	0.97	±0.04	3.14	±0.02
image	3.34	±0.26	14.05	±0.46	2.99	±0.14	3.20	±0.21	8.43	±0.10
ionosphere	10.66	±0.96	9.29	±0.92	12.06	±0.38	11.57	±1.23	13.50	±0.57
iris	5.88	±0.88	7.48	±0.82	4.67	±0.44	3.00	±0.85	2.00	±0.00
soya	8.00	±0.50	8.13	±0.36	17.69	±0.47	23.86	±0.80	5.61	±0.15
tic-tac-toe	14.72	±0.95	1.49	±0.17	9.14	±1.17	17.46	±1.42	30.42	±0.25
vote	3.54	±0.33	4.74	±0.40	9.84	±0.49	4.19	±0.40	9.77	±0.25
waveform	23.61	±0.41	30.81	±0.31	18.76	±0.36	14.52	±0.30	13.97	±0.16
wine	7.83	±1.63	6.95	±1.51	2.81	±0.74	2.36	±0.34	1.46	±0.30
Average	14.67	±0.89	14.78	±0.77	14.07	±0.77	13.89	±0.89	14.30	±0.39

Table 8: Sizes (in number of nodes) of MDTs induced with MLC4.5 and ODTs induced with AC4.5 and C4.5.

Domain	C-C4.5	C-AC4.5	MDT-CDP	MDT-BLA	MDT-CDP+BLA
australian	35.78	55.48	3.66	183.79	220.49
balance	32.75	91.85	1.12	1.02	1.16
breast-w	6.14	17.04	1.00	73.49	76.19
bridges-td	7.70	12.60	2.30	19.70	41.32
car	43.36	148.76	5.32	117.52	123.36
chess	10.20	15.82	2.06	75.05	79.39
diabetes	79.34	69.80	4.30	4.68	6.06
echo	22.22	45.70	2.08	2.98	4.48
german	100.22	75.14	5.64	517.40	535.05
glass	51.16	211.22	3.40	2.98	3.62
heart	21.36	44.48	1.82	63.68	64.72
hepatitis	15.76	33.82	1.70	49.62	49.77
hypo	4.70	24.92	1.18	37.16	41.23
image	65.69	306.11	4.24	7.08	11.10
ionosphere	18.96	37.40	3.28	2.98	4.00
iris	5.73	13.04	1.84	1.00	1.88
soya	80.03	480.27	1.68	71.44	75.98
tic-tac-toe	7.28	28.78	3.12	64.14	70.18
vote	9.18	24.78	1.16	104.99	129.31
waveform	344.19	255.26	6.34	4.14	7.66
wine	4.44	12.74	1.36	1.00	1.36
Average	46.01	95.48	2.79	66.94	73.73

Table 9: Classification errors (in %) of the eleven meta-level algorithms estimated using ten runs of 10-fold cross validation. The average values are calculated using arithmetic mean.

Data set	C-C4.5	C-AC4.5	MDT-CDP	MDT-BLA	MDT-CDP+BLA	P-VOTE	CD-VOTE	Select-Best	SCANN	Boosting	Bagging
australian	14.97	14.93	14.33	14.70	14.95	13.59	13.61	14.95	13.59	13.23	13.25
balance	7.82	8.64	7.16	7.09	7.19	14.05	12.21	7.08	10.29	24.53	16.54
breast-w	2.98	3.12	2.63	4.70	4.72	3.78	3.43	2.63	3.78	3.22	4.46
bridges-td	18.08	15.14	15.15	16.00	16.20	13.86	14.17	15.51	13.86	18.08	14.73
car	2.97	4.11	4.05	4.62	4.47	6.19	6.32	5.51	5.88	3.10	6.39
chess	0.48	0.51	0.49	0.52	0.52	0.95	0.71	0.56	0.95	0.32	0.56
diabetes	25.80	24.08	23.72	23.32	23.90	23.27	22.82	22.69	23.29	25.59	23.41
echo	35.30	35.20	32.63	32.71	32.87	31.37	31.83	32.54	32.40	34.71	30.71
german	27.64	26.04	24.94	26.79	26.88	25.58	25.42	24.55	25.37	25.61	24.46
glass	34.05	34.29	31.54	30.09	31.80	28.71	27.96	32.49	28.39	20.77	23.97
heart	19.05	18.87	16.17	19.98	19.84	16.52	18.30	15.80	15.97	20.11	18.96
hepatitis	18.90	18.47	16.89	18.36	18.14	17.36	17.15	16.63	15.88	16.50	17.02
hypo	0.76	0.86	0.73	0.78	0.78	1.01	1.02	0.72	0.82	1.06	0.76
image	2.74	2.63	2.31	2.62	2.31	2.37	2.08	3.28	1.95	1.36	2.35
ionosphere	10.04	8.75	9.55	9.81	9.32	8.63	8.24	10.12	7.58	6.01	6.98
iris	3.27	2.80	2.67	2.47	2.62	4.00	4.47	2.47	3.86	5.53	5.80
soya	6.59	6.64	5.68	6.13	6.03	6.43	6.43	5.68	6.20	6.62	6.77
tic-tac-toe	0.06	0.78	0.59	2.80	2.73	9.94	7.78	1.49	3.72	0.85	5.12
vote	4.32	4.22	3.81	5.26	5.12	3.96	3.68	3.84	3.96	4.32	3.68
waveform	16.00	14.70	13.94	14.06	13.99	14.35	14.64	13.98	14.11	15.23	16.34
wine	1.70	1.92	1.98	2.04	1.98	1.52	1.63	2.04	1.41	3.09	3.83
Average	12.07	11.75	11.00	11.66	11.73	11.78	11.61	11.17	11.11	11.90	11.72

Table 10: Relative improvement (in %) of the accuracy of the MDTs induced using class distribution properties (CDP) as ordinary attributes when compared to all the other meta-level algorithms.

Data set	C-C4.5	C-AC4.5	MDT-BLA	MDT-CDP+BLA	P-VOTE	CD-VOTE	Select-Best	SCANN	Boosting	Bagging
australian	4.28 +	4.02	2.52	4.15	-5.45 -	-5.29 -	4.15 +	-5.45 -	-8.31 -	-8.15 -
balance	8.44 +	17.13 +	-0.99	0.42	49.04 +	41.36 +	-1.13	30.42 +	70.81 +	56.71 +
breast-w	11.74 +	15.71 +	44.04 +	44.28 +	30.42 +	23.32 +	0.00	30.42 +	18.32 +	41.03 +
bridges-td	16.21 +	-0.07	5.31	6.48 +	-9.31 -	-6.92	2.32	-9.31 -	16.21 +	-2.85
car	-36.36 -	1.46	12.34 +	9.40 +	34.57 +	35.92 +	26.50 +	31.12 +	-30.65 -	36.62 +
chess	-2.08	3.92	5.77	5.77	48.42 +	30.99 +	12.50 +	48.42 +	-53.12 -	12.50 +
diabetes	8.06 +	1.50	-1.72	0.75	-1.93	-3.94 -	-4.54 -	-1.85	7.31 +	-1.32
echo	7.56 +	7.30	0.24	0.73	-4.02	-2.51	-0.28	-0.71	5.99 +	-6.25
german	9.77 +	4.22 +	6.91 +	7.22 +	2.50 +	1.89	-1.59 -	1.69 +	2.62 +	-1.96 -
glass	7.37 +	8.02 +	-4.82	0.82	-9.86 -	-12.80 -	2.92	-11.10 -	-51.85 -	-31.58 -
heart	15.12 +	14.31 +	19.07 +	18.50 +	2.12	11.64 +	-2.34	-1.25	19.59 +	14.72 +
hepatitis	10.63 +	8.55 +	8.01	6.89	2.71	1.52	-1.56	-6.36	-2.36	0.76
hypo	3.95 +	15.12 +	6.41 +	6.41 +	27.72 +	28.43 +	-1.39	10.98 +	31.13 +	3.95
image	15.69 +	12.17 +	11.83 +	0.00	2.53	-11.06 -	29.57 +	-18.46 -	-69.85 -	1.70
ionosphere	4.88	-9.14 -	2.65	-2.47	-10.66	-15.90 -	5.63	-25.99 -	-58.90 -	-36.82 -
iris	18.35	4.64	-8.10	-1.91	33.25 +	40.27 +	-8.10	30.83 +	51.72 +	53.97 +
soya	13.81 +	14.46 +	7.34	5.80	11.66 +	11.66 +	0.00	8.39 +	14.20 +	16.10 +
tic-tac-toe	-883.33 -	24.36 +	78.93 +	78.39 +	94.06 +	92.42 +	60.40 +	84.14 +	30.59	88.48 +
vote	11.81 +	9.72 +	27.57 +	25.59 +	3.79	-3.53	0.78	3.79	11.81 +	-3.53
waveform	12.88 +	5.17 +	0.85	0.36	2.86 +	4.78 +	0.29	1.20	8.47 +	14.69 +
wine	-16.47	-3.12	2.94	0.00	-30.26	-21.47	2.94	-40.43 -	35.92 +	48.30 +
Average	-4.09 15/2	7.90 12/1	14.47 8/0	13.91 8/0	22.33 10/3	19.64 10/5	7.81 5/2	13.95 9/6	9.36 13/6	22.26 10/4