Combining Multiple Models with Meta Decision Trees

Ljupčo Todorovski, Sašo Džeroski

Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Ljupco.Todorovski@ijs.si, Saso.Dzeroski@ijs.si

Abstract. The paper introduces meta decision trees (MDTs), a novel method for combining multiple models. Instead of giving a prediction, MDT leaves specify which model 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 models 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, MDTs combine models better than voting and stacking with ODTs. In addition, MDTs are much more concise than ODTs used for stacking and are thus a step towards comprehensible combination of multiple models.

1 Introduction

The task of combining multiple models can be broken down into two subtasks. The first is the *generation* of a diverse set of base-level models. Once the base-level models have been generated, the issue of *combination* of their predictions arises. This is the second subtask. The task of multiple model combination is the focus of this paper.

Several approaches for generating base-level models have been developed. One way is to generate multiple models with different learning algorithms for heterogeneous model representations as in [10]. Another way is to use a single learning algorithm with different initial settings. Multiple models can also be generated by applying a single base-level learning algorithm to different versions of learning data. Different methods for manipulating the set of learning examples can be used, such as random sampling with replacement (also called bootstrap aggregation) in bagging [2] or re-weighting misclassified training examples in boosting [6].

The techniques for combining the predictions obtained from the base-level models can be clustered in three combining paradigms: voting (used in bagging and boosting), stacking [15] and cascading [7]. In a voting scheme, each base-level model gives a vote for its prediction. The prediction receiving the most votes is the final prediction. In stacking, a learning algorithm is used to induce a meta-level model for combining the predictions of the base-level models. Cascading is an iterative process of combining classifiers: at each iteration, the learning data set is extended with the predictions obtained in the previous iteration.

This paper introduces meta decision trees (MDTs), a novel method for combining multiple models. The difference between meta and ordinary decision trees (ODTs) is that MDT leaves specify which base-level model should be used, instead of predicting the class value directly. The decisions are made based on the class probability distributions for the given example predicted by the base-level models. The method is general in the sense that it can be used to combine a set of predictions of base-level models, independently of how they are generated. We developed MLC4.5, a modification of C4.5 [12], for inducing meta decision trees. The description of the method is given in Section 2.

The performance of the proposed method is evaluated on a collection of twenty-one data sets. We combine models generated by five learning algorithms: two tree-learning algorithms C4.5 [12] and LTree [8], the rule-learning algorithm CN2 [4], the k-nearest neighbor (k-NN) algorithm [14] and a modification of the naive Bayes algorithm [9]. 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. Section 3 reports on the experimental methodology and results. The presented work is put in the context of previous work on combining multiple models in Section 4. Section 5 presents conclusions based on the empirical evaluation along with directions for further work.

2 Meta Decision Trees

2.1 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 model is to be used for classification of an example, instead of predicting the class value of the example directly like an ODT.

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. The role of these attributes is identical to the role of the attributes used for inducing ordinary decision trees. *Class* attributes are used in the leaf nodes only. Each base-level model has its class attribute: the values of the class attribute are equal to the predictions obtained by the base-level model. Thus, the class attribute assigned to the leaf node of the MDT decides which base-level model should be used for prediction.

Attributes in MDTs are properties of the class probability distributions predicted for a given example by the base-level models. Namely, the most general form of a prediction returned by a classification model for a given example is a probability distribution over the possible classes. Let the base-level classifier $\mathcal{C}_\mathcal{L}$ generated with learning algorithm $\mathcal{L}$ return the probability distribution $\mathbf{p}_\mathcal{L}(e)$, when applied to example e: $\mathbf{p}_{\mathcal{L}}(e) = (p_{\mathcal{L}}^{(1)}(e), p_{\mathcal{L}}^{(2)}(e), \ldots, p_{\mathcal{L}}^{(m)}(e))$, where m is the number of classes. The k-th element in this vector denotes the probability that the example e belongs to class c_k as estimated by the model $\mathcal{C}_{\mathcal{L}}$. The class c_* with the highest class probability $p_{\mathcal{L}}^{(*)}$ is predicted by base-level classifier $\mathcal{C}_{\mathcal{L}}$.

The following three properties of class probability distributions are used as attributes in MDTs. First, $\mathcal{L}$_maxprob is the highest class probability (i.e. the probability of the predicted class):

$$\mathcal{L}\text{\_maxprob} = \max_{k=1}^{m} p_{\mathcal{L}}^{(k)}(e).$$

Next, $\mathcal{L}$_entropy is the entropy of the class probability distribution:

$$\mathcal{L}\text{\_entropy} = -\sum_{k=1}^{m} p_{\mathcal{L}}^{(k)}(e) \cdot \log_2 p_{\mathcal{L}}^{(k)}(e).$$

Finally, the $\mathcal{L}_{\text{weight}}$ is the fraction of the training examples used to estimate the class distribution for example e. 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 was used to classify the example. This property does not apply to the nearest neighbor and naive Bayes classifiers.

Table 1. A meta decision tree learned in the balance domain.

```
ltree_entropy <= 0.37699:
|       knn_maxprob <= 0.75079: LTREE (*)
|       knn_maxprob > 0.75079: KNN
ltree_entropy > 0.37699:
|       knn_entropy > 1.49841: KNN
|       knn_entropy <= 1.49841:
|       |       c45_weight <= 0.11388: LTREE
|       |       c45_weight > 0.11388:
|       |       |       c45_maxprob <= 0.95: LTREE
|       |       |       c45_maxprob > 0.95: C45
```

Both the entropy and the maximum probability of a probability distribution can be interpreted as estimates of the confidence of the model in its prediction. If the probability distribution returned is highly spread, the maximum probability will be low and the entropy will be high, indicating the model is not very confident in its prediction. On the other hand, if the probability distribution returned is highly focussed, the maximum probability is high and the entropy low, thus indicating the model is confident in its prediction. Finally, the weight quantifies how reliable the model's estimate of its own confidence is: the higher the weight, the more reliable the estimate.

An example MDT is given in Table 1. The leaf denoted by (*) specifies that the LTree model is to be used to classify an example, if the entropy of the probability distribution returned by it is smaller than 0.38 and the maximum probability in the probability distribution returned by the k-NN model is smaller than 0.75. In sum, if the LTree model is confident in its prediction and the k-NN model is not so confident in its prediction, the leaf recommends using the LTree prediction, which is consistent with common sense in the domain of model combination. The other branch of the tree is based on a much smaller number of examples and is thus much less reliable. It also doesn't make much sense.

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

C4.5 is a greedy divide and conquer algorithm for building classification trees [12]. On 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. The gain criterion is based on the entropy of the class probability distribution of the examples in the current subset S of training examples:

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

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.

When adapting C4.5 for learning meta decision trees, we are interested in the accuracies of each of the individual models $\mathcal{C}_{\mathcal{L}}$ on the examples in S, i.e., the proportion of the examples in S that have a class equal to the corresponding class attribute. The newly introduced measure, used in MLC4.5, is defined as $A = \max_{\mathcal{L} \in LearningAlgorithms} \mathtt{accuracy}(\mathcal{C}_{\mathcal{L}}, S)$, where $\mathtt{accuracy}(\mathcal{C}_{\mathcal{L}}, S)$ denotes the relative frequency of examples in S that are correctly classified by base-level classifier $\mathcal{C}_{\mathcal{L}}$. The vector of accuracies does not have probability distribution properties (its elements do not sum to 1), so the entropy can not be calculated. That is the reason for replacing the entropy based measure with an accuracy based one:

$$\mathit{info}_{\mathcal{ML}}(S) = -\frac{1+A}{2} \cdot \log_2 \frac{1+A}{2} - \frac{1-A}{2} \cdot \log_2 \frac{1-A}{2}.$$

At present, we do not post-prune meta decision trees. The rest of the MLC4.5 algorithm is equivalent to the original C4.5 algorithm for building ordinary classification trees. 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, where $A = \max_{i=1}^{m} p(c_i, S)$, and $\mathit{info}_{\mathcal{A}}$ calculated from A using the same formula as for $\mathit{info}_{\mathcal{ML}}$.

3 Experiments

3.1 Experimental Methodology

In order to evaluate the performance of meta decision trees, we performed experiments on a collection of twenty-one data sets from the UCI Repository of

Machine Learning Databases and Domain Theories [11]. These data sets have been widely used in other comparative studies.

Five learning algorithms were used in the base-level experiments: two tree-learning algorithms C4.5 [12] and LTree [8], the rule-learning algorithm CN2 [4], the k-nearest neighbor (k-NN) algorithm [14] and a modification of the naive Bayes algorithm [9]. All algorithms were used with their default settings.

We used five different algorithms for combining classifiers. Two of them are voting schemes, and three are based on stacking:

P-VOTE is a simple plurality vote algorithm. According to this voting scheme, the example is classified in the class that is most frequently predicted by the base-level classifiers.

CD-VOTE is a refinement of the plurality vote algorithm for the case where class probability distributions are given by the base-level classifiers [5]. 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. The class predicted is the class with the highest class probability in the summed class distribution.

S-MLC4.5 is a stacking algorithm with meta-level classification trees built using MLC4.5.

S-AC4.5 is a stacking algorithm with ordinary classification trees built using AC4.5, a version of C4.5 with the accuracy measure presented in Section 2.

S-C4.5 is a stacking algorithm with ordinary classification trees built with C4.5.

The same set of attributes was used for all three stacking methods. Class attributes are treated as ordinary ones when inducing ordinary decision trees.

The stacking algorithm we use is similar to the standard stacking technique described in [15]. For time complexity reasons, we use stratified 10-fold cross validation for base-level classification instead of the leave-one-out method. By performing 10-fold cross validation, the class attributes in the meta-level training set are obtained by classifying testing examples that were not used for building the base-level classifiers. Another difference from the standard stacking technique is the use of class probability distributions obtained by the base-level classifiers. 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 2). 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.

Classification errors were measured using 10-fold stratified cross validation. Cross validation is repeated 10 times using a different random reordering of the examples in the data set. The same set of re-orderings were used for all experiments. The average and standard deviation (over the ten cross validations) of the classification error on the test examples are reported.

Table 2. Classification errors (in %) of different meta-level classifiers

Data set	P-VOTE	CD-VOTE	S-MLC4.5	S-AC4.5	S-C4.5
australian	13.96 ±0.71	13.81 ±0.57	14.14 ±0.83	15.45 ±1.19	14.90 ±0.68
balance	14.54 ±1.17	12.35 ±0.79	6.74 ±0.52	8.14 ±0.58	7.91 ±1.17
breast-w	3.86 ±0.19	3.49 ±0.15	2.97 ±0.14	3.30 ±0.32	2.94 ±0.28
bridges-td	13.66 ±0.65	14.35 ±0.80	14.09 ±1.16	17.30 ±2.55	17.06 ±2.27
car	6.02 ±0.22	6.13 ±0.27	3.95 ±0.39	3.80 ±0.33	3.17 ±0.40
chess	0.97 ±0.06	0.68 ±0.08	0.57 ±0.10	0.60 ±0.09	0.46 ±0.07
diabetes	23.13 ±0.65	23.41 ±0.59	24.04 ±0.92	25.08 ±1.25	25.53 ±0.91
echo	30.59 ±1.66	30.97 ±1.83	32.60 ±2.61	36.65 ±3.93	35.63 ±2.84
german	25.43 ±0.55	25.21 ±0.59	25.68 ±0.45	26.79 ±1.16	27.50 ±0.89
glass	27.92 ±0.77	26.65 ±1.51	29.35 ±1.65	36.17 ±2.84	35.10 ±2.43
heart	16.19 ±1.17	17.74 ±1.25	17.80 ±1.18	19.43 ±1.96	18.13 ±1.20
hepatitis	17.55 ±1.14	17.99 ±1.47	16.30 ±1.69	18.11 ±1.96	19.98 ±3.47
hypothyroid	1.01 ±0.05	0.97 ±0.04	0.82 ±0.06	0.95 ±0.08	0.75 ±0.06
image	2.49 ±0.18	2.16 ±0.21	2.51 ±0.17	2.81 ±0.22	2.80 ±0.12
ionosphere	8.20 ±0.79	7.98 ±0.66	9.18 ±0.96	9.81 ±1.32	9.66 ±1.44
iris	3.67 ±0.57	4.20 ±0.45	3.14 ±0.83	4.08 ±1.19	3.15 ±0.55
soya	6.96 ±0.31	6.74 ±0.35	6.11 ±0.60	7.11 ±0.84	6.94 ±0.49
tic-tac-toe	10.21 ±0.87	7.95 ±0.84	0.51 ±0.21	0.35 ±0.20	0.11 ±0.16
vote	3.93 ±0.35	3.86 ±0.37	4.12 ±0.50	4.47 ±0.80	4.24 ±0.54
waveform	14.38 ±0.26	14.73 ±0.32	14.30 ±0.15	15.60 ±0.34	16.11 ±0.45
wine	1.36 ±0.60	1.58 ±0.64	2.64 ±0.64	1.64 ±0.49	1.64 ±0.49
Average	**11.72** ±0.62	**11.57** ±0.66	**11.03** ±0.75	**12.27** ±1.13	**12.08** ±1.00

3.2 Experimental Results

Table 2 presents the classification errors of the five meta-level classifiers. The lowest average error is achieved using stacking with meta decision trees. A comparative analysis of the performance of **S-MLC4.5** versus the other four meta-level classifiers and the best base-level classifier (LTree) is given in Table 3.

The figures in Table 3 represent the relative error reduction achieved by using the **S-MLC4.5** algorithm as compared to each of the other algorithms, calculated as $1 - \mathit{s\_mlc4.5\_error} / \mathit{other\_method\_error}$. Positive/negative figures denote better/worse performance of **S-MLC4.5**. The statistical significance of the differences is tested using paired t-tests with significance level of 95%: $+/-$ to the right of a figure in the table means that **S-MLC4.5** is significantly better/worse. The average here is calculated as $1 - \mathtt{GeometricMean}(\mathit{s\_mlc4.5\_error} / \mathit{other\_method\_error})$.

When compared to the best base-level classifier (last column in Table 3), we can clearly see the improvement obtained with meta decision trees used for stacking. Significant improvement is achieved in 14 out of 21 data sets, the overall improvement being 32%.

Stacking with meta decision trees is significantly better than plurality vote in 7 domains and significantly worse in 6. However, the significant improvements are much higher than the significant drops of accuracy, giving overall accuracy improvement of 19%. Since **CD-VOTE** performs slightly better than plurality vote, smaller overall improvement of 16% is achieved. **S-MLC4.5** is significantly better in 9 data sets and significantly worse in 5.

Table 3. Classification error reduction (in %) achieved with stacking using meta decision trees as compared to other meta-level classifiers and the base-level classifier with smallest average error. The + and - signs indicate the significance of the difference.

Data set	**P-VOTE**	**CD-VOTE**	**S-AC4.5**	**S-C4.5**	**LTree**
australian	-1.29	-2.39	8.48 +	5.10 +	-2.24
balance	53.65 +	45.43 +	17.20 +	14.79 +	0.15
breast-w	23.06 +	14.90 +	10.00 +	-1.02	50.00 +
bridges-td	-3.15	1.81	18.55 +	17.41 +	3.76
car	34.39 +	35.56 +	-3.95	-24.61 -	64.67 +
chess	41.24 +	16.18 +	5.00	-23.91 -	26.92 +
diabetes	-3.93 -	-2.69	4.15	5.84 +	4.38 +
echo	-6.57 -	-5.26	11.05 +	8.50 +	8.09 +
german	-0.98	-1.86 -	4.14 +	6.62 +	4.75 +
glass	-5.12 -	-10.13 -	18.86 +	16.38 +	7.30 +
heart	-9.94 -	-0.34	8.39 +	1.82	-1.19
hepatitis	7.12	9.39	9.99 +	18.42 +	12.55 +
hypothyroid	18.81 +	15.46 +	13.68 +	-9.33 -	16.33 +
image	-0.80	-16.20 -	10.68 +	10.36 +	23.01 +
ionosphere	-11.95 -	-15.04 -	6.42	4.97	20.79 +
iris	14.44	25.24 +	23.04	0.32	-0.32
soya	12.21 +	9.35 +	14.06 +	11.96 +	75.03 +
tic-tac-toe	95.00 +	93.58 +	-45.71	-363.64 -	97.16 +
vote	-4.83	-6.74	7.83	2.83	2.14
waveform	0.56	2.92 +	8.33 +	11.24 +	2.32 +
wine	-94.12 -	-67.09 -	-60.98 -	-60.98 -	9.59
Average	**18.99**	**16.33**	**5.82**	**-5.29**	**32.34**

To compare stacking with MDTs and ODTs, we first look at the relative performance of **S-MLC4.5** and **S-C4.5**. **S-MLC4.5** performs significantly better in 11 and worse in 5 data sets. There is a 5% overall decrease of accuracy (this is a geometric mean), but this is entirely due to result in the tic-tac-toe domain, where all stacking methods perform very well. If we exclude the tic-tac-toe domain, a 3% overall increase is observed. We can thus say that **S-MLC4.5** performs slightly better in terms of accuracy. However, the MDTs are much smaller (the size reduction factor being 4, see Table 4), despite the fact that ODTs induced with C4.5 are post-pruned and MDTs are neither pre- nor post-pruned.

To get a clearer picture of the performance differences due to the increased representation power of MDTs as compared to ODTs, we compare **S-MLC4.5** and **S-AC4.5**. 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 stacking. The overall accuracy improvement is only 6%, but **S-MLC4.5** is significantly better than **S-AC4.5** in 13 out of 21 data sets and is significantly worse in only one. Furthermore, the MDTs are, on average, ten times smaller than the ODTs (see Table 4). 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 1 (Section 2.1).

Table 4. Sizes (in number of nodes) of meta decision trees and ordinary decision trees used for stacking

Data set	S-MLC4.5	S-AC4.5	S-C4.5
australian	17.16 ±2.67	88.48 ±3.98	35.00 ±3.04
balance	6.48 ±1.19	121.95 ±3.79	32.89 ±1.78
breast-w	5.82 ±1.30	30.76 ±3.00	6.64 ±2.39
bridges-td	4.72 ±1.06	21.04 ±1.83	7.52 ±1.19
car	27.04 ±3.71	181.20 ±3.93	43.90 ±2.71
chess	11.79 ±2.00	34.66 ±3.67	9.50 ±1.32
diabetes	18.92 ±3.71	123.00 ±4.73	79.20 ±5.36
echo	6.54 ±0.90	59.04 ±3.79	22.54 ±3.20
german	20.76 ±3.88	132.68 ±3.60	101.58 ±5.20
glass	7.44 ±0.74	226.12 ±6.63	49.98 ±4.56
heart	7.54 ±1.73	59.56 ±3.77	18.78 ±3.17
hepatitis	7.08 ±0.73	42.22 ±2.26	14.76 ±1.47
hypothyroid	7.68 ±1.72	40.46 ±4.14	4.50 ±0.92
image	19.44 ±2.37	320.14 ±10.59	63.97 ±3.27
ionosphere	12.48 ±2.07	51.30 ±2.84	19.48 ±2.34
iris	3.62 ±0.63	23.11 ±2.19	5.45 ±0.49
soya	8.38 ±1.06	436.57 ±15.08	81.43 ±11.29
tic-tac-toe	6.04 ±1.13	16.74 ±1.46	7.54 ±0.30
vote	9.62 ±0.93	38.98 ±2.17	8.90 ±1.44
waveform	37.84 ±5.66	479.55 ±8.26	353.31 ±21.65
wine	3.98 ±0.61	13.22 ±1.97	4.60 ±0.35
Average	**11.92** ±1.90	**120.99** ±4.46	**46.26** ±3.69

4 Related Work

An overview of methods for constructing ensembles of classifiers can be found in [5]. Two recent studies are closely related to our work: the SCANN method, based on stacking using correspondence analysis of the classifications of the base-level classifiers [10] and local cascade generalization [7]. The SCANN method outperforms the plurality vote scheme, especially in the case when the base-level classifiers are highly correlated. The SCANN method does not use any class probability distribution properties of the predictions by the base-level classifiers. Therefore, no comparison with the CD voting scheme is included in their study. The set of base-level classifiers used is similar to ours: C4.5 and another algorithm for trees, CN2 for rules, two nearest neighbor algorithms and a naive Bayesian classifier.

In local cascading generalization, the base-level classifiers are used in every node of the decision tree. New attributes, based on the class probability distribution of the example obtained by the base-level classifiers are generated at each step of the divide and conquer algorithm for building decision trees. The base-level classifiers used in this study are naive Bayes and Linear Discriminant. The integration of base-level classifiers is much tighter than in stacking. The similarity to our approach is that class probability distributions are used.

Ordinary decision trees have already been used for combining multiple models in [3]. However, the emphasis of their study is more on partitioning techniques for massive data sets and combining multiple models trained on different subsets of massive data sets. Our study focuses on combining multiple models generated on the same data set. Therefore, the obtained results are not directly comparable to theirs.

The present study is also related to our previous work on the topic of meta-level learning [13]. There we introduced an inductive logic programming (ILP) framework for learning the relation between data set characteristics and the performance of different (base-level) learning algorithms. MDTs use a representation language that is slightly more expressive than propositional decision trees, but much less than ILP.

5 Conclusions and Further Work

We have presented a new technique for combining classifiers based on meta decision trees (MDTs). MDTs increase the expressiveness of propositional decision trees to make them more suitable for stacking. The empirical evaluation of MDTs showed that they outperform ordinary decision trees in terms of accuracy and are much more concise. MDTs are usually so small that they can easily be inspected: we regard this as a step towards a comprehensible model of combining multiple models. In contrast, most existing work uses non-symbolic learning methods (e.g. neural networks) to stack classifiers [10].

There are several obvious directions for further work. For ordinary classification 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 (with the notable exception of Gama). It would be interesting to use other learning algorithms (neural networks, Bayesian classification) to combine models based on the probability distributions returned by them. A comparison of stacking with predictions of models only vs. with predictions and probability distribution properties would be also worthwhile.

The consistency of meta decision trees with common sense model combination knowledge, as briefly discussed in Section 2.1, opens another question for further research. The process of inducing meta-level classifiers should 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.

Note that meta decision trees are, in principle, transferable across domains, 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). MDTs can be also built using examples from data sets originating from different domains. Combining data from different domains for learning MDTs is an 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 [1]. In this case, attributes describing individual data set properties can be added to the class distribution properties in the meta-level learning data set.

Acknowledgements

The work reported was supported in part by the Slovenian Ministry of Science and Technology and by the EU-funded project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise (IST-1999-11495).

References

1. 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.
2. Breiman, L. (1996) Bagging Predictors. *Machine Learning* 24(2): 123–140.
3. 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.
4. 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-Werlag.
5. Dietterich, T. G. (1997) Machine-Learning Research: Four Current Directions. *AI Magazine* 18(4): 97–136.
6. 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.
7. Gama, J. (1998) Combining Classifiers by Constructive Induction. In *Proceedings of the Ninth European Conference on Machine Learning*.
8. Gama, J. (1999) Discriminant trees. In *Proceedings of the Sixteenth International Conference on Machine Learning*: 134-142. Morgan Kaufmann.
9. Gama, J. (2000) A Linear-Bayes Classifier. Technical Report. Artificial Intelligence and Computer Science Laboratory, University of Porto.
10. Merz, C. J. (1999) Using Correspondence Analysis to Combine Classifiers. *Machine Learning* 36(1/2): 33–58. Kluwer Academic Publishers.
11. Murphy, P. M. and Aha, D. W. (1994) *UCI repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.
12. Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
13. 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-Werlag.
14. Wettschereck, D. (1994) *A study of distance-based machine learning algorithms*. PhD Thesis, Department of Computer Science, Oregon State University, Corvallis.
15. Wolpert, D. (1992) Stacked Generalization. *Neural Networks* 5(2): 241–260.