Experiments with Heterogeneous Meta Decision Trees

Bernard Ženko Ljupčo Todorovski
Sašo Džeroski

Jozef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia
Bernard.Zenko, Ljupco.Todorovski, Saso.Dzeroski@ijs.si

Technical Report Jožef Stefan Institute (**IJS-DP 8638**)

June, 2002

1 Introduction

The task of constructing ensembles of classifiers [5] 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., [10]). Another possibility is to apply a single learning algorithm with different parameters settings to a single data set. Finally, methods like bagging [2] and boosting [6] 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 [19] and cascading [7]. 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. We developed an extension of meta decision trees (MDTs) [16] called *heterogeneous MDTs (HMDTs)*. MDTs use properties of the base-level predictions (which are class probability distributions) to decide which of the base level classifier to use for each example; these properties are independent of the data set. The first modification of HMDTs when compared to MDTs is that the model is induced on a union of different data sets instead of just on one. The second modification is that HMDTs use a set of data set properties in order to take into account where each example originates from. A model built on examples described with these two types of attributes is, at least in principle, applicable to arbitrary data set.

The rest of the report is organized as follows. First we describe meta decision trees, which form the groundwork of our approach. Heterogeneous meta decision trees are presented in Section 3. Section 4 presents results of our experiments, while the last section gives some conclusions.

2 Meta Decision Trees

The structure of a meta decision tree is identical to the structure of an ordinary decision tree (ODT). 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 in Table 1. In the meta-level data set M (Table 1a), 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 meta decision tree induced using the meta-level data set M is given in Table 1b). 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 1c) 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 1: The difference between ordinary and meta decision trees.

a) Meta-level data set M

$Conf_1$	C_1	$Conf_2$	C_2	*true class*
0.875	0	0.625	0	0
0.875	0	0.375	1	0
0.875	1	0.375	0	1
0.875	1	0.625	1	1
0.125	0	0.625	0	0
0.125	0	0.625	1	1
0.125	1	0.625	0	0
0.125	1	0.625	1	1

b) The MDT induced from M (by MLC4.5)

```
Conf1 <= 0.125:  C2
Conf1 > 0.125:  C1
```

c) The ODT induced from M (by C4.5)

```
C1 = 0:
|   Conf1 > 0.125 :  0
|   Conf1 <= 0.125 :
|   |   C2 = 0:  0
|   |   C2 = 1:  1
C1 = 1:
|   Conf1 > 0.125 :  1
|   Conf1 <= 0.125 :
|   |   C2 = 0:  0
|   |   C2 = 1:  1
```

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.

We use properties of the class probability distributions predicted by the base-level classifiers as ordinary attributes for inducing MDTs. These properties reflect the certainty and confidence of the predictions. Details about

the set of meta-level attributes are given in the following subsection.

2.1 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:

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

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

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

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 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 the classifier is certain in the predicted class value.

Table 2: A meta decision tree induced in the *image* domain.

```
IBk:Entropy <= 0.002369:  IBk (*)
IBk:Entropy > 0.002369
|    J48:maxProbability <= 0.909091:  IBk
|    J48:maxProbability > 0.909091:  J48
```

An example MDT, induced in the *image* domain from the UCI Repository, is given in Table 2. The leaf denoted by an asterisk `(*)` specifies that the IBk classifier is to be used to classify an example, if the entropy of the class probability distribution predicted by IBk is smaller than or equal to 0.002369. This is consistent with common-sense knowledge in the domain of classifier combination.

Note here another important property of MDTs : 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 in 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 set of meta-level attributes is domain independent. It depends only on the set of base-level classifiers. In sum, there are three reasons for the domain independence of MDTs: (1) the 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.

This domain independence enables induction of MDTs on a union of different domains—heterogeneous MDTs, which are the main topic of this paper.

2.2 MLJ4.8 - a Re-implementation of Meta Decision Trees in Java

In this subsection, we present MLJ4.8, a re-implementation of MLC4.5 [16] algorithm for induction of MDTs in Java. MLJ4.8 is based on J4.8 [18], a re-implementation of Quinlan's C4.5 [13] algorithm for inducing ordinary decision trees. There are several minor implementation differences between J4.8 and C4.5: in such cases we decided to use C4.5 as a reference, rather than J4.8. MLJ4.8 takes as input a meta-level data set consisted of the ordinary and class attributes. The only differences between MLJ4.8 and J4.8 are:

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 MLJ4.8 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. MLJ4.8 does not post-prune the induced MDTs.

The rest of the MLJ4.8 algorithm is identical to the original J4.8 algorithm. Below we describe J4.8's and MLJ4.8's measures for selecting attributes in internal nodes.

J4.8 is a greedy divide and conquer algorithm for building classification trees. 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. The gain criterion is based on the entropy of the class probability distribution of the examples in the current subset S of training examples:

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

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 MLJ4.8, we are interested in the accuracies of each of the base-level classifiers C from set $\mathcal{C}$ on the examples in data set S, i.e., the proportion of examples in S that have a class equal to the class attribute C. The measure, used in MLJ4.8, is defined as:

$$info_{\mathcal{ML}}(S) = 1 - \max_{C \in \mathcal{C}} \texttt{accuracy}(C, S),$$

where $\texttt{accuracy}(C, S)$ 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.

3 Heterogeneous Meta Decision Trees

Heterogeneous Meta Decision Trees (HMDTs) bring together two aspects of meta-level learning. The first aspect is choosing the most suitable machine learning algorithm(s) for a given data set [1]. The second one is combining the predictions obtained from multiple models induced using different machine learning algorithms [5]. HMDTs are (as allready mentioned) MDTs induced from more than one base-level data set in order to combine classifiers (the second aspect of meta-learning) by selecting the appropriate classifier for each individual example from different data sets. Inducing MDTs on more than one data set is possible because the same set of meta-level attributes (maximal class probability (`MaxProbability`) and entropy of the class probability distribution (`Entropy`) described in Section 2.1), summarizing the properties of class probability distributions predicted by the base-level classifiers, are used for different data sets. However, if we induce HMDTs using the same set of meta-level attributes as for MDTs, the selection of the appropriate classifier for an example would be literally independent of the data set where the example originates from.

In order to take the data set where an example originates from into account (the first aspect of meta-learning), HMDTs use additional meta-level attributes that summarize data set properties. Five such properties are used in our experiments:

- number of classes (`NumClasses`),
- number of examples (`NumExamples`),
- number of discrete (`NumNominalAtts`) and
- continuous (`NumNumericAtts`) attributes, and
- number of all attributes (`NumAtts`).

The set of meta-level attributes used in HMDTs is therefore the union of the meta-level attributes connected to each aspect of meta-level learning.

Table 3: A heterogeneous meta decision tree induced from twenty-one data sets using the extended set of meta-level attributes.

```
NaiveBayes:MaxProbability <= 0.936946
|   NumNominalAtts <= 14
|   |   NumAtts <= 5: NaiveBayes:Class
|   |   NumAtts > 5
|   |   |   NumInstances <= 768
|   |   |   |   NumInstances <= 690: J48:Class
|   |   |   |   NumInstances > 690: NaiveBayes:Class
|   |   |   NumInstances > 768
|   |   |   |   IBk:Entropy <= 1.231285
|   |   |   |   |   NumNominalAtts <= 10: IBk:Class
|   |   |   |   |   NumNominalAtts > 10: NaiveBayes:Class
|   |   |   |   IBk:Entropy > 1.231285: J48:Class
|   NumNominalAtts > 14: J48:Class
NaiveBayes:MaxProbability > 0.936946
|   NumClasses <= 3
|   |   NumNominalAtts <= 14: NaiveBayes:Class (*)
|   |   NumNominalAtts > 14: J48:Class
|   NumClasses > 3
|   |   NumNumericAtts <= 4: NaiveBayes:Class
|   |   NumNumericAtts > 4: IBk:Class
```

An example heterogeneous MDT induced from twenty data sets using the extended set of meta-level attributes is given in Figure 3. The leaf denoted by `(*)` specifies that the Naive Bayes model should be used to classify an example if (1) the maximum probability in the class probability distribution predicted by Naive Bayes is larger than 0.936946; (2) the number of classes of the data set is less than or equal to 3; (3) the number of nominal attributes in the data set is less than or equal to 14.

4 Experiments

4.1 Data Sets and Base-Level Experiments

In order to evaluate the performance of meta decision trees, we performed experiments on a collection of twenty-one heterogeneous data sets from the UCI Repository of Machine Learning Databases [11]. These data sets have been widely used in other comparative studies. The data set properties (used also as meta-level attributes, see Section 3) are given in Table 4.

Three learning algorithms were used in the base-level experiments: the tree learning algorithm J4.8, which is a re-implementation of C4.5, the k-nearest neighbor (k-NN or IBk) algorithm and the naive Bayes (NB) algorithm. We used their implementations in the Java programming language incorporated in the Weka data mining suite [18]. J4.8 was used with the default settings. For the IBk algorithm, k was set to 1 and inverse distance weighting was used. The naive Bayes algorithm used kernel density estimation. The output of each base-level classifier for each example consists of the predicted class and the class probability distribution. The classification errors of the base-level classifiers were estimated using 10-fold stratified cross validation. Cross validation is repeated 10 times using a different random seed to get different reordering of the examples in the data set. The same seeds were used for all experiments. The average and standard deviation (over the ten cross validations) of the classification error on unseen examples are presented in Table 5.

4.2 Meta-Level Experiments

In the meta-level experiments, we compared the performances of four different HMDT approaches to the simple plurality vote algorithm:

HMDT. Heterogeneous meta decision trees were induced on the union of all twenty-one data sets. Classification errors were measured using 10-fold stratified cross validation. Each data set was first partitioned into ten

Table 4: Data sets used for evaluation.

Data set	No. of examples	No. of classes	No. of attributes (disc/cont)	all	Max. probability	Class entropy
australian	690	2	(8/6)	14	0.56	0.99
balance	625	3	(0/4)	4	0.46	1.32
breast-w	699	2	(9/0)	9	0.66	0.92
bridges-td	102	2	(4/3)	7	0.85	0.61
car	1728	4	(6/0)	6	0.70	1.21
chess	3196	2	(36/0)	36	0.52	0.99
diabetes	768	2	(0/8)	8	0.65	0.93
echo	131	2	(1/5)	6	0.67	0.91
german	1000	2	(13/7)	20	0.70	0.88
glass	214	6	(0/9)	9	0.36	2.18
heart	270	2	(6/7)	13	0.56	0.99
hepatitis	155	2	(13/6)	19	0.79	0.74
hypo	3163	2	(18/7)	25	0.95	0.29
image	2310	7	(0/19)	19	0.14	2.78
ionosphere	351	2	(0/34)	34	0.64	0.94
iris	150	3	(0/4)	4	0.33	1.58
soya	683	19	(35/0)	35	0.13	3.79
tic-tac-toe	958	2	(9/0)	9	0.65	0.93
vote	435	2	(16/0)	16	0.61	0.96
waveform	5000	3	(0/21)	21	0.34	1.58
wine	178	3	(0/13)	13	0.40	1.56

folds with equal sizes and similar class distributions. The nine folds of all data sets were used for inducing the model, which is then tested on the remaining fold of each data set. Cross validation is repeated 10 times using a different random seed to get different reordering of the examples in the data set. The same seeds were used for all experiments. The average and standard deviation (over the ten cross validations) of the classification error on unseen examples are reported.

HMDT-R. Same as HMDT except that here HMDTs were induced on a meta-data set consisting of 200 examples from each data set only. These

Table 5: Classification errors (in %) of base-level classifiers J4.8, IBk and naive Bayes.

Data set	J4.8	IBk	Naive Bayes
australian	14.54±0.55	13.45±0.29	18.65±0.12
balance	22.43±0.35	9.90±0.20	8.48±0.14
breast-w	5.39±0.20	4.28±0.13	2.69±0.05
bridges-td	14.71±0	16.57±1.11	14.02±0.45
car	7.44±0.25	5.83±0.15	14.40±0.15
chess	0.60±0.05	2.87±0.11	12.16±0.06
diabetes	26.26±0.50	25.55±0.30	24.70±0.12
echo	34.58±2.39	34.73±1.55	27.33±0.44
german	28.82±0.88	26.01±0.51	25.43±0.19
glass	32.24±0.91	29.67±0.86	49.86±0.60
heart	22.19±1.24	18.52±0.48	15.67±0.34
hepatitis	20.90±0.97	17.29±0.63	15.35±0.29
hypo	0.72±0.02	2.79±0.09	1.81±0.02
image	3.18±0.16	2.84±0.07	14.29±0.06
ionosphere	10.26±0.47	13.28±0.33	8.15±0.21
iris	5.33±0.44	4.67±0.40	4.07±0.16
soya	7.54±0.35	8.96±0.26	7.12±0.10
tic-tac-toe	15.11±0.45	0.96±0.06	30.22±0.12
vote	3.54±0.17	6.97±0.22	9.82±0.07
waveform	23.62±0.25	14.42±0.10	19.24±0.05
wine	6.57±1.05	3.26±0.47	2.64±0.17
Average	14.57±0.55	12.52±0.40	15.53±0.19

200 examples were chosen with random sampling from the nine folds mentioned above. In total, the meta-data set consisted of 4200 examples.

HMDT-R-L1O. HMDTs are evaluated using the leave one data set out scheme. The meta-data set consists of 200 examples (selected with random sampling) from each data set except one (4000 examples in total). The model built on this meta-data set is then evaluated on the data set excluded from meta-data set. The procedure is repeated ten times with different reorderings of examples.

HMDT-R-L1O-P. Same as HMDT-R-L1O except that when building HMDT prepruning was used; the minimal number of examples in a leaf was set to 400.

Table 6: Classification errors (in %) of different meta-level experiments.

Data set	HMDT	HMDT-R	HMDT-R-L1O	HMDT-R-L1O-P	Voting
australian	13.62±0.26	16.32±0.64	17.07±0.48	17.14±0.46	13.81±0.34
balance	8.48±0.20	8.56±0.26	15.60±0.96	15.78±0.91	8.91±0.36
breast-w	2.88±0.07	3.13±0.21	4.13±0.44	4.25±0.52	3.46±0.13
bridges-td	15.10±1.18	15.98±1.06	14.90±1.09	14.90±0.70	15.78±0.80
car	5.17±0.21	5.95±0.21	10.89±1.01	11.01±1.40	6.49±0.16
chess	0.61±0.04	0.73±0.14	3.15±1.06	4.75±0.88	1.46±0.08
diabetes	25.39±0.41	25.39±0.64	27.54±0.85	27.72±0.97	24.01±0.33
echo	27.71±0.79	28.17±0.77	29.31±1.02	28.85±1.32	29.24±1.46
german	26.51±0.81	26.50±0.80	27.08±0.57	27.41±0.56	25.19±0.53
glass	32.06±1.27	31.36±0.57	45.70±2.50	48.64±1.23	29.67±0.80
heart	17.19±0.67	17.41±0.65	20.33±1.27	20.37±1.29	17.11±0.76
hepatitis	16.97±0.88	16.77±0.82	17.61±0.94	17.29±0.94	17.42±0.88
hypo	0.75±0.03	1.00±0.17	1.20±0.18	1.74±0.72	1.32±0.02
image	2.82±0.20	3.08±0.17	4.53±0.81	5.36±0.92	2.94±0.14
ionosphere	7.66±0.13	8.15±0.30	8.55±0.35	8.50±1.47	7.18±0.39
iris	4.13±0.27	4.33±0.36	4.27±0.36	4.93±0.87	4.20±0.28
soya	7.23±0.23	7.73±0.40	7.79±0.31	9.46±3.10	6.75±0.18
tic-tac-toe	1.19±0.25	1.53±0.37	23.54±1.49	23.25±3.98	9.24±0.33
vote	3.77±0.32	4.21±0.55	8.32±0.33	9.77±2.65	7.10±0.19
waveform	18.07±0.25	19.06±0.29	22.28±0.49	20.82±3.66	15.90±0.15
wine	3.12±0.46	3.37±0.42	3.56±0.53	3.65±0.82	1.74±0.11
Average	11.45±0.43	11.84±0.47	15.11±0.81	15.50±1.40	11.85±0.40

The voting algorithm was evaluated using 10-fold cross validation repeated ten times.

5 Experimental Results

The classification errors of different meta-level approaches are presented in Table 6. The figures in Table 7 represent the relative error reduction achieved by using the Voting algorithm as compared to each of the other algorithms, calculated as

$$1 - \frac{voting\_error}{other\_method\_error}.$$

Positive/negative figures denote better/worse performance of Voting. The statistical significance of the differences is tested using paired t-tests with

Table 7: Relative improvement in accuracy (in %) of different HMDT approches when compared to voting and its significance (+/– means significantly better/worse, x means insignificant).

	HMDT		HMDT-R		HMDT-R-L1O		HMDT-R-L1O-P	
Data set	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.
australian	-1,38	x	15,36	+	19,10	+	19,44	+
balance	-5,09	x	-4,11	x	42,87	+	43,51	+
breast-w	-20,40	–	-10,50	–	16,26	+	18,52	+
bridges-td	-4,55	x	1,23	x	-5,92	x	-5,92	–
car	-25,50	–	-9,04	–	40,35	+	41,04	+
chess	-138,26	–	-101,29	–	53,67	+	69,24	+
diabetes	5,44	+	5,44	+	12,81	+	13,39	+
echo	-5,51	x	-3,79	x	0,26	x	-1,32	x
german	4,98	x	4,94	+	6,98	+	8,10	+
glass	7,43	+	5,37	+	35,07	+	39,00	+
heart	0,43	x	1,70	x	15,85	+	16,00	+
hepatitis	-2,66	x	-3,85	x	1,10	x	-0,75	x
hypo	-77,54	–	-33,02	–	-9,97	x	23,69	x
image	-3,99	x	4,78	x	35,18	+	45,25	+
ionosphere	6,32	x	11,89	+	16,00	+	15,54	x
iris	-1,61	x	3,08	x	1,56	x	14,82	x
soya	6,68	+	12,69	+	13,35	+	28,64	x
tic-tac-toe	-676,31	–	-502,04	–	60,75	+	60,27	+
vote	-88,41	–	-68,85	–	14,64	+	27,31	x
waveform	12,00	+	16,57	+	28,62	+	23,61	+
wine	44,20	+	48,33	+	51,05	+	52,31	+
Average	-19.53		-10.33		24.21		29.39	
W/L		5+/6–		8+/6–		16+/0–		13+/1–

significance level of 95%: +/− to the right of a figure in the table means that Voting is significantly better/worse. The average here is calculated as

$$1 - \texttt{GeometricMean}(\frac{voting\_error}{other\_method\_error}).$$

6 Discussion and Conclusions

In general, HMDTs do not perform very well. They perform clearly worse than MDTs [15, 16]. But note that they are more generally applicable (across

different datasets) and may not need to be retrained for each new dataset. A HMDT that performs well would mean a new combining scheme, similar to voting, that does not need to be retrained on new datasets and takes dataset properties into account.

HMDT and HMDT-R perform better than voting. In terms of significant wins/losses the difference is slight. The difference in performance (advantage) is clearer in terms of average relative improvement (20% and 10%, respectively).

HMDT-L1O and HMDT-L1O-P perform much worse than voting. They are clearly worse on both the significant wins/losses ratio and the average relative improvement metrics. Pre-pruning (in HMDT-L1O-P) improves matters, but not much. Clearly, the examples from the domain at hand are of crucial importance for training a good HMDT.

This experimental evaluation was relatively limited. We only took into account the five simplest dataset properties. Further studies should take into account other features for describing datasets considered in the METAL project.

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*, pages 148-156. Morgan Kaufmann, San Francisco.

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

[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] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. [http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[12] Pfahringer, B., Bensusan, H. and Giraud-Carrier, C. (2000) Meta-Learning by Landmarking Various Learning Algorithms. To appear in *Proceedings of the Seventeenth International Conference on Machine Learning*.

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

[14] 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.

[15] Todorovski, L. and Džeroski, S. (2000) Meta Decision Trees. Submitted to *ECML2000 Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection*.

[16] Todorovski, L. and Džeroski, S. (2000) Combining multiple models with meta decision trees. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 54–64. Springer, Berlin.

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

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

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