A comparison of stacking with meta decision trees to other combining methods

Bernard Ženko, Ljupčo Todorovski, and Sašo Džeroski
Department of Intelligent Systems, Jožef Stefan Institute
Jamova 39, Ljubljana, Slovenia
{Bernard.Zenko, Ljupco.Todorovski, Saso.Dzeroski}@ijs.si

Abstract

Meta decision trees (MDTs) are a method for combining multiple classifiers. We present an integration of the algorithm MLC4.5 for learning MDTs into the Weka data mining suite. We compare classifier ensembles combined with MDTs to bagged and boosted decision trees, and to classifier ensembles combined with other methods: voting, grading, multi-scheme and stacking with multi-response linear regression.

1 Introduction

The task of constructing ensembles of classifiers [2] 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., [5]). Another possibility is to apply a single learning algorithm with different parameters settings to a single data set. Finally, methods like bagging [1] and boosting [3] generate multiple classifiers 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 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 [11] and cascading [4]. 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.

The rest of the paper is organized as follows. Meta decision trees are described in the next section. Section 3 reports on experimental methodology and briefly describes the meta-level classifiers used in our experiments. The last section presents the results and some conclusions.

2 Meta decision trees

Meta decision trees (MDTs) [9] adopt the stacking framework of combining base-level 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. An example MDT, induced in the *image* domain from the UCI Repository, is given below. 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.

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

The original algorithm MLC4.5 [9] for inducing MDTs was an extension of the C4.5 [6] algorithm for induction of ODTs. We have integrated the algorithm for inducing MDTs in the Weka data mining suite [10]. The algorithm MLJ4.8 is a modification of

Table 1: Classification errors (in %) of base-level classifiers (J4.8, IBk, and naive Bayes) and combining schemes (bagged and boosted decision trees, voting, grading, multi-scheme, stacking with MLR and stacking with MDTs).

Data set	J4.8	IBk	Naive Bayes	Bagging J4.8	Boosting J4.8	Voting	Grading	Multi-scheme	Stacking MLR	Stacking MDT
australian	14.54	13.45	18.65	13.67	15.58	13.81	14.04	13.78	14.16	13.77
balance	22.43	9.90	8.48	17.31	21.49	8.91	8.78	8.51	9.47	8.51
breast-w	5.39	4.28	2.69	4.98	3.71	3.46	3.69	2.69	2.73	2.69
bridges-td	14.71	16.57	14.02	14.90	19.41	15.78	15.10	15.78	14.12	16.08
car	7.44	5.83	14.40	6.78	4.16	6.49	6.10	5.83	5.61	5.02
chess	0.60	2.87	12.16	0.61	0.38	1.46	1.16	0.60	0.60	0.60
diabetes	26.26	25.55	24.70	24.62	28.53	24.01	24.26	25.09	23.78	24.74
echo	34.58	34.73	27.33	31.68	33.89	29.24	30.38	27.63	28.63	27.71
german	28.82	26.01	25.43	26.37	29.23	25.19	25.41	25.69	24.36	25.60
glass	32.24	29.67	49.86	26.03	23.18	29.67	30.75	32.06	30.93	31.78
heart	22.19	18.52	15.67	19.78	21.78	17.11	17.70	16.04	15.30	16.04
hepatitis	20.90	17.29	15.35	17.68	18.26	17.42	18.39	15.87	15.68	15.87
hypo	0.72	2.79	1.81	0.78	1.05	1.32	0.80	0.72	0.72	0.79
image	3.18	2.84	14.29	2.55	1.84	2.94	3.32	2.85	2.84	2.53
ionosphere	10.26	13.28	8.15	7.83	6.41	7.18	8.06	8.40	7.35	8.83
iris	5.33	4.67	4.07	5.73	5.80	4.20	4.40	4.73	4.47	4.73
soya	7.54	8.96	7.12	7.23	7.07	6.75	7.38	7.22	7.22	7.06
tic-tac-toe	15.11	0.96	30.22	6.80	3.43	9.24	6.08	0.96	0.58	0.96
vote	3.54	6.97	9.82	3.93	4.48	7.10	5.22	3.54	3.54	3.54
waveform	23.62	14.42	19.24	18.00	18.58	15.90	17.04	14.42	14.33	14.40
wine	6.57	3.26	2.64	5.11	4.04	1.74	1.80	3.26	2.87	3.26
Average	14.57	12.52	15.53	12.49	12.97	11.85	11.90	11.22	10.92	11.17

J4.8 (the Weka re-implementation of C4.5): the differences between MLJ4.8 and J4.8 closely mirror the ones between MLC4.5 and C4.5. Integrating MDTs into Weka lets us perform a variety of experiments in combining different sets of base-level classifiers, as well as comparisons to other methods for combining classifiers.

3 Experimental setup

In order to compare the performance of MDTs with that of other combining schemes, we perform experiments on a collection of twenty-one data sets from the UCI Repository of Machine Learning Databases and Domain Theories. Three learning algorithms are used in the base-level experiments: the tree learning algorithm J4.8, which is a re-implementation of C4.5 [6], the k-nearest neighbor (k-NN or IBk) algorithm and the naive Bayes (NB) algorithm. In all experiments, classification errors are estimated using 10-fold stratified cross validation. Cross validation is repeated ten times using different random generator seeds resulting in ten different sets of folds.

At the meta-level, the performances of seven algorithms for combining classifiers are compared. These are bagging and boosting of decision trees, voting, grading, multi-scheme, stacking with MLR, and stacking with MDTs. A short description of each of them follows.

Bagging uses random sampling with replacement in order to obtain different versions of a given data set. The size of each sampled data set equals the size of the original data set. On each of these versions of the data set the same learning algorithm, J4.8 in our case, is applied. Classifiers obtained in this manner are then combined with majority voting. For more information see [1].

Boosting first builds a classifier with some learning algorithm (again J4.8 in our case) from the original data set. The weights of the misclassified examples are then increased and another classifier is built using the same learning algorithm. The procedure is repeated several times. Classifiers derived in this manner are then combined using weighted voting. The AdaBoost.M1 variant of boosting was used in our experiments. For more information see [3].

Voting is the simple majority vote algorithm.

Grading is an algorithm that tries to identify and

Table 2: Relative improvement in accuracy (in %) achieved by stacking with MDTs as compared to bagging, boosting, voting, grading, multi-scheme and stacking with MLR; and its significance (+/– : significantly better/worse, x : insignificant).

	Bagging J48		Boosting J48		Voting		Grading		Multi Scheme		Stacking MLR	
Data set	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.	rel. im.	sig.
australian	-0.74	x	11.63	+	0.31	x	1.96	x	0.11	x	2.76	x
balance	50.83	+	60.39	+	4.49	+	3.10	+	0.00	x	10.14	+
breast-w	45.98	+	27.41	+	22.31	+	27.13	+	0.00	x	1.57	x
bridges-td	-7.89	–	17.17	+	-1.86	x	-6.49	x	-1.86	x	-13.89	–
car	25.96	+	-20.75	–	22.73	+	17.74	+	13.99	+	10.62	+
chess	1.55	x	-56.55	–	59.10	+	48.66	+	0.00	x	0.00	x
diabetes	-0.48	x	13.28	+	-3.04	–	-1.99	x	1.40	x	-4.05	–
echo	12.53	+	18.24	+	5.22	+	8.79	+	-0.28	x	3.20	+
german	2.92	+	12.42	+	-1.63	x	-0.75	x	0.35	x	-5.09	–
glass	-22.08	–	-37.10	–	-7.09	–	-3.34	x	0.87	x	-2.72	x
heart	18.91	+	26.36	+	6.28	x	9.41	+	0.00	x	-4.84	x
hepatitis	10.22	x	13.07	+	8.89	+	13.68	+	0.00	x	-1.23	x
hypo	-1.62	x	24.62	+	40.09	+	0.40	x	-9.61	x	-9.61	x
image	0.68	x	-37.65	–	13.72	+	23.63	+	11.23	+	10.82	+
ionosphere	-12.73	–	-37.78	–	-23.02	–	-9.54	x	-5.08	x	-20.16	–
iris	17.44	+	18.39	+	-12.70	–	-7.58	x	0.00	x	-5.97	x
soya	2.43	x	0.21	x	-4.55	–	4.37	+	2.23	x	2.23	x
tic-tac-toe	85.87	+	72.04	+	89.60	+	84.19	+	0.00	x	-64.29	–
vote	9.94	+	21.03	+	50.16	+	32.16	+	0.00	x	0.00	x
waveform	20.00	+	22.50	+	9.44	+	15.50	+	0.11	x	-0.53	x
wine	36.26	+	19.44	x	-87.10	–	-81.25	-	0.00	x	-13.73	x
Average	19.89		14.78		18.34		14.97		0.75		-4.07	
W/L		11+/3–		14+/5–		11+/6–		12+/1–		2+/0–		4+/5–

correct the incorrect predictions of the base-level classifiers. For each base-level classifier, one meta-level classifier is learnt, whose task is to predict when the base-level classifier will make an error. The training set for each of these meta-level classifiers is constructed using the graded (marked correct or incorrect) predictions of the corresponding base-level classifier as new class labels for the original attributes. The final prediction is derived by voting from the predictions of those base-level classifiers that are predicted to be correct by the corresponding meta-level classifier. For more information see [7].

Multi-scheme is an algorithm for selection by cross validation. New examples are classified by the base-level algorithm which has the least cross validation error on training data.

Stacking MLR uses a multi-response linear regression algorithm (MLR) as a meta-level learning algorithm. The meta-level data consist of the class probability distribution for each base-level classifier along with the actual class. For more information see [11] and [8]. MLR transforms a classification problem into a set of regression problems; one problem for each class value. Linear regression is then used to predict the probability of the selected class value. If there are discrete attributes in the data set, they are transformed to binary ones. For more information see [8].

Stacking MDT uses meta decision trees, described in Section 2 and [9], as a meta-level learning algorithm. The meta-level data consist of the maximal class probability and the entropy of the class probability distribution for each base-level classifier (as ordinary attributes) and the classes predicted by base-level classifiers (as class attributes) along with the actual class.

The performance of each of these algorithms is assessed in terms of its error rate. The performance of MDTs is compared to that of the other combining approaches. The relative accuracy improvement of classifier C_1 as compared to classifier C_2 is

$$1 - \texttt{error}(C_1)/\texttt{error}(C_2)$$

(in our case C_1 = MDTs). The average relative improvement is calculated using geometric mean:

$$1 - \texttt{geometric\_mean}(\texttt{error}(C_1)/\texttt{error}(C_2)).$$

The statistical significance of the difference in classification errors is tested using the paired t-test (exactly the same folds are used for C_1 and C_2) with significance level of 95%.

4 Results

The classification errors of all base-level and combining algorithms averaged over ten runs of ten-fold cross validation are presented in Table 1, while the relative improvement in accuracy and its significance are presented in Table 2.

Stacking with MDTs performs better than bagging and boosting of decision trees, which are the state of the art methods for learning ensembles of classifiers: MDTs are significantly better than bagging in 11 and worse in 3 domains with a 20% relative accuracy improvement; when compared to boosting MDTs are significantly better in 14 and worse in 5 domains with a 15% relative accuracy improvement.

A previous study of MDTs [9] shows that MDTs perform better than voting. Our study confirms these findings and proves that they are independent of a specific implementation (we used their re-implementation in Java programming language) and the set of base-level classifiers (we used a different and smaller set).

Grading performs better than bagging and boosting, but is generally not comparable to multi-scheme, stacking with MLR or stacking with MDTs. This is somewhat inconsistent with findings in [7], where the authors report that grading outperforms multi-scheme. However, they performed experiments on different data sets and they used a larger set of base-level classifiers.

The performance of multi-scheme is very similar to the performance of MDTs. The latter are significantly better on two data sets, while all the other differences are insignificant.

Stacking with MLR slightly outperforms stacking with MDTs (significantly better in 5 domains and worse in 4 domains, with a 4% relative improvement in accuracy). Note that stacking with MDTs performs comparably while using less information (only aggregate data on the class probability distribution is used by MDTs, while the complete class probability distribution is used by stacking with MLR). Furthermore, the set of attributes used in MDTs is domain independent once we fix the set of base-level classifiers and the language of MDTs is the same for all domains. Another advantage of the MDTs is their understandability: they provide information about the relative areas of expertise of the base-level classifiers.

Finally, from our experiments we can conclude, that the overall performance of three combining algorithms (multi-scheme, stacking with MLR, and stacking with MDTs) is significantly better than the overall performance of the other tested combining algorithms. The differences in accuracy between the best three algorithms are very small and can hardly be used to choose among them.

Acknowledgements

Many thanks to Alexander Seewald for providing his implementation of grading in Weka.

References

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

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

[3] 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.

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

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

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

[7] Seewald, A. K. and Fürnkranz, J. (2001) An Evaluation of Grading Classifiers. In *Advances in Intelligent Data Analysis: Proceedings of the Fourth International Symposium (IDA-01).* Springer, Berlin.

[8] Ting, K. M. and Witten, I. H. (1999) Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10: 271–289.

[9] 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.

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

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