Stacking with Multi-Response Model Trees

Sašo Džeroski and Bernard Ženko

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

Abstract. We empirically evaluate several state-of-the-art methods for constructing ensembles of classifiers with stacking and show that they perform (at best) comparably to selecting the best classifier from the ensemble by cross validation. We then propose a new method for stacking, that uses multi-response model trees at the meta-level, and show that it outperforms existing stacking approaches, as well as selecting the best classifier from the ensemble by cross validation.

1 Introduction

An ensemble of classifiers is a set of classifiers whose individual predictions are combined in some way (typically by voting) to classify new examples. One of the most active areas of research in supervised learning has been to study methods for constructing good ensembles of classifiers [3]. The attraction that this topic exerts on machine learning researchers is based on the premise that ensembles are often much more accurate than the individual classifiers that make them up.

Most of the research on classifier ensembles is concerned with generating ensembles by using a single learning algorithm [4], such as decision tree learning or neural network training. Different classifiers are generated by manipulating the training set (as done in boosting or bagging), manipulating the input features, manipulating the output targets or injecting randomness in the learning algorithm. The generated classifiers are then typically combined by voting or weighted voting.

Another approach is to generate classifiers by applying different learning algorithms (with heterogeneous model representations) to a single data set (see, e.g., [9]). More complicated methods for combining classifiers are typically used in this setting. Stacking [18] is often used to learn a combining method in addition to the ensemble of classifiers. Voting is then used as a baseline method for combining classifiers against which the learned combiners are compared. Typically, much better performance is achieved by stacking as compared to voting.

The work presented in this paper is set in the stacking framework. We argue that selecting the best of the classifiers in an ensemble generated by applying different learning algorithms should be considered as a baseline to which the stacking performance should be compared. Our empirical evaluation of several recent stacking approaches shows that they perform comparably to the best of the individual classifiers as selected by cross validation, but not better. We then

propose a new stacking method, based on classification by using model trees, and show that this method does perform better than other combining approaches, as well as better than selecting the best individual classifier.

Section 2 first summarizes the stacking framework, then surveys some recent results and finally introduces our stacking approach based on classification via model trees. The setup for the experimental comparison of several stacking methods, voting and selecting the best classifier is described in Section 3. Section 4 presents and discusses the experimental results and Section 5 concludes.

2 Stacking with Model Trees

We first give a brief introduction to the stacking framework, introduced by [18]. We then summarize the results of several recent studies in stacking [9, 13, 14, 12, 15]. Motivated by these, we introduce a stacking approach based on classification via model trees [5].

2.1 The Stacking Framework

Stacking is concerned with combining multiple classifiers generated by using different learning algorithms $L_1, \ldots, L_N$ on a single data set S, which consists of examples $s_i = (x_i, y_i)$, i.e., pairs of feature vectors (x_i) and their classifications (y_i). In the first phase, a set of base-level classifiers $C_1, C_2, \ldots C_N$ is generated, where $C_i = L_i(S)$. In the second phase, a meta-level classifier is learned that combines the outputs of the base-level classifiers.

To generate a training set for learning the meta-level classifier, a leave-one-out or a cross validation procedure is applied. For leave-one-out, we apply each of the base-level learning algorithms to almost the entire data set, leaving one example for testing: $C_k^i = L_k(S - s_i)$. We then use the learned classifiers to generate predictions for s_i: $\hat{y}_i^k = C_k^i(x_i)$. The meta-level data set consists of examples of the form $((\hat{y}_i^1, \ldots, \hat{y}_i^N), y_i)$, where the features are the predictions of the base-level classifiers and the class is the correct class of the example at hand.

In contrast to stacking, no learning takes place at the meta-level when combining classifiers by a voting scheme (such as plurality, probabilistic or weighted voting). 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.

2.2 Recent Advances

The most important issues in stacking are probably the choice of the features and the algorithm for learning at the meta-level. Below we review some recent research on stacking that addresses the above issues.

It is common knowledge that ensembles of diverse base-level classifiers (with weakly correlated predictions) yield good performance. [9] proposes a stacking method called SCANN that uses correspondence analysis do detect correlations between the predictions of base-level classifiers. The original meta-level feature space (the class-value predictions) is transformed to remove the dependencies, and a nearest neighbor method is used as the meta-level classifier on this new feature space.

[13] use base-level classifiers whose predictions are probability distributions over the set of class values, rather than single class values. The meta-level attributes are thus the probabilities of each of the class values returned by each of the base-level classifiers. The authors argue that this allows to use not only the predictions, but also the confidence of the base-level classifiers. Multi-response linear regression (MLR) is recommended for meta-level learning, while several learning algorithms are shown not to be suitable for this task.

[12] propose a method for combining classifiers called grading that learns a meta-level classifier for each base-level classifier. The meta-level classifier predicts whether the base-level classifier is to be trusted (i.e., whether its prediction will be correct). The base-level attributes are used also as meta-level attributes, while the meta-level class values are + (correct) and − (incorrect). Only the base-level classifiers that are predicted to be correct are taken and their predictions combined by summing up the probability distributions predicted.

[14] introduce a new meta-level learning method for combining classifiers with stacking: meta decision trees (MDTs) have base-level classifiers in the leaves, instead of class-value predictions. Properties of the probability distributions predicted by the base-level classifiers (such as entropy and maximum probability) are used as meta-level attributes, rather than the distributions themselves. These properties reflect the confidence of the base-level classifiers and give rise to very small MDTs, which can (at least in principle) be inspected and interpreted.

[15] report that stacking with MDTs clearly outperforms voting and stacking with decision trees, as well as boosting and bagging of decision trees. On the other hand, MDTs perform only slightly better than SCANN and selecting the best classifier with cross validation (SelectBest). [19] report that MDTs perform slightly worse as compared to stacking with MLR. Overall, SCANN, MDTs, stacking with MLR and SelectBest seem to perform at about the same level.

It would seem natural to expect that ensembles of classifiers induced by stacking would perform better than the best individual base-level classifier: otherwise the extra work of learning a meta-level classifier doesn't seem justified. The experimental results mentioned above, however, do not show clear evidence of this. This has motivated us to investigate the performance of state-of-the-art stacking methods in comparison to SelectBest and seek new stacking methods that would be clearly superior to SelectBest.

2.3 Stacking with Multi-Response Model Trees

We assume that each base-level classifier 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:

$$\mathbf{p}^C(x) = \left(p^C(c_1|x), p^C(c_2|x), \ldots p^C(c_m|x)\right),$$

where $\{c_1, c_2, \ldots c_m\}$ is the 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 meta-level attributes are thus the probabilities predicted for each possible class by each of the base-level classifiers, i.e., $p^{C_j}(c_i|x)$ for $i = 1, \ldots, m$ and $j = 1, \ldots, N$.

The experimental evidence mentioned above indicates that although SCANN, MDTs, stacking with MLR and SelectBest seem to perform at about the same level, stacking with MLR has a slight advantage over the other methods. It would thus seem as a suitable starting point in the search for better method for meta-level learning to be used in stacking. Stacking with MLR uses linear regression to perform classification. A natural direction to look into is the use of model trees (which perform piece-wise linear regression) instead of MLR: model trees have namely been shown to perform better than MLR for classification via regression [5].

MLR is an adaptation of linear regression. For a classification problem with m class values $\{c_1, c_2, \ldots c_m\}$, m regression problems are formulated: for problem j, a linear equation LR_j is constructed to predict a binary variable which has value one if the class value is c_j and zero otherwise. Given a new example x to classify, $LR_j(x)$ is calculated for all j, and the class k is predicted with maximum $LR_k(x)$.

In our approach, we use model tree induction instead of linear regression and keep everything else the same. Instead of m linear equations LR_j, we induce m model trees MT_j. M5′ [16], a re-implementation of M5 [10] included in the data mining suite Weka [17] is used to induce the trees. Given a new example x to classify, $MT_j(x)$ is calculated for all j, and the class k is predicted with maximum $MT_k(x)$. We call our approach stacking with multi-response model trees and denoted with SMM5 in the tables with experimental results.

3 Experimental Setup

In the experiments, we investigate the following issues:

- The (relative) performance of existing state-of-the-art stacking methods, especially in comparison to SelectBest.
- The performance of stacking with multi-response model trees relative to the above methods.
- The influence of the number of base-level classifiers on the (relative) performance of the above methods.

We look into the last topic because the recent studies mentioned above use different numbers of base-level classifiers, ranging from three to eight.

The Weka data mining suite [17] was used for all experiments, within which all the base-level and meta-level learning algorithms used in the experiments have been implemented.

3.1 Data Sets

In order to evaluate the performance of the different combining algorithms, we perform experiments on a collection of twenty-one data sets from the *UCI Repository of machine learning databases* [2]. These data sets have been widely used in other comparative studies.

3.2 Base-Level Algorithms

We perform two batches of experiments: one with three and one with seven base-level learners. The set of three contains the following algorithms:
- J4.8: a Java re-implementation of the decision tree learning algorithm C4.5 [11],
- IBk: the k-nearest neighbor algorithm of [1], and
- NB: the naive Bayes algorithm of [7].

The second set of algorithms contains, in addition to the above three, also the following four algorithms:
- K*: an instance-based algorithm which uses an entropic distance measure [6],
- KDE: a simple kernel density estimation algorithm,
- DT: the decision table majority algorithm of [8],
- MLR: the multi-response linear regression algorithm, as used by [13] and described in Section 2.3.

All algorithms are used with their default parameter settings, with the exceptions described below. IBk in the set of three learners uses inverse distance weighting and k was selected with cross validation from the range of 1 to 77. (IBk in the set of seven learners uses the default parameter values, i.e., no weighting and $k = 1$.) The NB algorithm in both sets uses the kernel density estimator rather than assume normal distributions for numeric attributes.

3.3 Meta-Level Algorithms

At the meta-level, we evaluate the performance of six different schemes for combining classifiers (listed below), each applied with the two different sets of base-level algorithms described above.
- Vote: The simple plurality vote scheme (see Section 2.1),
- Selb: The SelectBest scheme selects the best of the base-level classifiers by cross validation.
- Grad: Grading as introduced by [12] and briefly described in Section 2.2.
- Smdt: Stacking with meta decision-trees as introduced by [14] and briefly described in Section 2.2.
- Smlr: Stacking with multiple-response regression as used by [13] and described in Sections 2.2 and 2.3.
- Smm5: Stacking with multiple-response model trees, as proposed by this paper and described in Section 2.3.

Table 1. The relative performance of 3-classifier ensembles with different combining methods. The entry in row X and column Y gives the relative improvement of X over Y in % and the number of wins/loses.

	Vote	Selb	Grad	Smdt	Smlr	Smm5	Total
Vote		-21.53 7+/10–	-4.12 6+/5–	-22.45 6+/11–	-27.43 5+/11–	-47.06 2+/10–	26+/47–
Selb	17.72 10+/7–		14.33 11+/3–	-0.76 0+/2–	-4.85 2+/5–	-21.00 1+/9–	24+/26–
Grad	3.96 5+/6–	-16.72 3+/11–		-17.60 1+/12–	-22.39 2+/14–	-41.24 1+/13–	12+/56–
Smdt	18.34 11+/6–	0.75 2+/0–	14.97 12+/1–		-4.07 4+/5–	-20.10 2+/8–	31+/20–
Smlr	21.53 11+/5–	4.63 5+/2–	18.29 14+/2–	3.91 5+/4–		-15.40 1+/7–	36+/20–
Smm5	32.00 10+/2–	17.36 9+/1–	29.20 13+/1–	16.73 8+/2–	13.35 7+/1–		47+/7–

Table 2. The relative performance of 7-classifier ensembles with different combining methods. The entry in row X and column Y gives the relative improvement of X over Y in % and the number of wins/loses.

	Vote	Selb	Grad	Smdt	Smlr	Smm5	Total
Vote		-19.21 5+/12–	-6.73 2+/7–	-18.04 4+/9–	-24.40 2+/10–	-42.04 0+/10–	13+/48–
Selb	16.10 12+/5–		10.46 11+/4–	0.97 3+/3–	-4.37 5+/7–	-19.17 2+/7–	33+/26–
Grad	6.30 7+/2–	-11.68 4+/11–		-10.60 5+/7–	-16.56 2+/12–	-33.09 0+/12–	18+/44–
Smdt	15.29 9+/4–	-0.97 3+/3–	9.59 7+/5–		-5.39 5+/6–	-20.33 0+/11–	24+/29–
Smlr	19.62 10+/2–	4.19 7+/5–	14.21 12+/2–	5.11 6+/5–		-14.18 1+/5–	36+/19–
Smm5	29.60 10+/0–	16.08 7+/2–	24.86 12+/0–	16.89 11+/0–	12.42 5+/1–		45+/3–

3.4 Evaluating and Comparing Algorithms

In all the experiments presented here, classification errors are estimated using ten-fold stratified cross validation. Cross validation is repeated ten times using different random generator seeds resulting in ten different sets of folds. The same folds (random generator seeds) are used in all experiments. The classification error of a classification algorithm C for a given data set as estimated by averaging over the ten runs of ten-fold cross validation is denoted $\mathsf{error}(C)$.

For pair-wise comparisons of classification algorithms, we calculate 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: $1-\mathsf{error}(C_1)/\mathsf{error}(C_2)$. The average relative improvement across all domains is calculated using the geometric mean of error reduction in individual domains: $1-\mathsf{geometric\_mean}(\mathsf{error}(C_1)/\mathsf{error}(C_2))$. Note that this may be different from $\mathsf{geometric\_mean}(\mathsf{error}(C_2)/\mathsf{error}(C_1))-1$.

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 ($\mathsf{error}(C_1)$ and $\mathsf{error}(C_2)$ 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.

4 Experimental Results

The error rates of the 3-classifier and 7-classifier ensembles induced as described above on the twenty-one data set and combined with the different combining methods are given in Table 3. However, for the purpose of comparing the performance of different combining methods, Tables 1 and 2 are of much more interest: they give the average relative improvement of X over Y for each pair of combining methods X and Y, as well as the number of significant wins/losses. Below we highlight some of our more interesting findings.

4.1 State-of-the-Art Stacking Methods

Inspecting Tables 1 and 2, we find that we can partition the five combining algorithms (we do not consider Smm5 at this stage of the analysis) into three groups. Vote and Grad are at the lower end of the performance scale, Selb and Smdt are in the middle, while Smlr performs best. While Smlr clearly outperforms Vote and Grad, the advantage over Selb is slim (3 and 2 more wins than losses, about 4% relative improvement) and the advantage over Smdt even slimmer (1 more win than loss in both cases, 4 and 5% of relative improvement).

4.2 Stacking with Multi-Response Model Trees

Returning to Tables 1 and 2, this time paying attention to the relative performance of Smm5 to the other combining methods, we find that Smm5 is in a

league of its own. It clearly outperforms all the other combining methods, with a wins − loss difference of at least 4 and a relative improvement of at least 10%. The difference is smallest when compared to SMLR.

4.3 The Influence of the Number of Base-Level Classifiers

Studying the differences between Tables 1 and 2, we can note that the relative performance of the different combining methods is not affected too much by the change of the number of base-level classifiers. GRAD and SMDT seem to be affected most. The relative performance of GRAD improves, while that of SMDT worsens, when we go from 3 to 7 base-level classifiers: GRAD becomes better than VOTE, while SMDT becomes ever-so-slightly worse than SELB. SMM5 and SMLR are clearly the best in both cases.

5 Conclusions and Further Work

We have empirically evaluated several state-of-the-art methods for constructing ensembles of classifiers with stacking and shown that they perform (at best) comparably to selecting the best classifier from the ensemble by cross validation. We have proposed a new method for stacking, that uses multi-response model trees at the meta-level. We have shown that it clearly outperforms existing stacking approaches and selecting the best classifier from the ensemble by cross validation.

While this study clearly shows good performance of our method on standard UCI domains, it would be instructive to perform the same experiments on real applicative domains. Another issue to investigate is the influence of the parameters of the meta-level learner (M5') on overall performance. While conducting this study and a few other recent studies [19, 15], we have encountered quite a few contradictions between claims in the recent literature on stacking and our experimental results e.g., [9, 13, 12]. A comparative study including the data sets used in these papers and a few other stacking methods (such as SCANN) should resolve these contradictions and provide a clearer picture of who's who in stacking. We believe this is a worthwhile topic to pursue in near-term future work. We also believe that further research on stacking in the context of base-level classifiers created by different learning algorithms is in order, despite the current focus of the machine learning community on creating ensembles with a single learning algorithm with injected randomness or its application to manipulated training sets, input features and output targets. This should include the pursuit for better sets of meta-level features and better meta-level learning algorithms.

Acknowledgements

This work was supported in part by the METAL project (ESPRIT Framework IV LTR Grant Nr. 26.357). Many thanks to Ljupčo Todorovski for the cooperation on combining classifiers with meta-decision trees and the many interesting and stimulating discussions related to this paper. Thanks also to Alexander Seewald for providing his implementation of grading in Weka.

Table 3. Error rates (in %) of the learned ensembles of classifiers.

	3 base level classifiers						7 base level classifiers					
Data set	Vote	Selb	Grad	Smdt	Smlr	Smm5	Vote	Selb	Grad	Smdt	Smlr	Smm5
australian	13.81	13.78	14.04	13.77	14.16	14.29	13.99	14.84	14.46	15.06	13.97	14.26
balance	8.91	8.51	8.78	8.51	9.47	4.37	10.14	8.48	10.02	8.45	10.51	4.99
breast-w	3.46	2.69	3.69	2.69	2.73	2.82	3.65	2.69	3.65	2.69	2.72	2.70
bridges-td	15.78	15.78	15.10	16.08	14.12	14.61	15.39	16.47	15.39	17.45	15.59	15.69
car	6.49	5.83	6.10	5.02	5.61	1.52	6.73	5.69	5.32	3.72	4.24	1.38
chess	1.46	0.60	1.16	0.60	0.60	0.60	1.59	0.60	1.20	0.60	0.60	0.62
diabetes	24.01	25.09	24.26	24.74	23.78	24.10	24.10	23.11	24.38	24.27	23.70	24.05
echo	29.24	27.63	30.38	27.71	28.63	27.63	30.92	28.63	30.92	30.61	29.54	30.23
german	25.19	25.69	25.41	25.60	24.36	24.97	24.08	24.67	24.39	24.29	23.20	23.25
glass	29.67	32.06	30.75	31.78	30.93	31.26	25.79	25.19	26.54	25.56	24.63	25.05
heart	17.11	16.04	17.70	16.04	15.30	15.67	17.26	16.15	17.48	16.63	16.04	15.85
hepatitis	17.42	15.87	18.39	15.87	15.68	14.97	16.39	16.06	17.23	16.71	16.84	16.13
hypo	1.32	0.72	0.80	0.79	0.72	0.76	1.56	0.76	1.05	1.35	0.77	0.78
image	2.94	2.85	3.32	2.53	2.84	2.84	1.92	3.03	1.98	2.47	2.02	2.05
ionosphere	7.18	8.40	8.06	8.83	7.35	6.55	8.52	8.43	8.60	8.80	7.12	7.89
iris	4.20	4.73	4.40	4.73	4.47	4.47	5.00	4.40	4.87	4.40	4.93	5.20
soya	6.75	7.22	7.38	7.06	7.22	6.65	6.71	6.22	6.33	6.34	7.36	6.37
tic-tac-toe	9.24	0.96	6.08	0.96	0.58	0.26	3.58	0.96	2.46	0.96	0.64	0.27
vote	7.10	3.54	5.22	3.54	3.54	3.36	6.25	3.93	5.20	3.93	3.75	3.79
waveform	15.90	14.42	17.04	14.40	14.33	13.69	16.64	14.04	16.78	13.85	15.65	13.48
wine	1.74	3.26	1.80	3.26	2.87	3.03	1.46	2.30	1.46	2.19	2.08	2.02
Average	11.85	11.22	11.90	11.17	10.92	10.40	11.51	10.79	11.41	10.97	10.76	10.29

References

1. D. Aha, D.W. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
3. T. G. Dietterich. Machine-learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.
4. T. G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, Berlin, 2000. Springer.
5. E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.
6. G. C. John and E. T. Leonard. K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning*, pages 108–114, San Francisco, 1995. Morgan Kaufmann.
7. G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Francisco, 1995. Morgan Kaufmann.
8. R. Kohavi. The power of decision tables. In *Proceedings of the Eighth European Conference on Machine Learning*, pages 174–189, 1995.
9. C. J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1/2):33–58, 1999.
10. J. R. Quinlan. Learning with continuous classes. In *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.
11. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.
12. A. K. Seewald and J. Fürnkranz. An evaluation of grading classifiers. In *Advances in Intelligent Data Analysis: Proceedings of the Fourth International Symposium (IDA-01)*, pages 221–232, Berlin, 2001. Springer.
13. K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
14. L. Todorovski and S. Džeroski. 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, Berlin, 2000. Springer.
15. L. Todorovski and S. Džeroski. Combining classifiers with meta decision trees. *Machine Learning*, In press, 2002.
16. Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning*, Prague, 1997. University of Economics, Faculty of Informatics and Statistics.
17. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.
18. D. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.
19. B. Ženko, L. Todorovski, and S. Džeroski. A comparison of stacking with mdts to bagging, boosting, and other stacking methods. In *Proceedings of the First IEEE International Conference on Data Mining*, pages 669–670, Los Alamitos, 2001. IEEE Computer Society.