Towards a Theoretical Framework for Ensemble Classification

Alexander K. Seewald
Austrian Research Institute for Artificial Intelligence
Freyung 6/VI/7, A-1010 Wien, Austria
alexsee@oefai.at; alex@seewald.at

Abstract

Ensemble learning schemes such as AdaBoost and Bagging enhance the performance of a single classifier by combining predictions from multiple classifiers of the same type. The predictions from an ensemble of diverse classifiers can be combined in related ways, e.g. by voting or simply by selecting the best classifier via cross-validation - a technique widely used in machine learning. However, since no ensemble scheme is always the best choice, a deeper insight into the structure of meaningful approaches to combine predictions is needed to achieve further progress. In this paper we offer an operational reformulation of common ensemble learning schemes - Voting, *Selection by Crossvalidation* (X-Val), Grading and Bagging - as a Stacking scheme with appropriate parameter settings. Thus, from a theoretical point of view all these schemes can be reduced to Stacking with an appropriate combination method. This result is an important step towards a general theoretical framework for the field of ensemble learning.

1 Introduction

We apologize that space restrictions arc forcing us to be terse. For a more detailed version, see [Seewald, 2003]. We will first explain how Stacking works in order to lay the foundations for our functional definitions of meta classifiers later on. Fig. 1 shows Stacking on a hypothetical dataset.

During training, all base classifiers arc evaluated via cross-validation on the original dataset. Each classifier's output is a class probability distribution for every example, see Fig. 1(b) The concatenated class probability distributions of all base classifiers, followed by the class value, forms the meta-level training set for Stacking's meta classifier, see Fig. 1(c) After training the meta classifier, the base classifiers are retrained on the complete training data.

For testing, the base classifiers are queried for their class probability distributions. These form a meta-example for the meta classifier which outputs the final class prediction.

2 Definitions

An arbitrary training dataset with *n* examples and *k* classes, and a single test instance, is given. *N* arbitrary base classifiers have been cross-validated on this dataset, and afterwards retrained on the whole dataset. All base classifiers output class probability distributions, i.e. estimated probabilities for each class instead of deciding on a single class.

Then, P_{ijk} refers to the probability given by classifier *i* for class *j* on example number *k* during CV. $\overline{P}_{ik}$ is the class probability distribution of classifier *i* on instance *k*. $\overline{P}_i$ refers to

Attrs	Cl
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	c
$AttrVec_n$	a

(a) original training set

a	b	c
0.90	0.05	0.05
0.15	0.70	0.15
0.20	0.20	0.60
0.80	0.10	0.10

(b) sample class prob dist

$Classifier_1$				$Classifier_N$			
a	b	c		a	b	c	Cl.
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$		$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	a
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$		$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	b
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$		$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	c
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	...	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	a

(c) training set for Stacking's meta classifier

Figure 1: Illustration of Stacking on a dataset with three classes *(a, b* and c), *n* examples and TV base classifiers. $P_{i,j,k}$ refers to the probability given by classifier *i* for class *j* on example number *k*.

the class prob.dist. for classifier *i* on the unknown test instance. A fixed arbitrary order on the class values and *N* base classifiers is assumed. *Classk* is the true class for instance *k*. $AttrVec_k$ is the attribute vector of instance k. n is the number of instances. All indices are zero-based, e.g. the instance id *k* satisfies the equation $0 \leq k \leq n - 1$.

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} \quad (1)$$

$$\arg\max_i (A_i) \equiv \min\{i \mid A_i = \max_{\forall i} (A_i)\} \quad (2)$$

As we mentioned, we assume that all ensemble learning schemes return class probability distributions. If predictions are needed, the position of the maximum class probability in the vector - i.e. the predicted class - is easily obtained via formula (2) Trivially, Stacking with predictions can also be simulated by this variant; simply by transforming the class distributions meta-dataset to predictions via (2) prior to applying the meta classifier.

We can now characterize every ensemble learning scheme by what features it extracts from the meta-dataset during training and how these features define the mapping from meta-dataset to final class probability distribution.

2.1 Voting

Voting, a straight-forward extension of voting for distribution classifiers, is the simplest case. During training, no features are extracted from the meta-dataset. In fact Voting does not even need the internal crossvalidation. During testing, Voting determines the final class probability distribution as follows.

$$\overline{pred} = \sum_{i=1}^{N} \frac{\overline{P}_i}{N} \quad (3)$$

Thus, it can be easily seen that the meta-classifier defined by just computing the mean probability distribution of the base classifiers - as above - simulates Voting with probability distributions. Voting with predictions can be mapped similarity. In this case, we use $\overline{P'}_i$ instead of $\overline{P}_i$ in formula (3) $\overline{P'}_i$ is the vector of p., for all j, where

$$P'_{ij} = \begin{cases} 1 & \text{if } j = \arg\max_j (P_{ij}), \text{ for given } i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

2.2 X-Val

For X-Val, which chooses the best base classifier on each fold by an internal ten-fold CV, we first determine the accuracy per classifier, which can be computed directly from the meta-level dataset, see (6) Then, we derive as feature the id of the classifier with the maximum accuracy. Thus, the value of *bestC* corresponds to our learned model.

$$bestC = \arg\max_i (Acc_i) \quad (5)$$

$$Acc_i = \frac{1}{n} \sum_{k=1}^{n} \delta(\arg\max_j (P_{ijk}), Class_k) \quad (6)$$

During testing, X-Val simply returns $bestC$'s prediction.

$$\overline{pred} = \overline{P}_{bestC} \quad (7)$$

2.3 Grading

For Grading [Seewald and Furnkranz, 2001], the case is difficult. While we cannot simulate the original Grading, we will simulate a competitive variant equivalent to accuracy-weighted voting. More details see [Seewald, 2003].

During training, the accuracies of base classifiers are extracted using formula (6) The accuracies of all our base classifiers correspond to our learned model. During testing, we build the combined class probability distribution similar to Voting using predictions but with an additional weight - namely the accuracy we extracted earlier.

$$\overline{pred} = \frac{1}{\sum_{i=1}^{N} Acc_i} \sum_{i=1}^{N} \left[Acc_i \frac{\overline{P'}_i}{N} \right] \quad (8)$$

where $\overline{P'}_i$ is again the vector of P'_{ij} for all j.

2.4 Bagging

Bagging [Breiman, 1996] is another common ensemble technique. Here, the same type of classifier is repeatedly trained on new datasets, which have been generated from the original dataset via random sampling with replacement. Afterwards, the component classifiers are combined via simple unweighted voting.

Thus, we use same meta-classifier as for Voting with predictions. The number of base classifiers is equal to the iteration parameter of bagging - each base classifier for Stacking corresponds to one instantiation of the base learner for bagging. In order to simulate the random sampling from the training set, the base learner's training sets have to be modified before training, via formula (9)

$$nTS = \{[AttrVec_{i_j}, Class_{i_j}] | i_j = rand(0, n-1), for\ \forall\ 0 \le j \le n-1\} \quad (9)$$

During training, Formula (9) is used to create - for each base classifier separately - a training set of the same size as the original training set via random sampling from the original training set, exactly as in Bagging. These training sets are then used to train the base classifiers. This approach can also be seen as a probabilistic wrapper around each base classifier. No features are extracted from the meta-level dataset during training, as for Voting.

During testing, each base classifier gives a prediction. These predictions are then voted to yield the final prediction, exactly as for Voting with predictions, i.e. (3) modified via (4) - for more details refer to subsection 2.1. Concluding, we have shown the equivalence of Bagging and Stacking.

3 Discussion & Conclusion

By definition StackingC [Seewald, 2002], can also be mapped onto Stacking via a special meta classifier. In fact, the available implementation is a specialized subclass of a common meta classifier, MLR. Another recent variant, sMM5 [Dzeroski and Zenko, 2002], is also implemented via a special meta classifier and thus amenable to the same kind of mapping. However, AdaBoost [Freund and Schapire, 1996] cannot be simulated by Stacking because of its sequential nature.

While the given formal definitions of meta classifiers are mainly intended to enable further theoretical work, a direct implementation is also feasible. The cost penalty for the simulation is small, since training the meta classifiers usually contributes little to the total runtime.

Concluding, we have shown that Stacking is equivalent to common ensemble learning schemes, namely *Selection by Crossvalidation* (X-Val), Voting of either class probability distributions or predictions, a competitive variant of Grading, and Bagging. Recent variants such as StackingC [Seewald, 2002] and sMM5 [Dzeroski and Zenko, 2002] arc also equivalent. Thus we conclude that our approach offers a unique viewpoint on Stacking which is an important step towards a theoretical framework for ensemble learning.

Acknowledgements This research is supported by the Austrian *Fonds zur Forderung der Wissenschafdichen Forschung (FWF)* under grant no. P12645-INF. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture.

References

[Breiman, 1996] Breiman, L. (1996): Bagging Predictors. Machine Learning *(24),* pp. 123-140.

[Dzeroski and Zenko, 2002] Dzeroski S., Zenko B. (2002): Is Combining Classifiers Better than Selecting the Best One?, in Proceedings of the 19th International Conference on Machine Learning, 1CML-2002, Morgan Kaufmann Publishers, San Francisco, 2002.

[Freund and Schapire, 1996] Freund, Y., Schapire R.E. (1996): Experiments with a new boosting algorithm, Proceedings of the International Conference on Machine Learning (ICML-96), pages 148-156, Morgan Kaufmann.

[Seewald and Furnkranz, 2001] Seewald A.K., Furnkranz J. (2001): An Evaluation of Grading Classifiers, in Hoffmann F. et al. (eds.), Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001, Proceedings, Springer, Berlin, pp. 115-124,2001.

[Seewald, 2002] Seewald A.K. (2002): How to make Stacking Better and Faster While Also Taking Care of an Unknown Weakness, in Proceedings of the 19th International Conference on Machine Learning, ICML-2002, Morgan Kaufmann Publishers, San Francisco, 2002.

[Seewald, 2003] Seewald A.K. (2003): Towards a Theoretical Framework for Ensemble Classification (extended version), Technical Report, Austrian Research Institute for Artificial Intelligence, Vienna, TR-2003-08.

[Ting and Witten, 1999] Ting, K. M., Wittcn, 1. H. (1999): Issues in stacked generalization. Journal of Artificial Intelligence Research *10* (1999) 271-289.

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