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

In this paper, we show that the ensemble learning scheme Stacking [1] is universal in the sense that other common ensemble learning schemes systems such as Voting, *Selection by Crossvalidation* (X-Val), Grading and even Bagging can be mapped onto Stacking via distinct meta classifiers. We present operational definitions of these meta classifiers for each scheme. In each case, running Stacking with the appropriate meta classifier simulates the respective ensemble learning scheme's operation perfectly, although – as with every simulation – the runtime performance may be worse.

For Grading [Seewald and Fürnkranz, 2001] which would ordinarily not be mappable, we show that an alternative parameter setting can transform it into a mappable form without sacrificing its unique performance.

Finally we show that, by additionally modifying the training set of the base classifiers with a simple wrapper, Bagging [Breiman, 1996] can also be simulated. Thus Stacking can be seen as a general theoretical framework for these common ensemble learning schemes.

2 How does Stacking work?

We shall now explain how Stacking works in order to lay the foundations for our functional definitions of meta classifiers later on. Figure 1 shows Stacking on a hypothetical dataset with three classes, n examples and N base classifiers. Figure 1(a) shows the original dataset. Each example consists of an attribute vector of fixed length, followed by a class value.

During training, all base classifiers are evaluated via crossvalidation on the original dataset. Basically, a crossvalidation splits the dataset into J equal-sized folds, then uses $J-1$ folds for training and the remaining fold for testing. This process is repeated J times so that each fold is used for testing exactly once, thus generating one prediction for every example[2]. We follow the extension of [Ting and Witten, 1999] of using the base classifiers' class probabilities. Each classifier's output is therefore a class probability distribution for every example. Figure 1(b) shows how such a reasonable class probability distribution from a single classifier may look. The rows correspond to the examples from the original training set, see Figure 1(a).

The concatenated class probability distributions of all base classifiers in a fixed order, followed by the class value, forms the meta-level training set for Stacking's meta classifier, see Figure 1(c). After training the meta classifier, the base classifiers are retrained on the complete training data.[3] Thus, Stacking's meta classifier is free to learn arbitrary complex models to predict the true class from the class probabilities of its base classifiers, making it clearly the most flexible ensemble learning scheme.

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. Any meta classifier for Stacking must therefore learn a mapping from a vector of concatenated class probabilities to a predicted class value from training data, and later apply this learned mapping to a new vector.

[1] introduced in [Wolpert, 1992], extended by [Ting and Witten, 1999].

[2] $J = 10$ throughout this paper

[3] Interestingly, *not* retraining the base classifiers usually yields slightly worse results.

$Attrs$	$Cl.$
$AttrVec_1$	a
$AttrVec_2$	b
$AttrVec_3$	b
$AttrVec_4$	c
$\vdots$	$\vdots$
$AttrVec_n$	a

(a) original training set

a	b	c
0.90	0.05	0.05
0.15	*0.70*	0.15
0.10	*0.80*	0.10
0.20	0.20	*0.60*
$\vdots$	$\vdots$	$\vdots$
0.80	0.10	0.10

(b) sample class probability distribution

$Classifier_1$			$Classifier_2$				$Classifier_N$			
a	b	c	a	b	c		a	b	c	$class$
$P_{1,a1}$	$P_{1,b1}$	$P_{1,c1}$	$P_{2,a1}$	$P_{2,b1}$	$P_{2,c1}$	$\dots$	$P_{N,a1}$	$P_{N,b1}$	$P_{N,c1}$	a
$P_{1,a2}$	$P_{1,b2}$	$P_{1,c2}$	$P_{2,a2}$	$P_{2,b2}$	$P_{2,c2}$	$\dots$	$P_{N,a2}$	$P_{N,b2}$	$P_{N,c2}$	b
$P_{1,a3}$	$P_{1,b3}$	$P_{1,c3}$	$P_{2,a3}$	$P_{2,b3}$	$P_{2,c3}$	$\dots$	$P_{N,a3}$	$P_{N,b3}$	$P_{N,c3}$	b
$P_{1,a4}$	$P_{1,b4}$	$P_{1,c4}$	$P_{2,a4}$	$P_{2,b4}$	$P_{2,c4}$	$\dots$	$P_{N,a4}$	$P_{N,b4}$	$P_{N,c4}$	c
	$\vdots$			$\vdots$				$\vdots$		$\vdots$
$P_{1,an}$	$P_{1,bn}$	$P_{1,cn}$	$P_{2,an}$	$P_{2,bn}$	$P_{2,cn}$	$\dots$	$P_{N,an}$	$P_{N,bn}$	$P_{N,cn}$	a

(c) training set for Stacking's meta classifi er

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

2.1 Definitions

In this section we shall give some definitions for important concepts and terms. Without loss of generality let us assume that a fixed training dataset with n examples and k classes, and a single test instance[4], is given. N arbitrary base classifiers have already been cross-validated on this dataset, each yielding a prediction for all n examples, and have afterwards been retrained on the complete training dataset. We also assume that 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 the internal cross-validation. Let us now denote $\overline{P}_{ik}$ to signify the complete class probability distribution of classifier i on instance k. If no k is given, $\overline{P}_i$ refers to the class probability distribution for classifier i on the unknown instance during testing. For completeness we also have to assume a fixed arbitrary order on the class values so that each class is at the same position in all $\overline{P}_{ik}$ considered, and a fixed arbitrary order on the N base classifiers. $Class_k$ denotes the true class for instance k from the training set. $AttrVec_k$ corresponds to the attribute vector of instance k. n is the number of instances. We consider all indices to be zero-based, e.g. the instance id k satisfies the equation $0 \leq k \leq n-1$.

$\delta(x, y)$ is the well-known delta function (1). In case the notation is not generally known, we also define $\arg\max$ (2) to signify the first entry where the corresponding value is equal to the maximum. This allows to determine the predicted class from a given class probability distribution and also takes care of non-unique maximal values[5].

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

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

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, we can thus also simulate Stacking with predictions by Stacking with probability distributions; 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 during testing. Thus, each meta classifier which simulates the corresponding ensemble learning scheme can be defined by two characteristics: One, the features which are extracted from the meta-dataset and how they are computed. Two, how these features are used during testing to determine the final class probability distribution.

3 Mapping Ensemble Learning Schemes

We show that Stacking using class probability distributions is equivalent to the following ensemble learning schemes, given appropriate meta classifiers.

- Stacking is the stacking algorithm as implemented in WEKA, which follows [Ting and Witten, 1999]. It constructs the meta dataset by adding the entire predicted

[4] Since during testing, no learning takes place – i.e. the learned model of all classifi ers remains constant – instances can be processed in arbitrary order. Thus, demonstrating equivalence for a single test instance is suffi cient.

[5] In that case, choosing the more common class is also a reasonable alternative

class probability distribution instead of only the most likely class. We show that all other ensemble learning schemes can be mapped onto this variant of Stacking via specialized meta classifiers. Trivially, Stacking via predictions can also be simulated by transforming the class distributions meta-dataset to predictions prior to applying the meta classifier via formula (2).

- X-Val chooses the best base classifier on each fold by an internal ten-fold CV. This is also known as *Selection by Crossvalidation*, a widely used ensemble technique in machine learning.
- Voting is a straight-forward extension of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to the base classifier's estimate of the class probability distribution for the example. I.e. the mean class distribution of all classifiers is calculated. It is the only scheme which does not use an internal crossvalidation. We also show that more common voting of predictions can be simulated by Stacking.
- Grading is the grading algorithm [Seewald and Fürnkranz, 2001]. Basically, Grading trains one meta classifier for each base classifier which tries to predict when its associated base classifier fails. This decision is based on the original attributes from the dataset. A weighted voting of the base classifiers prediction gives the final class prediction. The confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier, is used as weight.
- 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.

We shall proceed to show how every one of them can be simulated by Stacking with an appropriate meta classifier. We give functional descriptions of the mapping from meta-dataset to features during training and from features to final prediction during testing. For Bagging, a simple wrapper is necessary around each base classifier, which simulates random sampling with replacement.

3.1 Voting

Voting is the simplest case. During training, no features are extracted from the meta-dataset. In fact Voting does not even need the internal crossvalidation. Since after training the base classifiers are retrained on the complete training set, the base classifiers are then equivalent to those normally used in Voting.

During testing, Voting determines the final class probability distribution as follows, i.e. as mean class probability distribution for the current unknown instance.

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

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 similarily. In this case, we use $\overline{P'}_i$ instead of $\overline{P}_i$ in formula (3). $\overline{P'}_i$ is the vector of P'_{ij} for all j, where

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

In essence, this simplifies the class probability distribution to a vector of zeros with just a one where the most probable class was earlier. Summing over these simplified class probability distributions is clearly equivalent to counting the number of votes per class over all classifiers. Again, the class with the highest number of votes is chosen as final prediction. Concluding, we have shown Stacking to be equivalent to Voting in either variant, using the proposed meta-classifier.

3.2 X-Val

For X-Val, we first determine the accuracy per classifiers as estimated by the internal crossvalidation, 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.

$$bestC = \arg\max_i (Acc_i) \tag{5}$$

where

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

During testing, X-Val simply returns the distribution from best base classifier $bestC$.

$$\overline{pred} = \overline{P}_{bestC} \tag{7}$$

3.3 Grading

For Grading, the case is quite difficult. Since Grading's meta classifiers[6] base their model on the original dataset's attributes, at first glance it seems to be impossible to map it onto Stacking as specialized meta-classifier – at least without utilizing bi-level stacking [Schaffer, 1994].

However, during an evaluation of Grading we noted that there is very little difference between meta classifiers, always less than 1%, see Table 1. This was also found by the original authors, see the technical report [Seewald and Fürnkranz, 2001]. This puzzled us for some time and eventually prompted us to run our own experiments using a baseline learner as meta classifier which always outputs a fixed probability distribution based only on the most common class. The idea was to find out how much of the performance gain is due to the combining scheme and thus independent of the meta-classifier.

[6] Note that both Stacking and Grading have the parameter *meta-classifier*.

We were surprised to note that this trivial meta classifier, ZeroR, is competitive to all other meta classifiers we evaluated and even once outperforms them all, see Table 1.[7] So it is a reasonable alternative meta classifier to IBk with ten nearest neighbors for Grading, which was used in [Seewald and Fürnkranz, 2001] and we propose it for further experiments.

What does this alternate meta classifier mean for Grading? Basically, Grading does not *grade* – it works solely because reasonable meta classifiers will be as good as the baseline accuracy, while getting better than that seems to be extremely hard. Based on Grading's combining scheme, the predictions are weighted by $p(+)$, so in this setting Grading is essentially equivalent to accuracy-weighted voting of the base classifiers predictions, where the accuracy is estimated via cross-validation.[8] Thus, while we cannot simulate the original Grading with multiple meta-classifiers via Stacking, we have at least shown that we can simulate a variant similar to Grading, which is competitive to the original Grading proposal and five other variants with different meta-classifiers – even once outperforming them all.

Given our proposed new meta classifier, it is now possible to map Grading onto Stacking. 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.

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

where $\overline{P'}_i$ is again the vector of P'_{ij} for all j. P'_{ij} is taken from formula (4).

A straightforward extension of this which we have not yet evaluated is accuracy-weighted voting of base classifier's class probability distributions, which is given in the following formula.

$$\overline{pred} = \frac{1}{\sum_{i=1}^{N} Acc_i} \sum_{i=1}^{N} \left[Acc_i \frac{\overline{P}_i}{N} \right] \tag{9}$$

Thus, we have shown that Stacking can simulate Grading without sacrificing its unique performance.

[7] By average accuracy it is even the best choice – however, a more realistic view is to consider it competitive to the original choice.

[8] The meta-datasets for Grading are different for each base classifier and consist of the original attributes, followed by a binary attribute which encodes whether the base classifier did (+) or did not (-) correctly predict the class of the respective instance during the cross-validation. The class distribution of this meta-dataset is directly related to the cross-validated accuracy of its base learner, i.e. $p(+) = Acc$ and $p(-) = 1 - Acc$ – so the better the learner performs, the more unbiased the class distribution becomes. Under these circumstances it is not unreasonable to expect this to be a very hard learning task, which can seldom be solved better than the baseline of always predicting $+$. Based on the reasonable assumption that all classifiers have an accuracy of at least 50%, the most common class will be $+$ and then Grading will be equivalent to accuracy-weighted voting.

3.4 Bagging

For Bagging, the same meta-classifier as for Voting with predictions is used. 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 (10).

$$newTrainSet = \{[AttrVec_{i_j}, Class_{i_j}] | i_j = rand(0, n-1), for\ \forall\ 0 \leq j \leq n-1\} \tag{10}$$

rand(a,b) ... generates uniform random numbers from the closed interval [a,b]

During training, Formula (10) 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 3.1. Concluding, we have shown the equivalence of Bagging and Stacking.

4 Others

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. Therefore, Stacking is also equivalent to both of these new variants, given appropriate meta classifiers.

However, AdaBoost [Freund and Schapire, 1996] cannot be simulated by Stacking because of its mainly sequential nature.[9]

[9] Or, to be more precise: While its training phase could potentially be simulated, using bi-level Stacking [Schaffer, 1994], replacing internal cross-validation estimates with training set performance estimates and utilizing multi-level vertical stacking – i.e. putting each classifier on top of the last one: one level for each iteration – and some elaborate wrappers between adjacent levels, its testing phase can regrettably *not* be simulated. AdaBoost computes a weighted vote of its component classifiers, whereas in Stacking the predictions of the classifiers are propagated upwards, beginning at the lowest level, and are thus processed by each component classifier in turn. So, in order to simulate AdaBoost, we would have to either modify the component classifiers as well, or change the basic structure of Stacking. We opted to leave this problem open until a more complete understanding of combining methods offers a simpler, less opaque and more helpful approach.

Table 1: Grading with different level 1 classifiers. The first column shows the accuracy of IBk(originally used in [Seewald and Fuernkranz, 2001]), all other columns show accuracy ratios for the respective meta-learners ($\frac{Acc(\mathsf{Meta}_i)}{Acc(\mathsf{IBk})}$). The last column shows the results for our baseline learner, ZeroR. Average accuracy and standard deviation per column are also shown.

Dataset	IBk	DecisionTable	J48	KernelDensity	KStar	MLR	NaiveBayes	ZeroR
audiology	84.51	0.9895	0.9895	0.9791	1.0105	0.9895	1.0052	1.0000
autos	81.95	1.0060	1.0060	0.9643	1.0000	0.9643	1.0060	1.0298
balance-scale	89.76	1.0000	0.9982	1.0000	1.0018	1.0000	1.0000	1.0000
breast-cancer	74.48	1.0000	0.9859	1.0047	1.0094	1.0094	1.0000	0.9906
breast-w	96.57	1.0000	1.0015	0.9970	1.0000	0.9985	1.0015	0.9984
colic	84.78	0.9936	1.0032	0.9968	1.0032	0.9840	0.9968	1.0000
credit-a	86.09	0.9916	0.9933	0.9949	0.9966	0.9848	0.9916	0.8816
credit-g	75.90	0.9934	0.9895	1.0000	0.9829	0.9789	0.9881	1.1343
diabetes	76.30	1.0085	1.0051	1.0051	1.0085	1.0034	1.0017	1.0085
glass	73.36	1.0191	1.0000	1.0255	1.0255	1.0382	1.0127	1.0319
heart-c	85.48	0.9730	0.9691	0.9730	0.9807	0.9653	0.9846	0.9884
heart-h	83.33	1.0122	1.0122	0.9755	0.9837	1.0041	1.0041	1.0082
heart-statlog	83.70	0.9956	0.9823	0.9602	0.9735	0.9823	0.9867	0.9956
hepatitis	81.94	1.0079	1.0236	1.0315	1.0157	1.0157	1.0472	1.0315
ionosphere	91.17	1.0063	1.0063	1.0000	1.0000	1.0063	1.0094	1.0093
iris	95.33	1.0000	1.0000	1.0070	1.0000	1.0070	1.0000	1.0000
labor	94.74	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
lymph	84.46	1.0160	1.0240	0.9840	1.0000	1.0240	1.0080	1.0320
primary-tumor	49.26	0.9641	0.9222	0.8922	0.9641	0.9641	1.0000	0.9582
segment	98.10	1.0000	0.9987	0.9969	0.9951	0.9969	0.9965	1.0000
sonar	86.06	0.9721	0.9441	0.9888	0.9777	0.9665	0.9832	0.9720
soybean	94.00	0.9969	1.0000	1.0000	0.9751	0.9984	1.0031	0.9984
vehicle	74.47	1.0254	1.0111	0.9889	0.9810	1.0333	1.0254	1.0191
vote	96.09	0.9976	1.0000	0.9976	0.9976	0.9952	0.9952	1.0000
vowel	98.79	0.9969	1.0010	0.9969	0.9980	0.9939	0.9980	0.9908
zoo	97.03	1.0000	1.0000	1.0102	0.9898	0.9898	1.0102	1.0000
Avg	85.29	0.9987	0.9949	0.9912	0.9950	0.9959	1.0021	1.0030
$\pm$	10.81	0.0137	0.0217	0.0258	0.0143	0.0196	0.0130	0.0396

5 Experimental Issues

While the given formal definitions of meta classifiers are mainly intended to enable further theoretical work, a direct implementation is also feasible. On the assumption that our models are correct, an implementation could serve to investigate runtime performance, i.e. the cost penalty for the simulation. However, since training the meta classifier usually contributes little to the total runtime – the main cost is due to the internal cross-validation of all base classifiers – we consider this unnecessary. In most cases, the expected runtime cost of the simulation is expected to be comparable to that of the original system. In case of Grading it is even expected to be slightly less, since our proposed meta classifier is much faster than the original classifier proposed in [Seewald and Fürnkranz, 2001]. Only for Voting, which does not need the internal cross-validation at all, the runtime cost of the simulation is expected to be about one order of magnitude higher. We believe that the advantage of having a comprehensive description of all these schemes within a single framework outweighs this cost penalty for Voting.

6 Related Research

To our knowledge, there is no related research concerned with the theoretical equivalence or practical simulation of ensemble learning schemes. The conceptual closeness of most ensemble learning schemes is of course no surprise; however, we seem to have been the first to formalize this closeness towards achieving a general theoretical framework.

7 Conclusion

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, and a competitive variant of Grading. We have given functional descriptions of suitable meta classifiers for Stacking which simulate the operation of these ensemble learning schemes. By a simple wrapper we were also able to simulate Bagging. Recent variants such as StackingC[Seewald, 2002] and sMM5 [Dzeroski and Zenko, 2002] can also be simulated in the same way. So all these schemes can essentially be reduced to Stacking with an appropriate combining scheme. Thus we conclude that our approach offers a unique viewpoint on Stacking which is an important step towards a theoretical framework for ensemble learning.

One possible venue for future research may be to build tailored meta classifiers for specific problems, using the definitions of common ensemble learning schemes as background knowledge to guide the search process. Research into alternative meta classifiers for Stacking seems also a reasonable course, given that two recent variants (StackingC, sMM5) have been successful in this area using quite simple approaches.

Acknowledgements

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

References

[Breiman, 1996] Breiman, L. (1996): Bagging Predictors. Machine Learning *(24)*, 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, ICML-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, pages 148-156, Morgan Kaufmann, San Francisco.

[Schaffer, 1994] Schaffer, C. (1994): Cross-validation, stacking and bi-level stacking: Meta-methods for classification learning. In P. Cheeseman and R. W. Oldford (Eds.), *Selecting models from data: Artificial Intelligence and Statistics IV*, 51–59. Springer-Verlag.

[Seewald and Fürnkranz, 2001] Seewald A.K., Fürnkranz 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/Heidelberg/New York/Tokyo, pp.115-124, 2001. Also available as Technical Report (earlier version) TR-2001-01, Austrian Research Institute for Artificial Intelligence, Vienna, Austria. `www.oefai.at`

[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.

[Ting and Witten, 1999] Ting, K. M., Witten, I. 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.