Stacked Generalization: when does it work?

Kai Ming Ting and Ian H. Witten
Department of Computer Science
University of Waikato
Hamilton, New Zealand
{kaiming,ihw}@cs.waikato.ac.nz

Abstract

Stacked generalization is a general method of using a high-level model to combine lower-level models to achieve greater predictive accuracy. In this paper we address two crucial issues which have been considered to be a 'black art' in classification tasks ever since the introduction of stacked generalization in 1992 by Wolpert: the type of generalizer that is suitable to derive the higher-level model, and the kind of attributes that should be used as its input. We demonstrate the effectiveness of stacked generalization for combining three different types of learning algorithms.

1 Introduction

Stacked generalization is a way of combining multiple models that have been learned for a classification task [Wolpert, 1992]. The first step is to collect the output of each model into a new set of data. For each instance in the original training set, this data set represents every model's prediction of that instance's class, along with its true classification. During this step, care is taken to ensure that the models are formed from a batch of training data that does not include the instance in question, in just the same way as ordinary cross-validation. The new data is treated as the data for another learning problem, and in the second step a learning algorithm is employed to solve this problem. In Wolpert's terminology, the original data and the models constructed for it in the first step are referred to as *level-0 data* and *level-0 models*, respectively, while the set of cross-validated data and the second-stage learning algorithm are referred to as *level-1 data* and the *level-1 generalizer*.

In this paper, we show how to make stacked generalization work for classification tasks by addressing two crucial issues which Wolpert [1992] originally described as 'black art' and have not been resolved since. The two issues are (i) the type of attributes that should be used to form level-1 data, and (ii) the type of level-1 generalizer in order to get improved accuracy using the stacked generalization method.

Breiman [1996a] demonstrated the success of stacked generalization in the setting of ordinary regression. The level-0 models are regression trees of different sizes or linear regressions using different number of variables. But instead of selecting the single model that works best as judged by (for example) cross-validation, Breiman used the different level-0 regressors' output values for each member of the training set to form level-1 data. Then he used least-squares linear regression, under the constraint that all regression coefficients be non-negative, as the level-1 generalizer. The non-negativity constraint turned out to be crucial to guarantee that the predictive accuracy would be better than that achieved by selecting the single best predictor.

Here we show how stacked generalization can be made to work reliably in classification tasks. We do this by using the output class probabilities generated by level-0 models to form level-1 data. Then, for the level-1 generalizer we use a version of least squares linear regression adapted for classification tasks. We find the use of class probabilities to be crucial for the successful application of stacked generalization in classification tasks. However, the non-negativity constraints found necessary by Breiman in regression are irrelevant to improved predictive accuracy in our classification situation.

In Section 2, we formally introduce the technique of stacked generalization and describe pertinent details of each learning algorithm used in our experiments. Section 3 describes the results of stacking three different types of learning algorithms. The following section describes related work, and is followed by a summary of our conclusions and future work.

2 Stacked Generalization

Given a data set $\mathcal{L} = \{(y_n, x_n), n = 1, \ldots, N\}$, where y_n is the class value and x_n represents the attribute values of the nth instance, randomly split the data into J almost equal parts $\mathcal{L}_1, \ldots, \mathcal{L}_J$. Define $\mathcal{L}_j$ and $\mathcal{L}^{(-j)} = \mathcal{L} - \mathcal{L}_j$ to be the test and training sets for the jth fold of a J-fold cross-validation. Given K learning algorithms, which we call *level-0 generalizers*, invoke the kth algorithm on the data in the training set $\mathcal{L}^{(-j)}$ to induce a model $\mathcal{M}_k^{(-j)}$, for $k = 1, \ldots, K$. These are called *level-0 models.*

For each instance x in $\mathcal{L}_j$, the test set for the jth cross-validation fold, let $v_k^{(-j)}(x)$ denote the prediction of the model $\mathcal{M}_k^{(-j)}$ on x. Let

$$z_{kn} = v_k^{(-j)}(x_n).$$

At the end of the entire cross-validation process, the data assembled from the outputs of the K models is

$$\mathcal{L}_{CV} = \{(y_n, z_{1n}, \ldots, z_{Kn}), n = 1, \ldots, N\}.$$

This is the *level-1 data.* Use some learning algorithm that we call the *level-1 generalizer* to derive from this data a model $\tilde{\mathcal{M}}$. This is the *level-1 model.* To complete the training process, models $\mathcal{M}_k$, $k = 1, \ldots, K$, are derived using all the data in $\mathcal{L}$.

Now let us consider the classification process, which uses the models $\mathcal{M}_k$, $k = 1, \ldots, K$, in conjunction with $\tilde{\mathcal{M}}$. Given a new instance, models $\mathcal{M}_k$ produce a vector $(z_1, \ldots, z_K)$. This vector is input to the level-1 model $\tilde{\mathcal{M}}$, whose output is the final classification result for that instance. This completes the stacked generalization method as proposed by Wolpert [1992], and also used by Breiman [1996a] and LeBlanc and Tibshirani [1993].

As well as the situation described above, which results in the level-1 model $\tilde{\mathcal{M}}$, the present paper also considers a further situation where the output from the level-0 models is a set of class probabilities rather than a single class prediction. If model $\mathcal{M}_k^{(-j)}$ is used to classify an instance x in $\mathcal{L}_j$, let $P_{ki}^{(-j)}(x)$ denote the probability of the ith output class, and write

$$z_{kin} = P_{ki}^{(-j)}(x_n).$$

As the level-1 data, assemble together the class probability vector from all K models, along with the original class:

$$\mathcal{L}'_{CV} = \{(y_n, z_{11n}, .., z_{1In}, \ldots, z_{k1n}, .., z_{kIn}, \\ \ldots, z_{K1n}, .., z_{KIn}), n = 1, \ldots, N\}$$

(assuming there are I classes). Denote the level-1 model derived from this as $\tilde{\mathcal{M}}'$ to contrast it with $\tilde{\mathcal{M}}$. The rest of the procedure is the same as the original one.

The following two subsections describe the algorithms used as level-0 and level-1 generalizers in the experiments reported in Section 3.

2.1 Level-0 Generalizers

Three learning algorithms are used as the level-0 generalizers: C4.5, a decision tree learning algorithm [Quinlan, 1993]; NB, a re-implementation of a Naive Bayesian classifier [Cestnik, 1990]; and IB1, a variant of a lazy learning algorithm [Aha *et al.*, 1991] which employs the p-nearest-neighbor method using a modified value-difference metric for nominal and binary attributes [Cost and Salzberg, 1993]. For each of these learning algorithms we now show the formula that we use for the estimated output class probabilities $P_i(x)$ for an instance x (where, in all cases, $\sum_i P_i(x) = 1$).

C4.5: Consider the leaf of the decision tree at which the instance x falls. Let m_i be the number of (training) instances with class i at this leaf, and suppose the majority class at the leaf is $\hat{I}$. Let $E = \sum_{i \neq \hat{I}} m_i$. Then, using the Laplace estimator,

$$P_{\hat{I}}(x) = 1 - \frac{E+1}{\sum_i m_i + 2},$$

$$P_i(x) = (1 - P_{\hat{I}}(x)) \times \frac{m_i}{E}, \text{ for } i \neq \hat{I}.$$

NB: Let $P(i|x)$ be the posterior probability of class i, given instance x. Then

$$P_i(x) = \frac{P(i|x)}{\sum_i P(i|x)}.$$

IB1: Suppose p nearest neighbors are used; denote them by $\{(y_s, x_s), s = 1, \ldots, p\}$ for instance x. (We use $p = 3$ in the experiments.) Then

$$P_i(x) = \frac{\sum_{s=1}^{p} f(y_s)/d(x, x_s)}{\sum_{s=1}^{p} 1/d(x, x_s)},$$

where $f(y_s) = 1$ if $i = y_s$ and 0 otherwise, and d is the Euclidean distance function.

In all three learning algorithms, the predicted class of the level-0 model, given an instance x, is that $\hat{I}$ for which $P_{\hat{I}}(x) > P_i(x)$ for all $i \neq \hat{I}$.

2.2 Level-1 Generalizers

We compare the effect of four different learning algorithms as the level-1 generalizer: C4.5, IB1(using $p = 21$ nearest neighbors),[1] NB, and a multi-response linear regression algorithm, MLR. Only the last needs further explanation.

MLR is an adaptation of a least-squares linear regression algorithm that Breiman [1996a] used in regression

[1] A large p value is used following Wolpert's [1992] advice that "...it is reasonable that 'relatively global, smooth ...' level-1 generalizers should perform well."

settings. Any classification problem with real-valued attributes can be transformed into a multi-response regression problem. If the original classification problem has I classes, it is converted into I separate regression problems, where the problem for class ℓ has instances with responses equal to one when they have class ℓ and zero otherwise.

The input to MLR is level-1 data, and we need to consider the situation for the model $\tilde{\mathcal{M}}'$, where the attributes are probabilities, separately from that for the model $\tilde{\mathcal{M}}$, where they are classes. In the former case, where the attributes are already real-valued, the linear regression for class ℓ is simply

$$LR_\ell(x) = \sum_k^K \sum_i^I \alpha_{ki\ell} P_{ki}(x).$$

In the latter case, the classes are unordered nominal attributes. We map them into binary values in the obvious way, setting $P_{ki}(x)$ to 1 if the class of instance x is ℓ and zero otherwise; and then use the above linear regression.

Choose the linear regression coefficients $\{\alpha_{ki\ell}\}$ to minimize

$$\sum_j \sum_{(y_n,x_n)\in\mathcal{L}_j} (y_n - \sum_k \sum_i \alpha_{ki\ell} P_{ki}^{(-j)}(x_n))^2.$$

The coefficients $\{\alpha_{ki\ell}\}$ are constrained to be non-negative, following Breiman's [1996a] discovery that this is necessary for the successful application of stacked generalization to regression problems. The non-negative-coefficient least-squares algorithm described by Lawson and Hanson [1995] is employed here to derive the linear regression for each class. We show later that, in fact, the non-negative constraint is unnecessary in classification tasks.

With this in place, we can now describe the working of MLR. To classify a new instance x, compute $LR_\ell(x)$ for all I classes and assign the instance to that class ℓ which has the greatest value: $LR_\ell(x) > LR_{\ell'}(x)$ for all $\ell' \neq \ell$.

3 Stacking C4.5, NB and IB1

3.1 When does stacking work?

The experiments in this section show that

- for successful stacked generalization it is necessary to use output class probabilities rather than class predictions—that is, $\tilde{\mathcal{M}}'$ rather than $\tilde{\mathcal{M}}$;
- only MLR is suitable for the level-1 generalizer.

We use two artificial datasets and eight real-world datasets from the UCI Repository of machine learning databases [Merz and Murphy, 1996]. Details of these are given in Table 1.

For the artificial datasets—Led24 and Waveform—each training dataset $\mathcal{L}$ is generated using a different seed. The algorithms used for the experiments are then tested on a separate dataset of 5000 instances. Results are expressed as the average error rate of ten repetitions of this entire procedure.

Table 1: Details of the datasets used in the experiment.

Datasets	# Samples	# Classes	# Attr & Type
Led24	200–5000	10	10N
Waveform	300–5000	3	40C
Horse	368	2	3B+12N+7C
Credit	690	2	4B+5N+6C
Vowel	990	11	10C
Euthyroid	3163	2	18B+7C
Splice	3177	3	60N
Abalone	4177	3	1N+7C
Nettalk(s)	5438	5	7N
Coding	20000	2	15N

N-nominal; B-binary; C: Continuous.

For the real-world datasets, W-fold cross-validation is performed. In each fold of this cross-validation, the training dataset is used as $\mathcal{L}$, and the models derived are evaluated on the test dataset. The result is expressed as the average error rate of the W-fold cross-validation. Note that this cross-validation is used for evaluation of the entire procedure, and is quite different from the J-fold cross-validations employed as part of the stacked generalization operation. However, both W and J are set to 10 in the experiments.

Table 2 shows the average error rates of C4.5, NB and IB1, and BestCV, which is the best of the three, selected using J-fold cross-validation. As expected, BestCV is almost always the classifier with the lowest error rate.[2]

Table 3 shows the result of stacked generalization using the level-1 model $\tilde{\mathcal{M}}$, for which the level-1 data comprises the classifications generated by the level-0 models, and $\tilde{\mathcal{M}}'$, for which the level-1 data comprises the probabilities generated by the level-0 models. Results are shown for the best two level-1 generalizers in each case (full results see Ting and Witten [1997a]), along with BestCV. The lowest error rate for each dataset is given in bold.

Table 4 summarizes the results in Table 3 in terms of a comparison of each level-1 model with BestCV totaled over all datasets. Clearly, the best level-1 model is $\tilde{\mathcal{M}}'$ derived using MLR. It performs better than BestCV in nine datasets and equally well in the tenth. The best performing $\tilde{\mathcal{M}}$ is derived from NB, which performs better than BestCV in seven datasets but significantly worse in two.

The datasets are shown in the order of increasing size. MLR performs significantly better than BestCV in the

[2] Note that BestCV does not always select the same classifier in all W folds. That is why its error rate is not always equal to the lowest error rate among the three classifiers.

Table 2: Ave. error rates of C4.5, NB and IB1, and the best among them selected using J-fold cross-validation.

Datasets	Level-0 Generalizers			
	C4.5	NB	IB1	BestCV
Led24	35.4	35.4	32.2	32.8
Waveform	31.8	17.1	26.2	17.1
Horse	15.8	17.9	15.8	17.1
Credit	17.4	17.3	28.1	17.4
Vowel	22.7	51.0	2.6	2.6
Euthyroid	1.9	9.8	8.6	1.9
Splice	5.5	4.5	4.7	4.5
Abalone	41.4	42.1	40.5	40.1
Nettalk(s)	17.0	15.9	12.7	12.7
Coding	27.6	28.8	25.0	25.0

Table 3: Ave. error rates for stacking C4.5, NB and IB1.

Datasets		$\tilde{\mathcal{M}}$		$\tilde{\mathcal{M}}'$	
	BestCV	NB	MLR	IB1	MLR
Led24	32.8	32.4	33.3	**32.1**	**32.1**
Waveform	17.1	19.2	17.2	17.8	**16.8**
Horse	17.1	**14.9**	16.3	17.7	15.2
Credit	17.4	16.1	17.4	**14.3**	16.2
Vowel	2.6	3.8	2.6	3.3	**2.5**
Euthyroid	**1.9**	**1.9**	**1.9**	2.0	**1.9**
Splice	4.5	3.9	**3.8**	**3.8**	**3.8**
Abalone	40.1	38.5	38.1	39.2	**37.9**
Nettalk(s)	12.7	11.9	12.6	12.0	**11.5**
Coding	25.0	23.1	23.2	21.2	**20.7**

Table 4: Summary of Table 3—Comparison of BestCV with $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{M}}'$.

	$\tilde{\mathcal{M}}$		$\tilde{\mathcal{M}}'$	
	NB	MLR	IB1	MLR
#Win vs. #Loss	2-7	2-5	4-6	0-9

four largest datasets.[3] This indicates that stacked generalization is more likely to give significant improvements in predictive accuracy if the volume of data is large—a direct consequence of more accurate estimation using cross-validation.

MLR has an advantage over the other three level-1 generalizers in that its model can easily be interpreted. Examples of the combination weights it derives (for the probability-based model $\tilde{\mathcal{M}}'$) appear in Table 5 for the Splice dataset. The weights indicate the relative importance of the level-0 generalizers for each prediction class. In this dataset, NB is the dominant generalizer for predicting class 2, NB and IB1 are both good at predicting class 3, and all three generalizers make a worthwhile contribution to the prediction of class 1.

Table 5: Weights generated by MLR (model $\tilde{\mathcal{M}}'$) for the Splice dataset. The first column indicates the class.

	C4.5			NB			IB1		
C	α_{11}	α_{12}	α_{13}	α_{21}	α_{22}	α_{23}	α_{31}	α_{32}	α_{33}
1	0.23	0.00	0.00	0.43	0.00	0.00	0.36	0.00	0.00
2	0.00	0.15	0.00	0.00	0.72	0.00	0.00	0.12	0.00
3	0.00	0.01	0.08	0.00	0.00	0.52	0.00	0.01	0.40

Table 6: Average error rates of three versions of MLR. NC - no constraints; NI - no intercept.

Datasets	MLR with		
	NC	NI	Non-Negativity
Led24	34.1	34.1	32.1
Waveform	16.8	16.8	16.8
Horse	15.8	15.8	15.2
Credit	16.2	16.2	16.2
Vowel	2.4	2.4	2.5
Euthyroid	1.9	1.9	1.9
Splice	3.7	3.7	3.8
Abalone	37.9	37.9	37.9
Nettalk(s)	11.5	11.5	11.5
Coding	20.7	20.7	20.7

3.2 Are non-negativity constraints necessary?

Both Breiman [1996a] and LeBlanc and Tibshirani [1993] use the stacked generalization method in a regression setting and report that it is necessary to constrain the regression coefficients to be non-negative in order to guarantee that stacked regression improves predictive accuracy. Here we investigate this finding in the domain of classification tasks.

To assess the effect of the non-negativity constraint on performance, three versions of MLR are employed to derive the level-1 model $\tilde{\mathcal{M}}'$:

i each linear regression in MLR is calculated with an intercept constant (that is, $I + 1$ weights for the I classes) but without any constraints;

ii each linear regression is derived with neither an intercept constant (I weights for I classes) nor constraints;

iii each linear regression is derived without an intercept constant, but with non-negativity constraints (I non-negative weights for I classes).

The third version is the one used for the results presented earlier. Table 6 shows the results of all three versions. They all have almost indistinguishable error rates. We conclude that in classification tasks, non-negativity constraints are not necessary to guarantee that stacked generalization improves predictive accuracy.

However, there is another reason why it is a good idea to employ non-negativity constraints. Table 7 shows an example of the weights derived by these three versions

[3] We regard a difference of more than two standard errors as significant (95% confidence).

Table 7: Weights for the Euthyroid dataset with three versions of MLR: (i) no constraints, (ii) no intercept, and (iii) non-negativity constraints.

			C4.5		NB		IB1	
	C	α_0	α_{11}	α_{12}	α_{21}	α_{22}	α_{31}	α_{32}
i	1	–0.23	0.99	0.06	–0.05	–0.03	0.30	0.20
	2	1.02	–1.03	–0.10	0.17	0.16	–0.17	–0.07
ii	1	–	0.99	0.05	0.08	0.09	–0.05	–0.14
	2	–	–1.01	–0.07	–0.41	–0.42	1.40	1.49
iii	1	–	0.93	0.00	0.00	0.00	0.09	0.00
	2	–	0.00	0.93	0.01	0.00	0.00	0.07

Table 8: Ave. error rates of BestCV, Majority Vote and MLR (model $\tilde{\mathcal{M}}'$), along with the standard error (#SE) between BestCV and the worst level-0 generalizers.

Dataset	#SE	BestCV	Majority	MLR
Horse	0.5	17.1	**15.0**	15.2
Splice	2.5	4.5	4.0	**3.8**
Abalone	3.3	40.1	39.0	**37.9**
Led24	8.7	32.8	**31.8**	32.1
Credit	8.9	17.4	**16.1**	16.2
Nettalk(s)	10.8	12.7	12.2	**11.5**
Coding	12.7	25.0	23.1	**20.7**
Waveform	18.7	17.1	19.5	**16.8**
Euthyroid	26.3	**1.9**	8.1	**1.9**
Vowel	242.0	2.6	13.0	**2.5**

of MLR on the Euthyroid dataset. The third version, shown in row (iii), supports a more perspicuous interpretation of each level-0 generalizer's contribution to the class predictions than do the other two. In this dataset C4.5 is the dominant generalizer, as evidenced by its high weights. However, the negative weights render the interpretation of the other two versions much less clear.

3.3 How does stacked generalization compare to majority vote?

Let us now compare the error rate of $\tilde{\mathcal{M}}'$, derived from MLR, to that of majority vote, a simple decision combination method which requires neither cross-validation nor level-1 learning. Table 8 shows the average error rates of BestCV, majority vote and MLR. In order to see whether the relative performances of level-0 generalizers have any effect on these methods, the number of standard errors (#SE) between the error rates of the worst performing level-0 generalizer and BestCV is given, and the datasets are re-ordered according to this measure. Since BestCV almost always selects the best performing level-0 generalizer, small values of #SE indicate that the level-0 generalizers perform comparably to one another, and vice versa.

MLR compares favorably to majority vote, with seven wins versus three losses. Out of the seven wins, six have significant differences (the only exception is for the Splice dataset); whereas all three losses have insignificant differences. Thus the extra computation for cross-validation and level-1 learning seems to have paid off.

It is interesting to note that the performance of majority vote is related to the size of #SE. Majority vote compares favorably to BestCV in the first seven datasets, where the values of #SE are small. In the last three, where #SE is large, majority vote performs worse. This indicates that if the level-0 generalizers perform comparably, it is not worth using cross-validation to determine the best one, because the result of majority vote—which is far cheaper—is not significantly different. The same applies when majority vote is compared with MLR. MLR performs significantly better in the five datasets that have large #SE values, but only one in the other cases.

Summary

- None of the four learning algorithms used to obtain model $\tilde{\mathcal{M}}$ perform satisfactorily.
- MLR is the best of the four learning algorithms to use as the level-1 generalizer for obtaining model $\tilde{\mathcal{M}}'$.
- When obtained using MLR, $\tilde{\mathcal{M}}'$ has lower predictive error rate than the best model selected by J-fold cross-validation, for almost all datasets used in the experiments.
- Another advantage of MLR over the other three level-1 generalizers is its interpretability. The weights α_{ki} indicates the different contributions that each level-0 model makes to the prediction classes.
- Model $\tilde{\mathcal{M}}'$ can be derived by MLR with or without non-negativity constraints. Such constraints make little difference to the model's predictive accuracy.
- The use of non-negativity constraints in MLR has the advantage of interpretability. Non-negative weights α_{ki} support easier interpretation of the extent to which each model contributes to each class.
- When derived using MLR, model $\tilde{\mathcal{M}}'$ compares favorably with majority vote.

4 Related work

Our analysis of stacked generalization was motivated by that of Breiman [1996a], discussed earlier, and LeBlanc and Tibshirani [1993]. LeBlanc and Tibshirani [1993] examine the stacking of a linear discriminant and a nearest neighbor classifier and show that, for one artificial dataset, a method such as MLR performs better with non-negativity constraints than without. Our results show that these constraints are irrelevant to MLR's predictive accuracy in the classification situation.

The limitations of MLR are well-known [Duda and Hart, 1973]. For a I-class problem, it divides the description space into I convex decision regions. Every region is singly connected, and the decision boundaries are linear hyperplanes. This means that MLR is most suitable for problems with unimodal probability densities. Despite these limitations, MLR still performs better as a level-1 generalizer than IB1, its nearest competitor in deriving $\tilde{\mathcal{M}}'$. These limitations may hold the key for a fuller understanding of the behavior of stacked generalization. Jacobs [1995] reviews linear combination methods like that used in MLR.

Previous work on stacked generalization, especially as applied to classification tasks, has been limited in several ways. They have a different focus and evaluate the results on just a few datasets [LeBlanc and Tibshirani, 1993; Chan and Stolfo, 1995; Kim and Bartlett, 1995; Fan *et al.*, 1996].

5 Conclusions and future work

We have addressed two crucial issues for the successful implementation of stacked generalization in classification tasks. First, class probabilities should be used instead of the single predicted class as input attributes for higher-level learning. Second, the multi-response least squares linear regression technique should be employed as the high-level generalizer.

When combining three different types of learning algorithms, this implementation of stacked generalization was found to achieve better predictive accuracy than both model selection based on cross-validation and majority vote. Unlike stacked regression, non-negativity constraints in the least-squares regression are not necessary to guarantee improved predictive accuracy in classification tasks. However, these constraints are still preferred because they increase the interpretability of the level-1 model.

This paper concentrates on finding conditions under which stacked generalization works. A better understanding of why it works in this particular configuration may open up other possibilities for further improvement of stacked generalization.

The implication of our successful implementation of stacked generalization is that earlier model combination methods which employs (weighted) majority vote, averaging, or other computations that do not make use of level-1 learning, can now apply this learning to improve their predictive accuracy. This includes methods which combine models derived from a single learning algorithm such as bagging and arcing [Breiman, 1996b; 1996c]. The full version of this paper [Ting and Witten, 1997a] and a subsequent investigation [Ting and Witten, 1997b] include studies of this kind of model combination.

Acknowledgments

The authors are grateful to the New Zealand Marsden Fund for financial support for this research.

References

[Aha *et al.*, 1991] David W. Aha, Dennis Kibler and Marc K. Albert. Instance-Based Learning Algorithms, *Machine Learning, 6*, pp. 37-66, 1991.

[Breiman, 1996a] Leo Breiman. Stacked Regressions, *Machine Learning*, Vol. 24, pp. 49-64, 1996.

[Breiman, 1996b], Leo Breiman. Bagging Predictors, *Machine Learning*, Vol. 24, No. 2, pp. 123-140, 1996.

[Breiman, 1996c], Leo Breiman. Bias, Variance, and Arcing Classifiers, *Technical Report 460*, Department of Statistics, University of California, Berkeley, CA.

[Cestnik, 1990] B. Cestnik. Estimating Probabilities: A Crucial Task in Machine Learning, in *Proceedings of the European Conference on Artificial Intelligence*, pp. 147-149, 1990.

[Chan and Stolfo, 1995] Philip K. Chan and Salvatore J. Stolfo. A Comparative Evaluation of Voting and Meta-learning on Partitioned Data, in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 90-98, Morgan Kaufmann.

[Cost and Salzberg, 1993] Scott Cost and S. Salzberg. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features, *Machine Learning, 10*, pp. 57-78.

[Duda and Hart, 1973] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*, Wiley-Interscience.

[Fan *et al.*, 1996] David W. Fan, Philip K. Chan, S.J. Stolfo. A Comparative Evaluation of Combiner and Stacked Generalization, in *Proceedings of AAAI-96 workshop on Integrating Multiple Learned Models*, pp. 40-46.

[Jacobs, 1995] Robert A. Jacobs. Methods of Combining Experts' Probability Assessments, *Neural Computation 7*, pp. 867-888, MIT Press.

[Kim and Bartlett, 1995] Keehoon Kim and Eric B. Bartlett. Error Estimation by Series Association for Neural Network Systems, *Neural Computation 7*, pp. 799-808, MIT Press.

[Lawson and Hanson, 1995] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*, SIAM Publications.

[LeBlanc and Tibshirani, 1993] Michael LeBlanc and Robert Tibshirani. Combining Estimates in Regression and Classification, *Technical Report 9318*, Department of Statistics, University of Toronto.

[Merz and Murphy, 1996] Christopher J. Merz and P.M. Murphy. *UCI Repository of machine learning databases* [http:// www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Dept of Information and Computer Science.

[Quinlan, 1993] J.Ross Quinlan. *C4.5: Program for machine learning*, Morgan Kaufmann.

[Ting and Witten, 1997a] Kai Ming Ting and Ian H. Witten. Stacked Generalization: when does it work?. *Working Paper 97/3*, Department of Computer Science, University of Waikato.

[Ting and Witten, 1997b] Kai Ming Ting and Ian H. Witten. Stacking Bagged and Dagged Models, to appear in Proceedings of the Fourteenth International Conference on Machine Learning.

[Wolpert, 1992] David H. Wolpert. Stacked Generalization, *Neural Networks*, Vol. 5, pp. 241-259, Pergamon Press.