A Fusion of Stacking with Dynamic Integration

Niall Rooney, David Patterson
Northern Ireland Knowledge Engineering Laboratory
Faculty of Engineering, University of Ulster
Jordanstown, Newtownabbey, BT37 OQB, U.K.
{nf.rooney, wd.patterson}@ulster.ac.uk

Abstract

In this paper we present a novel method that fuses the ensemble meta-techniques of Stacking and Dynamic Integration (DI) for regression problems, without adding any major computational overhead. The intention of the technique is to benefit from the varying performance of Stacking and DI for different data sets, in order to provide a more robust technique. We detail an empirical analysis of the technique referred to as *weighted Meta – Combiner* (wMetaComb) and compare its performance to Stacking and the DI technique of Dynamic Weighting with Selection. The empirical analysis consisted of four sets of experiments where each experiment recorded the cross-fold evaluation of each technique for a large number of diverse data sets, where each base model is created using random feature selection and the same base learning algorithm. Each experiment differed in terms of the latter base learning algorithm used. We demonstrate that for each evaluation, wMetaComb was able to outperform DI and Stacking for each experiment and as such fuses the two underlying mechanisms successfully.

1 Introduction

The objective of ensemble learning is to integrate a number of base learning models in an ensemble so that the generalization performance of the ensemble is better than any of the individual base learning models. If the base learning models are created using the same learning algorithm, the ensemble learning schema is *homogeneous* otherwise it is *heterogeneous*. Clearly in the former case, it is important that the base learning models are not identical, and theoretically it has been shown that for such an ensemble to be effective, it is important that the base learning models are sufficiently accurate and *diverse* in their predictions [Krogh & Vedelsby, 1995], where *diversity* in terms of regression is usually measured by the ambiguity or variance in their predictions. To achieve this in terms of homogeneous learning, methods exist that either manipulate, for each model, the training data through instance or feature sampling methods or manipulate the learning parameters of the learning algorithm. Brown *et al.*[2004] provide a recent survey of such ensemble generation methods.

Having generated an appropriate set of diverse and sufficiently accurate models, an ensemble integration method is required to create a successful ensemble. One well–known approach to ensemble integration is to combine the models using a learning model itself. This process, referred to as Stacking [Wolpert, 1992], forms a meta-model created from a meta-data set. The meta-data set is formed from the predictions of the base models and the output variable of each training instance. As a consequence, for each training instance $I =< x, y >$ belonging to training set D, meta-instance $MI =< \hat{f}_1(x), .. \hat{f}_N(x), y >$ is formed, where $\hat{f}_i(x)$ is the prediction of base model $i, i = 1..N$ where N is the size of the ensemble. To ensure that the predictions are unbiased, the training data is divided by a J-fold cross-validation process into J subsets, so that for each iteration of a cross-validation process, one of the subsets is removed from the training data and the models are re-trained on the remaining data. Predictions are then made by each base model on the held out sub-set. The meta-model is trained on the accumulated meta-data after the completion of the cross validation process and each of the base models is trained on the whole training data. Breiman[1996b] showed the successful application of this method for regression problems using a linear regression method for the meta-model where the coefficients of the linear regression model were constrained to be non-negative. However there is no specific requirement to use linear regression as the meta-model [Hastie *et al.*, 2001] and other linear algorithms have been used particularly in the area of classification [Seewald, 2002, Džeroski, & Ženko, 2004].

A technique which appears at first very similar in scope to Stacking (and can be seen as a variant thereof) is that proposed by Tsymbal *et al.*[2003] referred to as Dynamic Integration. This technique also runs a *J*-fold cross validation

process for each base model, however the process by which it creates and uses meta-data is different. Each meta-instance consists of the following format $< x, Err_1(x), .., Err_N(x), y >$ where $Err_i(x)$ is the prediction error made by each base model for each training instance $< x, y >$. Rather than forming a meta-model based on the derived meta-data, Dynamic Integration uses a k-nearest neighbour (k-NN) lazy model to find the nearest neighbours based on the original instance feature space (which is a subspace in the meta-data) for a given test instance. It then determines the total training error of the base models recorded for this meta-nearest neighbour set. The level of error is used to allow the dynamic selection of one of the base models or an appropriate weighted combination of all or a select number of base models to make a prediction for the given test instance. Tsymbal *et al.* [2003] describes three Dynamic Integration techniques for classification which have been adapted for regression problems [Rooney *et al.*, 2004]. In this paper we consider one variant of these techniques referred to as Dynamic Weighting with Selection (DWS) which applies a weighting combination to the base models based on their localized error, to make a prediction, but excludes first those base models that are deemed too inaccurate to be added to the weighted combination.

A method of injecting diversity into a homogeneous ensemble is to use random subspacing [Ho, 1998a, 1998b]. Random subspacing is based on the process whereby each base model is built with data with randomly subspaced features. Tsymbal *et al.* [2003] revised this mechanism so that a variable length of features are randomly subspaced. They shown this method to be effective in creating ensembles that were integrated using DI methods for classification.

It has been shown empirically that Stacking and Dynamic Integration based on integrating models, generated using random subspacing, are sufficiently different from each other in that they can give varying generalization performance on the same data sets [Rooney *et al.*, 2004]. The main computational requirement in both Stacking and Dynamic Integration, is the use of cross-validation of the base models to form meta-data. It requires very little additional computational overhead to propose an ensemble meta-learner that does both in one cross validation of the ensemble base models and as a consequence, is able to create both a Stacking meta-model and a Dynamic Integration meta-model, with the required knowledge for the ensemble meta-learner to use both models to form predictions for test instances. The focus of this paper is on the formulation and evaluation of such an ensemble meta-learner, referred to as weighted Meta-Combiner (wMetaComb). We investigated whether wMetaComb will on average outperform both Stacking for regression (SR) and DWS for a range of data sets and a range of base learning algorithms used to generate homogeneous randomly subspaced models.

2 Methodology

In this section, we describe in detail the process of wMetaComb. The training phase of wMetaComb is shown in Figure 1..

Algorithm : wMetaComb: Training Phase

Input: D

Output: $\hat{f}^{SR}$, $\hat{f}^{DI}$, $\hat{f}_{i,i=1..N}$

(* step 1 *)

partition D into J–fold partitions $D_{i=1..J}$

initialise meta-data set $MD^{SR} = \varnothing$

initialise meta-data set $MD^{DI} = \varnothing$

For i = 1..J-2

$D_{train} = D \setminus D_i$

$D_{test} = D_i$

build N learning models $\hat{f}_{i,i=1..N}$ using D_{train}

For each instance $< \mathbf{x}, y >$ in D_{test}

Form meta-instance $MI^{SR} : < \hat{f}_1(\mathbf{x}), .., \hat{f}_N(\mathbf{x}), y >$

Form meta-instance $MI^{DI} : < \mathbf{x}, Err_1(\mathbf{x}), .., Err_N(\mathbf{x}), y >$

Add MI^{SR} to MD^{SR}

Add MI^{DI} to MD^{DI}

EndFor

Endfor

Build SR meta-model $\hat{f}^{SR}$ using the meta-data set MD^{SR}

Build DI meta-model $\hat{f}^{DI}$ using the meta-data set MD^{DI}

(* step 2 *)

i=J

$D_{test} = D_i$

Determine $TotalErr_{SR}$ and $TotalErr_{DI}$ using D_{test}

(* step 3 *)

i = J-1

$D_{train} = D \setminus D_i$

$D_{test} = D_i$

build N learning models $\hat{f}_{i,i=1..N}$ using D_{train}

For each instance $< \mathbf{x}, y >$ in D_{test}

Form meta-instance $MI^{SR} : < \hat{f}_1(\mathbf{x}), .., \hat{f}_N(\mathbf{x}), y >$

Form meta-instance $MI^{DI} : < \mathbf{x}, Err_1(\mathbf{x}), .., Err_N(\mathbf{x}), y >$

Add MI^{SR} to MD^{SR}

Add MI^{DI} to MD^{DI}

EndFor

Retrain the $\hat{f}_{i,i=1..N}$ on D

Build SR meta-model $\hat{f}^{SR}$ using the meta-data set MD^{SR}

Build DI meta-model $\hat{f}^{DI}$ using the meta-data set MD^{DI}

Figure 1. Training Phase of wMetaComb

The first step in the training phase of wMetaComb consists of building SR and DI meta-models on *J*-1 folds of the training data (step 1). The second step involves testing the meta-models using the remaining training fold, as an indicator of their potential generalization performance. It is important to do this as even though models are not created

using the complete meta-data, they are tested using data that they are not trained on, in order to give a reliable estimate of their test performance. The performance of the meta-models is recorded by their total absolute error, $TotalError_{SR}$ and $TotalError_{DI}$ within step 2. The meta-models are then re-built using the entire meta-data completed by step 3. The computational overhead of wMetaComb in comparison to SR by itself centres only on the cost of training the additional meta-model. However as the additional meta-model is based on a lazy nearest neighbour learner, this cost is trivial.
The prediction made by wMetaComb is made by a simple weighted combination of its meta-models based on their total error performance determined in step 2. Denote $TotalErr = < TotalError^{SR}, TotalError^{DI} >$. The individual weight for each meta-model is determined by finding the normalized error for each meta-model:

$$normerror_i = TotalError_i / \sum_{i=1..2} TotalError_i$$

A normalized weighting for each meta-model is given by:

$$mw_i = e^{-\frac{normerror_i}{T}} \qquad (1)$$

$$norm\_mw_i = \frac{mw_i}{\sum_i mw_i} \qquad (2)$$

Equation 1 is influenced by a weighting mechanism that Wolpert & Macready [1996] proposed for combining base models generated by stochastic process such as used in Bagging [Breiman,1996a].
The weighting parameter T determines how much relative influence to give to each meta-model based on their normalized error e.g. suppose $normError_1 = 0.4$ and $normError_2 = 0.6$, then Table 1 gives the normalized weight values for different values of T. This indicates that a low value of T will give large imbalance to the weighting, whereas a high value of T almost equally weights the meta-learners regardless of their difference in normalized errors.

T	$norm\_mw_1$	$norm\_mw_2$
0.1	0.88	0.12
1.0	0.55	0.45
10.0	0.505	0.495

Table 1. The effect of the weighting parameter

Based on these considerations, we applied the following heuristic function for the setting of T, which increases the imbalance in the weighting dependent on how great the normalized errors differ.

$$T = \frac{|\, normError_1 - normError_2 \,|}{0.01} \qquad (3)$$

wMetaComb forms a prediction for a test instance x by the following combination of the predictions made by the meta-models:

$$norm\_mw_1 * \hat{f}^{SR}(x) + norm\_mw_2 . \hat{f}^{DI}(x)$$

3 Experimental Evaluation and Analysis

In these experiments, we investigated the performance of wMetaComb in comparison to SR and DWS. The work was carried out based on appropriate extensions to the WEKA environment, such as the implementation of DWS. The ensemble sets consisted of 25 base homogeneous models where the data for each base model was randomly subspaced [Tsymbal *et al.*, 2003]. Note that random subspacing was a pseudo-random process so that a feature sub-set generated for each model *i* in an ensemble set is the same for all ensemble sets created using different base learning algorithms and the same data set. The meta-learner for SR and the Stacking component within wMetaComb was based on the model tree M5 [Quinlan, 1992], as this has a larger hypothesis space than Linear Regression. The number of nearest neighbours for DWS k-NN meta-learner was set to 15. The performance of DWS can be affected by the size of k for particular data sets, but for this particular choice of data sets, previous empirical analysis had shown that the value of 15, gave overall a relatively strong performance. The value of J during the training phase of each meta-technique was set to 10.
In order to evaluate the performance of the different ensemble techniques, we chose 30 data sets available from the WEKA environment [Witten & Frank, 1999]. These data sets represent a number of synthetic and real-world problems from a variety of different domains, have a varying range in their attribute sizes and many contain a mixture of both discrete and continuous attributes. Data sets which had more than 500 instances, were sampled without replacement to reduce the size of the data set to a maximum of 500 instances, in order to make the evaluation computationally tractable. Any missing values in the data set were replaced using a mean or modal technique. The characteristics of data sets are shown in Table 2.
We carried out a set of four experiments, where each experiment differed purely in the base learning algorithm deployed to generate base models in the ensemble. Each experiment calculated the relative root mean squared error (RRMSE) [Setiono *et al.*, 2002] of each ensemble technique for each data set based on a 10 fold cross validation measure.

Data set	Original Data set size	Data set size in experiments	Number of input Attributes		
			Continuous	Discrete	Total
abalone	4177	500	8	0	8
autohorse	205	205	15	10	25
autoMpg	398	398	5	3	8
autoprice	159	159	15	10	25
auto93	93	93	16	5	21
bank8FM	8192	500	8	0	8
bank32nh	8192	500	32	0	32
bodyfat	252	252	14	0	14
breastTumor	286	286	1	8	9
cal_housing	20640	500	8	0	8
cloud	108	108	4	2	6
cpu	209	209	6	1	7
cpu_act	8192	500	21	0	21
cpu_small	8192	500	12	0	12
echomonths	130	130	6	3	9
elevators	16559	500	18	0	18
fishcatch	158	158	5	2	7
friedman1	40768	500	10	0	10
housing	506	500	13	0	13
house_8L	22784	500	8	0	8
house_16H	22784	500	16	0	16
lowbwt	189	189	7	2	9
machine_cpu	209	209	6	0	6
pollution	60	60	15	0	15
pyrim	74	74	27	0	27
sensory	576	500	0	11	11
servo	167	167	4	0	4
sleep	62	62	7	0	7
strike	625	500	5	1	6
veteran	137	137	7	0	7

Table 2. The data sets' characteristics

The RRMSE is a percentage measure. By way of comparison of the ensemble approaches in general, we also determined the RRMSE for a single model built on the whole unsampled data, using the same learning algorithm as used in the ensemble base models. We repeated this evaluation four times, each time using a different learning algorithm to generate base models. The choice of learning algorithms was based on a choice of simple and efficient methods in order to make the study computationally tractable, but also to have representative range of methods with different learning hypothesis spaces in order to determine whether wMetaComb was a robust method independent of the base learning algorithm deployed. The base learning algorithms used for the four experiments were 5 nearest neighbours (5-NN), Linear Regression (LR), Locally weighted regression [Atkeson *et al.*, 1997] and the model tree learner M5.

The results of the evaluation were assessed by performing a 2 tailed paired t test (p=0.05) on the cross fold RRMSE for each technique in comparison to each other for each data set. The results of the comparison between one technique A and another B were then summarised in the form *wins:losses:ties (ASD)* where *wins* is a count of the number of data sets where technique *A* outperformed technique B significantly, *losses* is a count of the number of data sets where technique B outperformed technique A significantly and *ties* a count where there was no significant difference. The *significance gain* is the difference in the number of wins and the number of losses. If this zero or less, no gain was shown. If the gain is less than 3 this is indicative of only a slight improvement.

We determined the average significance difference (*ASD*) between the RRMSE of the two techniques, averaged over $wins + losses$ data sets only where a significant difference was shown. If technique A outperformed B as shown by the significance ratio, the ASD gave us a percentage measure of the degree by which A outperformed B (if technique A was outperformed by B, then the ASD value is negative). The ASD by itself of course has no direct indication of the performance of the techniques and is only a supplementary measure. A large ASD does not indicate that technique A performed considerably better than B if the difference in the number of wins to losses was small. Conversely even if the number of wins was much larger than the number of losses but the ASD was small, this reduces the impact of the result considerably. This section is divided into a discussion of the summary of results for each individual base learning algorithm then a discussion of the overall results.

3.1 Base Learner: *k-NN*

Table 3 shows a summary of the results for the evaluation when the base learning algorithm was 5-NN. Clearly all the ensemble techniques strongly outperformed single 5-NN whereby in terms of significance alone, wMetaComb shown the largest improvement with a gain of 22. Although SR showed fewer data sets with significant improvement, it had a greater ASD than wMetaComb. wMetaComb outperformed DWS with a gain of 7, and showed a small improvement over SR, with a gain of 3. SR showed a significance gain of only 1 over DWS.

Technique	**5-NN**	**SR**	**DWS**	**wMetaComb**
5-NN		0:11:19 (-16.07)	0:20:10 (-12.28)	0:22:8 (-14.47)
SR	11:0:19 (16.07)		3:2:25 (3.82)	0:3:27 (-6.8)
DWS	20:0:10 (12.28)	2:3:25 (-3.82)		1:8:21 (-5.28)
wMetaComb	22:0:8 (14.47)	3:0:27 (6.8)	8:1:21 (5.28)	

Table 3. Comparison of methods when base learning algorithm was 5-NN

3.2 Base Learner: *LR*

Table 4 shows that all ensembles techniques outperformed single LR, with wMetaComb showing the largest improvement with a gain of 13 and an ASD of 11.99. wMetaComb outperformed SR and DWS with a larger improvement shown in comparison to DWS than to SR both in terms of significance and ASD. SR outperformed DWS with a significance gain of 7 and ASD of 5.1.

Technique	LR	SR	DWS	wMetaComb
LR		2:10:18 (-11.38)	1:11:18 (-5.56)	0:13:17 (-11.99)
SR	10:2:18 (11.38)		11:4:15 (5.1)	0:5:25 (-5.58)
DWS	11:1:18 (5.56)	4:11:15 (-5.1)		3:13:14 (-6.02)
wMetaComb	13:0:17 (11.99)	5:0:25 (5.58)	13:3:14 (6.02)	

Table 4. Comparison of methods when base learning algorithm was LR

3.3 Base Learner: *LWR*

Table 5 shows that all ensembles techniques outperformed single LWR, with wMetaComb showing the largest improvement with a significance gain of 16. wMetaComb outperformed SR and DWS with a larger improvement shown over DWS than SR in terms of significance and ASD. SR outperformed DWS with a significance gain of 4 and an ASD of 6.24.

Technique	LWR	SR	DWS	wMetaComb
LWR		1:9:20 (-13.81)	0:16:14 (-12.28)	0:16:14 (-12.28)
SR	9:1:20 (13.81)		6:2:22 (6.24)	0:5:25 (-10.48)
DWS	16:0:14 (12.28)	2:6:22 (-6.24)		0:9:21 (-8.28)
wMetaComb	16:0:14 (15.91)	5:0:25 (10.48)	9:0:21 (8.28)	

Table 5. Comparison of methods when base learning algorithm was LWR

Table 5 shows that all ensembles techniques outperformed single LWR, with wMetaComb showing the largest improvement with a significance gain of 16. wMetaComb outperformed SR and DWS with a larger improvement shown over DWS than SR in terms of significance and ASD. SR outperformed DWS with a significance gain of 4 and ASD of 6.24.

3.4 Base Learner: M5

Technique	M5	SR	DWS	wMetaComb
M5		3:1:26 (4.4)	1:6:23 (-3.45)	0:8:22 (-4.51)
SR	1:3:26 (-4.4)		2:6:22 (-2.62)	0:11:19 (-4.48)
DWS	6:1:23 (3.45)	6:2:22 (2.62)		0:6:24 (-3.53)
wMetaComb	8:0:22 (4.51)	11:0:19 (4.48)	6:0:24 (3.53)	

Table 6. Comparison of methods when base learning algorithm was M5

Table 6 shows a much reduced performance for the ensembles in comparison to the single model, when the base learning algorithm was M5. In fact, SR showed no improvement over M5, and wMetaComb and DWS although they improved on M5 in terms of significance did not show an ASD as large as for the previous learning algorithms. The reduced performance with M5 is indicative that in the case of certain learning algorithms, random subspacing by itself is not able to generate sufficiently accurate base models, and needs to be enhanced using search approaches that improve the quality of ensemble members such as considered in [Tsymbal *et al.,* 2005]. wMetaComb showed improvements over SR and to a lesser degree both in terms of significance and ASD values.

Overall, it can be seen that wMetaComb gave the strongest performance of the ensembles for 3 out of 4 base learning algorithms (5-NN, LR, M5) in comparison to the single model, in terms of its significance gain, and tied in comparison to DWS for LWR. For 3 out of the 4 learning algorithms (LR, LWR, M5), wMetaComb also had the highest ASD compared to the single model. wMetaComb outperformed SR for all learning algorithms. The largest significant improvement over SR was recorded with M5 with a significance gain of 11, and the largest ASD in comparison to SR was recorded with LWR of 10.48. The lowest significant improvement over SR was recorded with 5-NN, LR, and LWR with a significant gain of 5, and lowest ASD of 4.48 with M5.

wMetaComb outperformed DWS for all four base learning algorithms. The degree of improvement was larger in comparison to DWS than SR. The largest significant improvement over DWS was shown with LR with a significance gain of 10 and the largest ASD of 8.28 with LWR. The lowest improvement was shown with M5 both in terms of gain and ASD which were 6 and 3.53 respectively. In effect, wMetaComb was able to fuse effectively the overlapping expertise of the two meta-techniques. This was seen to be case independent of the learning algorithm employed. In addition, wMetaComb provided benefit whether overall for a given experiment, SR was stronger than DWS or not (as

was the case with LR and LWR, but not M5), indicating the general robustness of the technique.

4 Conclusions

We have presented a technique called wMetaComb that merges the technique of Stacking for regression and Dynamic integration for regression, and showed that it was successfully able to benefit from the individual meta-techniques' often complementary expertise for different data sets. This was demonstrated based on 4 sets of experiments each using different base learning algorithms to generate homogeneous ensemble sets. It was shown that for each experiment, wMetaComb outperformed Stacking and the Dynamic Integration technique of Dynamic Weighting with Selection. In future, we intend to generalize the technique so as to allow the combination of more than two meta-ensemble techniques. In this, we may consider combinations of Stacking using different learning algorithms or other Dynamic integration combination methods.

References

[Atkeson *et al*., 1997] Atkeson, C. G., Moore, A. W., and Schaal, S.. Locally weighted learning. *Artificial Intelligence Review*, 11:11-73, Kluwer, 1997.

[Breiman, L., 1996a]. Bagging Predictors. *Machine learning*, 24:123-174, Kluwer, 1996.

[Breiman, L., 1996b] Stacked Regressions. *Machine Learning*, 24:49-64, Kluwer, 1996.

[Brown *et al*., 2004] Brown, G., Wyatt, J., Harris, R. amd Yao, X. Diversity Creation Methods: A Survey and Categorisation. *Information Fusion 6(1):5-20,* Elsevier,2004.

[Džeroski, & Ženko, 2004] Džeroski, S. & Ženko, B. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning* 54(3):255-273, Kluwer. 2004.

[Hastie *et al.*, 2001] Hastie, T., Tibshirani, R., Friedman, J. *The Elements of Statistical Learning:Data mining, Inference and Prediction*. Springer series in Statistics , 2001.

[Ho, 1998a] Ho, T.K. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832-844, 1998.

[Ho, 1998b] Ho, T.K. Nearest Neighbors in Random Subspaces. In *Proceedings of the Joint IAPR Workshops on Advances in Pattern Recognition*, LNCS, Vol. 1451, pp. 640-648, Springer-Verlag, 1998.

[Krogh & Vedelsby, 1995] Krogh, A. and Vedelsby, J. Neural Networks Ensembles, Cross validation, and Active Learning. In *Advances in Neural Information Processing System*s , pp. 231-238, MIT Press, 1995.

[Quinlan, 1992] Quinlan, R. Learning with continuous classes, In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pp. 343-348, World Scientific, 1992.

[Rooney *et al.*, 2004] Rooney, N., Patterson, D., Anand S, & Tsymbal, A. Dynamic Integration of Regression Models. In *Proceedings of the 5th International Multiple Classifier Systems Workshop*. LNCS Vol. 3077, pp. 164-173, Springer, 2004.

[Seewald, 2002] Seewald, A. Exploring the Parameter State Space of Stacking. In *Proceedings of the 2002 IEEE International Conference on Data Mining,* pp. 685-688, 2002.

[Setiono *et al.*, 2002] Setiono, R., Leow, W.K., & Zurada, J. Extraction of Rules from Artificial Neural Networks for Nonlinear Regression. *IEEE Transactions on Neural Networks*, 13(3):564-577, 2002.

[Tsymbal *et al.*, 2003] Tsymbal, A., Puuronen, S., Patterson, D. Ensemble feature selection with the simple Bayesian classification, *Information Fusion*, 4:87-100, Elsevier, 2003.

[Tsymbal *et al.*, 2005] Tysmbal, A. Pechenizkiy, M., Cunningham, P. Sequential Genetic Search for Ensemble Feature Selection. In *Proc. of the 19th International Joint Conference on Artificial Intelligence*, pp. 877-882, 2005.

[Witten & Frank, 1999] Witten, I. and Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.

[Wolpert, 1992] Wolpert, D. Stacked Generalization. *Neural Networks* 5, pp. 241-259, 1992.

[Wolpert & Macready, 1996] Wolpert, D. & Macready, W. Combining Stacking with Bagging to Improve a Learning Algorithm, Santa Fe Institute Technical Report 96-03-123, 1996.