Effective Stacking of Distributed Classifiers

Grigorios Tsoumakas and **Ioannis Vlahavas**[1]

Abstract. One of the most promising lines of research towards discovering global predictive models from physically distributed data sets is local learning and model integration. Local learning avoids moving raw data around the distributed nodes and minimizes communication, coordination and synchronization cost. However, the integration of local models is not a straightforward process. Majority Voting is a simple solution that works well in some domains, but it does not always offer the best predictive performance. Stacking on the other hand, offers flexibility in modelling, but brings along the problem of how to train on sufficient and at the same time independent data without the cost of moving raw data around the distributed nodes. In addition, the scalability of Stacking with respect to the number of distributed nodes is another important issue that has not yet been substantially investigated. This paper presents a framework for constructing a global predictive model from local classifiers that does not require moving raw data around, achieves high predictive accuracy and scales up efficiently with respect to large numbers of distributed data sets.

1 INTRODUCTION

Nowadays, physically distributed databases are increasingly being used for knowledge discovery. The ever-growing size of data being stored in today's information systems, inevitably leads to distributed database architectures, as data cannot fit in a single machine. Moreover, the offices or departments of many organizations and companies are scattered across a country or across the world, and central storage of data is often inefficient. In addition, globalization, business-to-business commerce and online collaboration between organizations rose lately the need for inter-organizational data mining.

Discovering interesting patterns from data residing in systems with the aforementioned properties is a challenging task. One has to deal with the physical distribution of data, as well as consider legal or social restrictions that might prevent data from being moved around.

One of the most promising lines of research towards discovering global predictive models from physically distributed data sets is local learning and model integration. This avoids moving raw data around the distributed nodes and minimizes communication, coordination and synchronization cost. However, the integration of local models is not a straightforward process.

Majority Voting is a simple solution that works well in some domains, but it does not always offer the best predictive performance. Stacking on the other hand, offers flexibility in modelling, but brings along the problem of how to train on sufficient and at the same time independent data without the cost of moving any of the raw data. In addition, the scalability of Stacking with respect to the number of distributed nodes is another important issue that has not yet been substantially investigated.

This paper presents a framework for combining distributed classifiers without moving raw data around that achieves high predictive accuracy and scales up efficiently with respect to large numbers of distributed data sets. The main contribution of this work is: i) a new methodology for training-based approaches to distributed data mining that avoids the problem of gathering parts of the raw data for training purposes and ii) an alternative classifier combination strategy to normal Stacking that scales up efficiently to large numbers of distributed data sets.

The rest of the paper is organized as follows. In Section 2 we review related methodologies for combining classifiers induced from distributed data sets. Section 3 presents the motivation underlying our approach, which is then presented in Section 4. Section 5 exhibits experimental results and finally in Section 6 we conclude and pose future research directions.

2 RELATED WORK

The way that multiple classifiers are combined is an interesting and important research issue that has been considerably investigated. When only the label of the predicted class is available, then Majority Voting [7] is used as a combination method. In this case, the class with the maximum classifier predictions is the final result.

When a measure of belief, confidence, certainty or other about the classification is available along with the class label, then a number of different rules for linear combination of these measures have been suggested, like Sum, Min, Max, Prod and Median. [8] is an interesting study of these rules.

An alternative class of classifier combination approaches involves learning a global classifier that combines the output of a number of classifiers. Stacked Generalization [14], also known as Stacking in the literature, is a method that combines multiple classifiers by learning the way that their output correlates with the true class on an independent set of instances. At a first step, N classifiers C_i, i = 1..N are induced from each of N data sets D_i, i = 1..N. Then, for every instance e_j, j = 1..L of an evaluation set E, independent of the D_i data sets, the output of the classifiers $C_i(e_j)$ along with the true class of the instance class(e_j) is used to form an instance m_j, j = 1..L of a new data set M, which will then serve as the meta-level training set. Each instance will be of the form: $C_1(e_j), C_2(e_j), \ldots, C_N(e_j)$, class($e_j$). Finally, a global classifier GC is induced directly from M. If a new instance appears for classification, the output of all local models is first calculated and then propagated to the global model, which outputs the final result.

Any algorithm suitable for classification problems can be used for learning the C_i and GC classifiers. Independence of the actual algorithm used for learning C_i, is actually one of the advantages of

[1] Dept. of Informatics, Aristotle University of Thessaloniki, 54006 Greece, email: {greg,vlahavas}@csd.auth.gr

Stacking, as not every algorithm might be available for each data set and not the same algorithm performs best for every data set. As far as the algorithm for learning the GC is concerned, Ting and Witten [11] showed that among a decision tree learning algorithm (C4.5), an instance based learning algorithm variant (IB1), a multi-response linear regression algorithm (MLR) and a Naive Bayes classifier, MLR had the best performance. Experimental results from the same work showed that on average Stacking is better than Majority Voting in terms of accuracy.

Chan and Stolfo [2], applied the concept of Stacked Generalization to distributed data mining, via their Meta-Learning methodology. They focused on combining distributed data sets and investigated various schemes for structuring the meta-level training examples. They showed that Meta-Learning exhibits better performance with respect to Majority Voting for a number of domains. Furthermore, it can be effectively used both in a distributed environment, and for scaling up learning to very large databases. It is also combined with an excellent agent-based architecture, which offers the ability to deal with heterogeneous environments [9]. However, it does not scale efficiently to domains with large number of classes and distributed nodes as it is shown in Section 3.2.

Knowledge Probing [5] builds on the idea of Meta-Learning and in addition uses an independent data set, called the probing set, in order to discover a comprehensible model. The output of a meta-learning system on this independent data set together with the attribute value vector of the same data set are used as training examples for a learning algorithm that outputs a final model. In this elegant way, the disadvantage of having a black box is overcome and the result is a transparent predictive model. However, the choice of size and origin of the probing set are issues that have to be thoroughly investigated, especially in the context of a distributed environment. In addition, as it is based on Meta-Learning, it suffers from the same problems in scaling up.

Davies and Edwards [3] follow a different approach to discover a single comprehensible model out of the distributed information sources. The main idea in the DAGGER algorithm is to selectively sample each distributed data set to form a new data set that will be used for inducing the single model. The new data set will consist of the union of the minimal spanning example sets that support every class at each data set. This method shows promising results in terms of accuracy and has the advantage of producing a single comprehensible model. However, it cannot be effectively used for mining physically distributed data sets, because it requires gathering some of the raw data in a single place.

An approach that attempts direct model integration, is that by Hall et al. [6]. Their approach involves learning decision trees in parallel from disjoint data, converting trees to rules and then combining the rules into a single rule set. The rule combination step is based on taking the merge of the N rule sets and resolving any conflicts that arise, based on work by Williams [13]. This approach has the advantage of producing a single comprehensible model, but is focused on combining models described by rules. Furthermore it does not consider all cases of rule conflicts, but only simple ones that derive from the choice of split point for continuous attributes by the C4.5 decision tree algorithm.

3 MOTIVATION

The motivation for this work was the following two problems that arise in approaches based on Stacking for large-scale distributed data mining: i) How to gather the necessary independent evaluation data without moving raw data around the distributed nodes and ii) How to deal with the increased complexity of the meta-level training process with respect to the number of distributed databases.

3.1 Gathering independent evaluation data

In Stacking and other classifier combination methodologies that involve training at the combination phase, it is necessary to gather instances that are independent of the ones that were used for training the local classifiers. The obvious reason is to avoid overfitting of the final model to the data.

The standard procedure is to hold out some instances from each distributed data set and use them for learning the global classifier. This however, has two major drawbacks. The first is that the meta-level training set competes with the local data sets over independent instances. Inevitably the global model is induced using only a sample of all available data. The second is that the meta-level instances need to be gathered in a single place in order to be used for global training. Current approaches usually perform simulated experiments and do not take into account the issue of moving raw data around, which could be time-consuming especially in the case of many distributed nodes.

To circumvent the first drawback, cross validation can be applied to every local algorithm. Each instance in the test set of each fold of the cross validation will be used for the creation of an instance of the meta-level training set. Therefore, after the complete process, each instance available at each local data set will have contributed to an instance in the meta-level training set, allowing the global classifier to make use of the complete data. However, this is time-consuming, deprives the local classifiers of some training data and requires even more raw data to be moved around.

3.2 Scaling up to many distributed databases

As discussed in Section 2, Stacking uses the output of a number of classifiers on an independent evaluation data set to form a meta-level training set. The number of attributes in a meta-level instance is equal to the number of distributed databases. Therefore, the complexity of learning the global classifier from the meta-level data set is proportional to the number of distributed databases.

Furthermore, Ting and Witten [11] have showed that better accuracy can be achieved when stacking the probability[2] distribution for every class instead of just the class with the highest probability. This actually worsens the complexity of learning the global classifier, because the number of attributes becomes the product of the number of distributed databases and the number of classes in the domain.

The two points mentioned above lead to the conclusion that there is a problem in scaling up Stacking, especially for domains with many classes. To exemplify this, consider combining 1000 classifiers for the task of English letter recognition. In this domain there are 26 classes, the letters A to Z. Following the Stacking approach in this particular example will produce meta-level instances with 26000 attributes. It is practically impossible to learn a global model from such a data set within acceptable time. In an experimental study of Stacking using the posterior probabilities of the classifiers output in this domain [12], up to 20 local classifiers were only used due to increased computational complexity.

[2] We will use the term probability here as the degree of certainty, belief or confidence that classifiers output along with a decision. Probability distribution is then a vector containing such measures for every class.

A possible way to deal with this issue would be to follow a divide-and-conquer approach, where the classifiers are grouped in ensembles of smaller size. Then, for each of these ensembles a meta-classifier is induced, leading to another ensemble of these new classifiers and another level of learning. This can be generalized in a hierarchical way with as many levels of classifiers as necessary. However, this has an opposite effect on the previous problem, as new independent instances are required for every new level in the hierarchy.

4 OUR APPROACH

We hereafter present our approach on Distributed Stacking of multiple classifiers that attempts to counter the problems of large-scale distributed data mining explained in the previous section. It can be broken down into the following phases:

1. *Local Learning*. Suppose that there are N distributed data sets. At each node N_i, i = 1..N, we use an available classification learning algorithm to train a local predictive model L_i from all instances of that node. There is no restriction to the choice of algorithm used in this phase. However, better performance can be achieved if the induced models output a distribution of certainty measures for every class, instead of a single label. Such models are more informative and the combination of their output is more effective. Note also that all data at a node contribute to the induction of its local model. Our approach does not demand holding a percentage of the data for training the global model. After the induction of the local model, each node broadcasts it to all other nodes. In the end of this phase, each node will contain every local model.
2. *Classifier Combination*. At each node N_i, i = 1..N, all models apart from the local one L_j, j = 1..N, $j \neq i$ will be combined to form a global classifier G_i. Leaving model L_i out ensures unbiased results, because the combination of local models requires training on the local data upon which L_i was induced. The combination will happen using the following classifier combination strategy, that performs Stacking of the averages of classifier predictions according to the class:
   - For every instance d_k, k = 1..size(D_i), of data set D_i at node N_i we calculate the posterior probability distribution $PD_{jc}(d_k)$ of each model L_j for every class c. We then average these probabilities according to the class over all classifiers and create a meta-level training instance m_k, k = 1..size(D_i). This instance has as attributes the calculated averages along with the true class, class(d_k), of the instance.
   - After the construction of all meta-level training instances m_k, a global model G_i is induced from these data. This global model describes the combined knowledge of all models apart from the local one with respect to the local data. As in the local learning phase, there is no restriction to the choice of algorithm used. Each node will then broadcast the induced global model to all other nodes. In the end of this phase, each node will contain every local and global model.

When a new instance appears for classification at a node, we first calculate the output of every local model. Then, we give as input to every global model G_i the average of the output of all corresponding local models L_j, j = 1..N, $j \neq i$. The final classification is given by combining the output of the global models using the Sum rule, which essentially outputs the class that has the maximum average posterior probability, since we do not incorporate any prior knowledge regarding the classifiers.

4.1 Advantages

First of all, our approach does not demand any communication of raw data to any of the distributed nodes, but only the broadcast of local and global classifiers to all nodes. Models however, have negligible size and do not add any significant communication overhead.

In addition, all initial raw data contribute to the creation of the final classification model. Each distributed data set serves as the training set for a local model and the evaluation set for a global model. As the final classification is given by the combination of all global models, all initial raw data have contributed to it. This contribution is made independently up to a certain point. Each global model is in a way independent of each other, because it is trained on different data.

On the other hand each global model combines the classification behavior of the same more or less local models (one local classifier difference). This process is similar to cross-validation, but in this case with respect to the models. We leave one model out and combine the rest on the data that was used to train that model. This process is expected to increase the stability of the classification result, an important issue today in data mining.

The size of the meta-level training examples stays constant, equal to the number of classes in the domain, irrespective of the number of local classifiers. This achieves tractability of modelling the classifier ensemble behavior in the case of large numbers of distributed data sets. The disadvantage is that we lose the fine grain modelling of how each classifier contributes to the prediction of Stacking. Instead, we get a coarse model of how the classifiers behave as an ensemble.

Despite the fact that our approach involves the same procedure as Stacking to take place as many times as the distributed nodes are, it actually does not add more computational complexity. The reason is that each global model learning process is executed in parallel at each distributed node.

Furthermore, our approach can be effectively used for scaling up learning to very large databases. A very large database can be horizontally fragmented to many smaller parts. Our approach has low computational complexity with respect to the number of participating data sets, and therefore even a huge database can be effectively mined by breaking it up into as many parts as necessary.

5 Experimental Results

In order to evaluate the proposed framework, a set of experiments that compare it to Stacking and Majority Voting were conducted using five of the largest real world and synthetic data sets from the UCI Machine Learning Repository [1]. The details of these data sets are described in Table 1.

The setup of the experiments was the following. Initially, the original data set is randomly split into a percentage (25%) of evaluation data and the rest (75%) is again randomly split into a variable number (10, 20, 40 and 80) of distributed data sets. This allows us to simulate a variety of distributed database configurations and examine how the methodologies scale up with respect to the number of distributed nodes. Then, C4.5 [10] is used to learn a local predictive model from each one of the distributed data sets. The predictions of these local models on the evaluation data are then combined using the three methodologies.

Stacking was implemented using the strategy that has been found to be the most successful as explained in Section 3.2. The probabilistic predictions of all local models are first combined on the evaluation data set to form the meta-level training examples. Then a global classifier is induced from these examples using the C4.5 algorithm

Table 1. Details of data sets used in the experiments

		Attributes			Missing
Data Set	Size	Discrete	Continuous	Classes	Values (%)
Adult	48.842	8	6	2	0.95
Letter	20.000	0	16	26	0
Nursery	12.960	8	0	5	0
Shuttle	58.000	0	9	7	0
Waveform	5.000	0	40	3	0

again. Accuracy results for this classifier are obtained using 10-fold cross validation. In our approach, C4.5 was also used for learning the global classifiers at each node.

Apart from classification accuracy, a measure of time that the global model takes to be trained is also calculated. This allows the comparison of our approach's complexity with Stacking. Majority Voting doesn't involve training at the classifier combination phase.

In order to obtain realistic results, the whole experiment described above is performed 10 times and the average of the partial output of each run is calculated. The final results on the five data sets are described in Table 2.

Table 2. Average accuracy and training time of experiments

		Accuracy (%)			Time (sec.)		
Data Set	D.N.	S	DS	MV	S	DS	MV
Adult	10	85.54	85.15	85.09	15	11	-
	20	85.19	84.86	84.72	21	11	-
	40	84.80	84.48	84.36	33	12	-
	80	84.50	83.98	82.64	57	12	-
Letter	10	70.41	75.80	77.21	45	14	-
	20	69.32	72.65	74.46	88	14	-
	40	69.12	68.56	71.72	171	16	-
	80	69.88	62.42	69.18	359	22	-
Nursery	10	95.78	94.90	94.61	4	2	-
	20	95.69	93.72	93.59	6	2	-
	40	95.49	92.70	90.28	11	2	-
	80	96.37	90.01	89.39	20	2	-
Shuttle	10	99.76	99.70	99.63	20	6	-
	20	99.76	99.65	99.57	37	6	-
	40	99.76	99.43	99.50	76	7	-
	80	99.72	99.39	99.47	156	8	-
Waveform	10	75.04	80.43	79.72	2	2	-
	20	73.55	80.84	80.62	3	2	-
	40	72.73	81.99	81.92	4	2	-
	80	72.83	82.66	82.18	6	2	-

The first column provides the name of the data sets, while the next one gives the number of distributed data sets and thus classifiers. The next two groups of columns provide information about the accuracy of the final global model in percent and the time to train the global model in seconds. Each group of columns gives comparative results with respect to each of the three methodologies, Stacking (S), our approach (DS) and Majority Voting (MV).

As the results show, our approach exhibits better performance than Majority Voting in 14 of the 20 experiments. Majority Voting was better in 2 experiments with the Shuttle data and 4 experiments with the Letter data. In 2 of those it was also better than Stacking. This verifies the fact that our approach, as a methodology that uses training at the meta-level, achieves better accuracy than Majority Voting. The differences might appear small in some cases, but in this paper we are primarily interested in performance with respect to time. Therefore, we used C4.5 in the implementation of Stacking and our approach for simplicity instead of the more efficient MLR.

However, the predictive performance of our approach is worse than that of Stacking in 14 of the 20 experiments. This comes at no surprise as Stacking uses the full posterior probabilities of the local models' predictions, while our alternative uses only the averages of the predictions according to the class. This way we loose information and the derived global models are inferior in accuracy in comparison with the models that Stacking produces.

What we loose in accuracy however, we gain in time. Our approach has a computational complexity that is independent of the number of distributed nodes. This way it achieves tractability of building the final model in contrast to Stacking that requires a great amount of time for training at the combination phase.

This is also evident from the plots in Figures 1 and 2, which depict the relationship between the necessary time to train the global model in the experiments for Stacking and our approach. Figure 1 shows that the time to build the global model increases linearly with respect to the number of distributed sites for each data set. This occurs because when the probability distribution of the local classifiers output is used, the attribute vector in the global learning phase increases by as many attributes as the classes in the domain problem. The attribute vector in a problem with C classes and N distributed nodes has size equal to $C*N$. Therefore, depending on the algorithm used, the complexity of the learning problem at the combination phase is related to the product $C*N*L$, where L is the size of the meta-level training set.

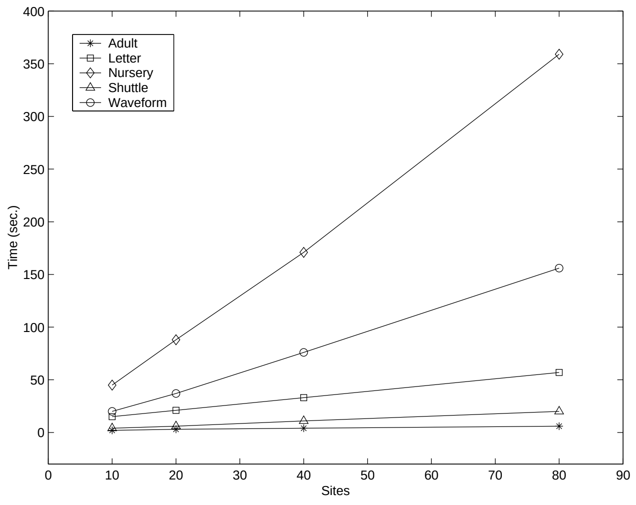

Figure 1. Time vs. distributed nodes for Stacking

Figure 2 shows that the time to build the global model stays in most cases almost constant with respect to the number of distributed sites for each data set. When the averages of the posterior probabilities of the local classifiers output are used, the attribute vector in the global learning phase is constant and equal to the classes in the domain problem. The size of the attribute vector in a problem with C classes and N distributed nodes will always be equal to C. Therefore, depending on the algorithm used, the complexity of the learning problem at the combination phase is related to the product $C*L$, where L is the size of the meta-level training set. The variation present in the Letter and Shuttle data sets is due to the fact that the calculation of averages also involves time. This however is very small compared to the time needed for global learning.

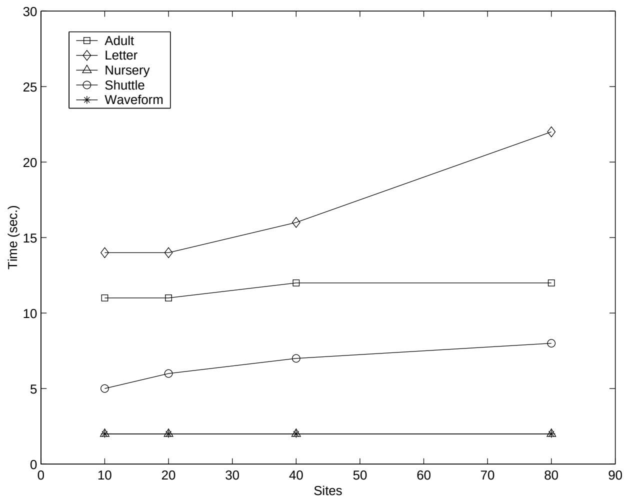

Figure 2. Time vs. distributed nodes for our approach

6 Conclusions

This work presented a framework for Distributed Stacking of multiple classifiers that offers two key advantages: i) It does not require raw data being moved around the distributed nodes and ii) It scales up well with respect to large number of classifiers. Our framework is based on local learning and model Stacking using the average probability distribution of the local classifiers' output according to the class as input to the second level classifier. In addition, Stacking takes place in parallel at all distributed nodes using all local classifiers apart from the one that was derived from each node's data. The final result derives from combining the induced global models using the Sum rule.

In the future we plan to explore the effectiveness of other combination strategies, for example using other rules than Sum and possibly including the original data at the meta-level training instances. We also intend to experiment with different learning algorithms at the meta-level than C4.5, for example fuzzy learning algorithms, which could potentially improve the predictive accuracy.

The implementation of our methodology is currently executed on a single machine, where we sequentially process the parts of a large data set. The next step is to implement a distributed computing architecture that will allow the parallel execution of the local and global model learning phases. This will speed up the execution of experiments, allow the calculation of time regarding the actual model communication process and bring the implementation closer to being a usable system rather than an experimental platform.

Furthermore, we plan to investigate the possibility of using the growing standard of PMML [4] as the model description language in our system implementation. This will add to the independence of the actual learning algorithms used for local and global learning phases and the compatibility with respect to other data mining systems, provided that they also use the same standard.

ACKNOWLEDGEMENTS

We would like to thank the two anonymous referees for their comments which helped improve this paper.

REFERENCES

[1] Catherine L. Blake and Christopher J. Merz. UCI repository of machine learning databases.
http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[2] Philip Chan and Salvatore Stolfo, 'Meta-learning for multistrategy and parallel learning', in *Proceedings of the Second International Workshop on Multistrategy Learning*, (1993).

[3] Winston Davies and Pete Edwards, 'Dagger: A new approach to combining multiple models learned from disjoint subsets', *Machine Learning*, (2000).

[4] The Data Mining Group. An introduction to pmml 2.0.
http://www.dmg.org/pmmlspecs_v2/pmml_v2_0.html, 2001.

[5] Yike Guo and Janjao Sutiwaraphun, 'Probing knowledge in distributed data mining', in *Proceedings of the PAKDD'99 Conference*, Beijing, China, (1999).

[6] Lawrence Hall, Nitesh V. Chawla, and Kevin W. Bowyer, 'Decision tree learning on very large data sets', in *Proceedings of the IEEE SMC Conference*, pp. 2579–2584, San Diego, California, (1998).

[7] Fumitaka Kimura and Malayappan Shridhar, 'Handwritten numerical recognition based on multiple algorithms', *Pattern Recognition*, **24**(10), 969–983, (1991).

[8] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas, 'On combining classifi ers', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–238, (March 1998).

[9] Andreas Prodromidis, Philip Chan, and Salvatore Stolfo, 'Meta-learning in distributed data mining systems: Issues and approaches', in *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, (2000).

[10] Ross J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, San Mateo, 1993.

[11] Kai Ming Ting and Ian Witten, 'Stacked generalization: When does it work?', in *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pp. 866–871, (1997).

[12] Kai Ming Ting and Ian Witten, 'Stacking bagged and dagged models', in *Proceedings of the 14th International Conference on Machine Learning*, pp. 367–375, (1997).

[13] Graham John Williams, *Inducing and Combining Multiple Decision Trees*, Ph.D. dissertation, Australian National University, 1990.

[14] David Wolpert, 'Stacked generalization', *Neural Networks*, **5**, 241–259, (1992).