2nd Hellenic Conf. on AI, SETN-2002, 11-12 April 2002, Thessaloniki, Greece,
Proceedings, Companion Volume, pp. 249-256

Distributed Data Mining of Large Classifier Ensembles

Grigorios Tsoumakas and Ioannis Vlahavas

Dept. of Informatics, Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece
{greg, vlahavas}@csd.auth.gr

Abstract. Nowadays, classifier ensembles are often used for distributed data mining in order to discover knowledge from inherently distributed information sources and scale up learning to very large databases. One of the most successful methods used for combining multiple classifiers is Stacking. However, this method suffers from very high computational cost in the case of large number of distributed nodes. This paper presents a new classifier combination strategy that scales up efficiently and achieves both high predictive accuracy and tractability of problems with high complexity. It induces a global model by learning from the averages of the local classifiers' output. This way, fast and effective combination of large number of classifiers is achieved.

1 Introduction

Classification is the most investigated machine learning task in the last 15 years [7] with numerous algorithms and a variety of application domains. Classifier Ensembles [3] is a rather recent subarea of machine learning that has been mainly used for increasing the predictive accuracy of single classifiers, scaling up learning algorithms to very large data sets and learning from inherently distributed data. It has been applied with success to a number of applications, playing an important role to new research areas like Distributed Data Mining, Multiple Classifier Systems and Information Fusion.

A particular approach that has been successfully applied to Distributed Data Mining is Meta-Learning [2]. It is based on the methodology of Stacked Generalization, also found as Stacking in the literature, originally developed by Wolpert [9]. Meta-Learning involves learning a global classifier that models the way that the output of many local classifiers correlates with the true class.

The accumulation of huge amounts of data and the increased networking of information systems in our times, raises the question of how well can Stacking scale up to large number of distributed nodes. This paper identifies a key problem towards this direction and presents an alternative classifier combination strategy that induces a global model by learning from the averages of the local classifiers' output. This way, fast and effective distributed data mining of large classifier ensembles is achieved.

The rest of the paper is organized as follows. In Section 2, we review related methodologies on combining multiple classifiers for distributed data mining. In

Section 3, we identify the scaling problem of Stacking and present a new classifier combination strategy to deal with it. In Section 4 we present and discuss comparative experimental results, and finally in Section 5 we conclude and pose future research directions.

2 Related Work

The simplest method to combine a number of classifiers is majority voting. The combined classification, outputs the class that gets the most votes from the ensemble. An improved version of this method is to sum the posterior probabilities of all the classifiers with respect to all classes and output the class with the biggest sum. The disadvantage of majority voting is that if more than half of the classifiers output a bad class then the result will be incorrect.

To deal with this deficiency, approaches that learn how to combine the output of a number of classifiers were introduced. Stacked Generalization [9] is a method that combines multiple classifiers by learning the way that their output correlates with their true class on an independent set of instances. In the first step, N classifiers Ci, i = 1..N are induced from each of N data sets Di, i = 1..N. Then, for every instance ej, j = 1..L of an evaluation set E, independent of the Di data sets, the output of the classifiers Ci(ej) along with the true class of the instance class(ej) are used to form an instance mj, j = 1..L of a new data set M. Each instance will be of the form C1(ej), C2(ej), ..., CN(ej), class(ej). In the last step, a global classifier GC is induced from M.

Any algorithm suitable for classification problems can be used for learning the Ci and GC classifiers. Independence of the actual algorithm used for learning Ci, is actually one of the advantages of this method, 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 [8] 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.

Chan and Stolfo [2], applied the concept of Stacked Generalization to distributed data mining, via their Meta-Learning methodology. They focused on combining distributed disjoint data sets and investigated various schemes for structuring the meta-level training examples. They found the best one to be using the dominating predictions of the distributed classifiers along with the correct class. They also showed that Meta-Learning exhibits superior 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 [5]. However, this method has the disadvantage that the meta-level model is not comprehensible. It describes the way base classifiers correlate with the correct class and not the way data correlate with the correct class. In addition it does not scale efficiently to domains with large number of classes and distributed nodes.

Knowledge Probing [4] builds on the idea of meta-learning and in addition uses an independent data set, called the probing set, in order to discover knowledge about the data. 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 it suffers from the same problems in scaling up.

3 Scaling Up to Large Classifier Ensembles

As discussed in Section 2, stacking uses the output of a number of classifiers on an independent evaluation data set to form a data set of meta-level instances. The number of attributes in a meta-level instance is equal to the number of participating classifiers in the ensemble. Therefore, the complexity of learning the global classifier from the meta-level data set is also proportional to the ensemble size.

Furthermore, Ting and Witten [8] have showed that stacking is effective when the distribution of the posterior probability of the classifier estimates with respect to all classes is used. This actually worsens the complexity of learning the global classifier, because the number of attributes becomes the product of the number of participating classifiers and the number of classes in the domain.

The two points mentioned above led us to the conclusion that there is a problem in scaling up stacking, especially for domains with many classes. To exemplify this, consider combining 1.000 classifiers for the task of English letter recognition. In this domain there are 26 classes, the letters A to Z. Stacking in this particular example will produce meta-level instances with 26.000 attributes. It is practically impossible to learn a global model from such a data set within acceptable time for a class of algorithms including the most successful for stacking MLR.

3.1 Our Approach

We propose a new strategy for multiple classifier combination that deals with the above-mentioned problem of stacking, while keeping the advantage of modeling complex classifier ensemble behavior. The main idea is forming the meta-level training examples using the average distribution of the posterior probability of the classifier estimates with respect to all classes and then learning a final classifier out of it as before.

Consider a classification task with C classes, and an ensemble of N classifiers $C_i,\ i = 1..N$, that we want to combine using our strategy on an independent evaluation data set E. Then, for every instance $e_j,\ j = 1..size(E)$, we calculate the probability distribution of each classifier's C_i output with respect to all classes,

$PD_i = [P_{i1}\ P_{i2}\ \ldots\ P_{iC}]$, where P_{ik} is the posterior probability that classifier C_i outputs class k. Finally, we average the PD_i over all classifiers and get a meta-level training example $m_j = [(D_{11} + \ldots + D_{N1})/N\ \ldots\ (D_{1C} + \ldots + D_{NC})/N]$.

The most important advantage is that the size of the attributes 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 way, we achieve tractability of modeling the ensemble behavior in the case of large classifier ensembles.

In the previous example of English letter recognition, the combination of 1000 classifiers with the new strategy will produce meta-level instances with only 26 attributes. Compared to the 26000 (or 1000 if distribution is not used) attributes in stacking, it is obvious that there is a huge difference in the required computational resources to learn the global model.

The disadvantage is that we lose the fine grain modeling of how each classifier contributes to the prediction of stacking. Instead, we get a coarser model of how the classifiers behave as an ensemble. But this is necessary to counter the complexity in the large-scale scenario.

However, since learning is used to produce the final model, the methodology can still model complex concepts and give correct classification even when more than half of the participating classifiers output an erratic decision.

4 Experimental Results

In order to evaluate the proposed strategy, a set of experiments that compared it to stacking and majority voting were conducted using four of the largest data sets from the UCI Machine Learning Repository [1]. The data sets were split into a variable number of parts to simulate a distributed environment. The names and details of these data sets are described in Table 1.

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

The setup of the experiments is the following. Initially, the original data set is randomly split into a percentage (25%) of evaluation data and the rest (75%) is randomly split again into a variable number (10, 20, 50 and 100) of distributed data sets. C4.5 [6] is then used to learn a local predictive model from each one of the distributed data sets. Next, the predictions of all local models

are combined to form the meta-level training examples with both the traditional stacking strategy (using the maximum sum of probabilities) and the proposed one. Finally, a global classifier is induced from these examples using the C4.5 algorithm again.

Accuracy measurements for this classifier are obtained using 10-fold cross validation. Moreover, the whole experiment described above is performed 10 times and the final results derive from averaging the partial results of each run. This way realistic results are obtained.

Apart from the accuracy of the classification, a measure of time is also recorded. This is the average time to create the meta-level training examples (in one run of the experiment) and train the global classifier (in one fold of the cross validation). Furthermore, in order to get a machine independent measure of the computational complexity of each strategy, the size of the meta-level data set is also recorded. This can be especially helpful in cases where the large number of distributed nodes and classes makes the experiments too time consuming (recall that global training is performed 100 times in each experiment to get realistic results) to be completed.

The final results on the four data sets are described in Table 2. The first three columns provide useful information about the data sets, while the next one gives the number of distributed data sets and thus classifiers. The next three group of columns provide information about the meta-level training set size in megabytes, the accuracy of the final global model in percent and the time in seconds. Each group of columns gives comparative results with respect to each of the three strategies, namely stacking (S), the proposed average stacking (AS) and the classic majority voting (MV).

First of all, we can see that with respect to time, our approach has a similar performance to majority voting. Its fast and has constant complexity with respect to the number of classes and linear complexity with respect to the number of distributed nodes, as mentioned in the theoretical analysis of the strategy earlier in this paper. In contrast, stacking has linear complexity with respect to the number of distributed nodes, but also linear complexity with respect to the number of classes. This is why the largest training time was seen in the letter data set that has 26 classes. The same conclusion can be drawn, by looking at the size of the meta-level training examples, as the dominating time in an experiment is the time to train the global classifier from the meta-level training examples.

Regarding the accuracies, there is no consistent pattern that can be derived from all the experiments. However, a first general pattern that can be noticed is that as the number of distributed sites increases, the performance of all the strategies decreases. This is obviously due to the fact that local classifiers are trained on a few data.

Moreover, our approach is better than majority voting and worse than stacking in most experiments excluding the letter dataset. In some cases it exhibits better performance even from stacking (shuttle, 10 and 100 nodes), while in other cases it performs worse than majority voting (adult, 20 and 50 nodes).

Table 2. Details of experimental results

		Meta Size (Mb)			Accuracy (%)			Time (sec.)		
Data Set	D.N.	S	AS	MV	S	AS	MV	S	AS	MV
Adult	10	1.05	0.14	-	85.46	85.17	85.09	16	11	10
	20	1.95	0.14	-	85.35	84.31	84.55	22	11	10
	50	4.85	0.14	-	84.92	84.18	84.31	42	11	11
	100	9.15	0.14	-	84.44	84.17	83.23	79	12	11
Letter	10	1.50	0.25	-	71.09	71.21	78.54	49	7	2
	20	2.85	0.30	-	70.42	69.38	75.89	92	9	3
	50	6.97	0.41	-	70.41	66.99	72.09	239	12	6
	100	14.54	0.49	-	70.81	65.40	69.78	512	17	9
Nursery	10	0.23	0.05	-	94.15	93.35	92.60	4	2	2
	20	0.49	0.05	-	93.98	93.14	90.47	5	2	2
	50	1.17	0.05	-	93.42	92.38	87.47	12	2	2
	100	2.50	0.05	-	95.05	92.30	85.66	23	2	2
Shuttle	10	1.83	0.29	-	99.72	99.80	99.61	20	6	3
	20	3.56	0.34	-	99.72	99.69	99.54	34	6	4
	50	7.84	0.33	-	99.69	99.66	99.44	85	7	5
	100	13.85	0.35	-	99.64	99.72	99.39	198	9	6

Actually, one can consider our approach as standing in the middle of majority voting and stacking in terms of predictive power, and thus such performance was expected.

However, the difference in accuracy is small in comparison to the difference in time. Our approach achieves tractability in the case of large number of classifiers and classes in comparison to stacking and increased accuracy in comparison to majority voting. It can solve problems that stacking would not be able to solve within acceptable time and achieves that with higher accuracy than majority voting.

The only exception in the general patterns that were derived from the experimental results, is the letter dataset. In this dataset majority voting performs much better than the other two methods with the exception of the case of 100 nodes, where stacking exhibited the best accuracy, although within unacceptable time. The difference of the letter data set with the other data sets is the much larger number of classes. Further investigation of this issue should be done to check whether there is a consistent pattern in the case of very large classes.

5 Conclusion and Future Work

This paper presented a new strategy for multiple classifier combination, that exhibits high classification accuracy and low computational complexity. It is therefore very useful for distributed data mining from large classifier ensembles.

Currently we are working on an expansion of this work aiming to take under consideration issues such as resource and location constraints, that often appear in high performance distributed data mining. Moreover, we plan to test soft aggregation operators, such as max and min, for the combination of the local classifiers estimates, and in addition use a fuzzy learning algorithm for the final step of global classifier induction. Preliminary results from earlier experiments towards this end, were promising.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proceedings of the Second International Workshop on Multistrategy Learning*, 1993.
3. Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, 2000.
4. Y. Guo and J. Sutiwaraphun. Probing knowledge in distributed data mining. In *Proceedings of the PAKDD'99 Conference*, Beijing, China, 1999.
5. A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches, 2000.
6. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo, 1993.
7. L. Saitta. Machine learning: A technological roadmap. Technical report, University of Amsterdam, 2000.
8. K. Ting and I. Witten. Stacked generalization: When does it work?, 1997.
9. David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.