Scaling Learning by Meta-Learning over Disjoint and Partially Replicated Data

Philip K. Chan
Computer Science
Florida Institute of Technology
Melbourne, FL 32901
pkc@cs.fit.edu

Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, NY 10027
sal@cs.columbia.edu

Abstract

Many existing learning algorithms assume that the entire data set fits into main memory, which is not feasible for massive amounts of inherently distributed data. One approach we explore to handling a large data set is to partition the data into disjoint subsets, run the learning algorithm on each of these subsets of data, and combine the results by some means. The results achieved to date show promising results, but in nearly all cases accuracy of the final combined classifier is not as great as a single classifier computed from the entire data set. In this paper we evaluate our approach, called *meta-learning*, to learning from partitioned data where we relax the restriction that each subset of training data is disjoint, i.e. some amount of replication of training data is allowed. We anticipated data replication could improve overall accuracy, however, our findings suggest the contrary.

1 Introduction

With the coming age of very large network computing, it is likely that orders of magnitude more data in databases will be available for various learning problems of real world importance. Many existing learning algorithms require all the data to be resident in main memory, which is clearly untenable in many realistic databases. In certain cases, data is inherently distributed and cannot be localized on any one machine (for competitive business reasons, for instance). In such situations, it may not be possible, nor feasible, to inspect all of the data at one processing site to compute one primary "global" classifier.

Popular incremental learning algorithms such as [7, 10], aim to solve the scaling problem by piecemeal processing of a large data set that is consumed in a sequential fashion. Others have studied approaches based upon direct parallelization of a learning algorithm run on a centralized multiprocessor with "scalable" main-memory resources [13]. A review of such approaches has appeared elsewhere [5]. An alternative approach we study here is to apply *data reduction* techniques, where one may partition the data into a number of smaller *disjoint* training subsets, apply some learning algorithm on each subset (perhaps all in parallel), followed by a phase that combines the learned results in some principled fashion.

In such schemes one may logically presume that accuracy will suffer; i.e., combining results from a large number of separately learned classifiers trained on disjoint data may not be as accurate as learning one global classifier trained on the entire data set. Some amount of important information may be lost when reducing the data. Our proposed *meta-learning* techniques, which were first presented in [4], involves applying a learning algorithm to sets of predictions (treated as training data) made by a set of base classifiers in order to *learn how to combine* their collective predictions. Comparative results from applying meta-learning and other voting based strategies to learning from disjoint subsets were discussed in [5].

In this paper we provide an evaluation of our techniques and another published method on combining classifiers learned from partitioned subsets where some amount of *replication* is allowed. We believe this situation might be common in certain real world

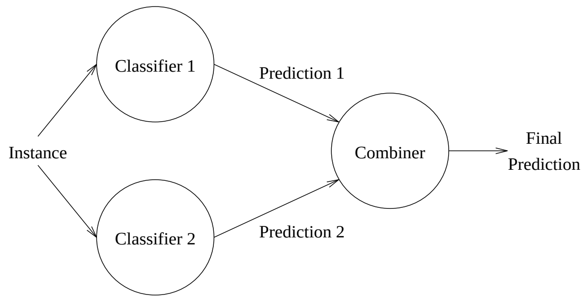

Figure 1: A combiner with two classifiers.

contexts. In situations where very large inherently distributed data exist, some amount of common information may be inevitable. Alternatively, in an attempt to boost predictive accuracy and to reduce inherent bias introduced by disjoint training data one may wish to purposefully replicate some amount of common training data. Howerer, this would incur higher computational cost. The question is, therefore, whether replication of training data buys anything?

2 Meta-learning Techniques

Our approach to combining multiple classifiers is to *meta-learn* a set of new classifiers, or *meta-classifiers*, whose training data are derived from a set of predictions generated by a set of base classifiers. Our techniques fall into two general categories: the *arbiter* and *combiner* strategies. Due to space limitations, only the combiner strategies are discussed here. Detailed descriptions of the *arbiter* strategies can be found in [3].

Combiner In the *combiner* [3] strategy, the predictions of the learned base classifiers on the training set form the basis of the meta-learner's training set. A *composition rule*, which varies in different schemes, determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier, that we call a *combiner*. In classifying an instance, the base classifiers first generate their predictions, and then a combiner produces a final prediction (see Figure 1). The aim of this strategy is to combine the predictions from the base classifiers by learning a relationship or function between base predictions and the correct classification.

We experimented with two schemes for the composition rule. First, the predictions, $C_1(x)$, $C_2(x)$, ... $C_k(x)$, for each example x in the validation set of examples, E, are generated by the k base classifiers. These predicted classifications are used to form a new set of "meta-level training instances," T, which is used as input to a learning algorithm that computes a combiner. The manner in which T is computed varies as defined below. In the following definitions, $class(x)$ and $attribute\_vector(x)$ denote respectively the correct classification and the entire set of attribute values of tuple x, where x is a member of the validation set, E.

1. Return meta-level training instances with the correct classification and the predictions; i.e., $T = \{(class(x), C_1(x), C_2(x), ...C_k(x)) \mid x \in E\}$. This scheme was also used by Wolpert [11]. (For further reference, this scheme is denoted as *class-combiner*.)

2. Return meta-level training instances as in *class-combiner* with the addition of the attribute vectors; i.e., $T = \{(class(x), C_1(x), C_2(x), ...C_k(x), attribute\_vector(x)) \mid x \in E\}$. (This scheme is denoted as *class-attribute-combiner*.)

The next section discusses our findings from experiments where a controlled amount of "arbitrary replication" is permitted in each subset of data used to train the base classifiers.

3 Learning Tasks and Experimental Setup

Two decision-tree inductive learning algorithms were used in our experiments: ID3 [8] and CART [1], both obtained from NASA Ames Research Center in the IND package [2]. Two data sets were used in our studies. The DNA splice junction (SJ) data set[1] [9] contains 3,190 training instances of nucleotides and the type of splice junction, if any, at the center of each sequence. There are three possible junctions, and hence 3 categorical classes in this task. The protein coding region (PCR) data set[2] [6] contains DNA nucleotide sequences and their binary classifications (coding or non-coding). The PCR data set has 20,000 sequences. These two data sets represent two different kinds of learning tasks: one is difficult to learn (PCR at 70+% accuracy for over 20,000 training examples) and the other is easy to learn (SJ at 90+% accuracy for 3190 examples).

We varied the number of equi-sized subsets of training data from 2 to 64 ensuring each was disjoint but

[1] Courtesy of Towell, Shavlik and Noordewier.

[2] Courtesy of Craven and Shavlik.

with a distribution of examples of each class proportional to that of the entire database. We measure the overall predictive accuracy as the ratio of correct classifications over the total number of samples in a *test set* which is disjoint from the training data. Results presented in the following section are averages over 10-fold cross validation experiments.

Since the data we are dealing with here is relatively small, we have the opportunity to evaluate the various proposed schemes by comparing their accuracy against a "global classifier," i.e. a classifier that is learned from the entire data set. We call the accuracy of this global classifier produced by ID3 and CART the "baseline case," which is plotted in our graphs as the "one subset" case on the X-axis. In addition to our meta-learning strategies, a Bayesian statistical approach (*bayesian-belief*) as presented in [12] was also used in our experiments to combine classifiers.

4 Experimental Results on Replicated Data

In our experiments each partition of data allowed some amount of replication. We prepare each learning task by generating subsets of training data for the base classifiers according to the following generative scheme.

1. Starting with N disjoint subsets, randomly choose from any of these sets one example X, distinct from any other previously chosen in a prior iteration.

2. Randomly choose a number r from $1 \ldots (N-1)$, i.e. the number of times this example will be replicated.

3. Randomly choose r subsets (not including the subset from which X was drawn) and assign X to those r subsets.

4. Repeat this process until the size of the largest (replicated) subset is reached to some maximum (as a percentage, Δ, of the original training subset size).

In the experiments reported here, Δ ranged from 0% to 30%. Each set of incremental experimental runs, however, chooses an entirely new distribution of replicated values. No attempt was made to maintain a prior distribution of training data when incrementing the amount of replication. This "shot gun" approach provides us with some sense of a "random learning problem" that we may be faced with in real world scenarios where replication of information is likely inevitable or purposefully orchestrated.

The graphs in Figure 2 plot the results for only the *class-combiner* and *bayesian-belief* strategies. Results from other strategies are not shown. The results in all cases are conclusive: *replication essentially buys nothing!* In each case no measurable improvement in predictive accuracy is seen no matter which learning algorithm or combining scheme is used.

These negative results for replication are in fact positive from the perspective of computational performance! One may presume that applying a number of instances of a learning algorithm to disjoint training data results in a set of base classifiers each biased towards its own partition of data. Combining two or more such biased base classifiers by meta-learning attempts to share knowledge among the base classifiers and to reduce each individual's bias. Replication of training data is an alternative attempt to reduce this bias. Common or shared information replicated across subsets of training data at the onset of learning attempts to provide each learned base classifier with a "common view" of the learning task. The results here show that meta-learning from disjoint training data does an effective job of sharing knowledge among separate classifiers anyway. In fact, the overhead that may be attributed to replicated data (since the same data is being treated multiple times by separate learning processes) may be comfortably avoided, i.e. meta-learning on purely disjoint data seems to achieve good performance, at perhaps optimal speeds due to optimal data reduction.

These rather surprising results are of course limited to the learning algorithms and data sets used in this study. Additional experiments on other learning tasks are required to confirm this behavior. Even so, these two "arbitrarily chosen" tasks have provided impressive evidence of this interesting behavior.

5 Concluding Remarks

We have demonstrated that meta-learning over base classifiers trained on disjoint data is not measurably improved by schemes that attempt to replicate common information for initial training of the base classifiers. From a practical perspective this implies we can comfortably apply meta-learning techniques to disjoint partitions of data to maximize computational performance without a substantive negative impact on predictive accuracy.

We are exploring the use of meta-learning in improving the accuracy performance of local learned models by merging them with ones imported from remote sites. That is, at each site, learned models from other sites are also available; however, raw data are not shared among sites. This happens when companies are willing to share "black-box" models, but not the proprietary raw data from which those models are derived for competitive reasons. Futhermore, we investigate the effects on local accuracy when the local underlying training data overlap with those at remote sites. Studies in measuring the independence among base classifiers and simplfying the final meta-learned structure by pruning related base classifiers are also underway.

Acknowledgment

This work was performed at Columbia University and has been partially supported by grants from New York State Science and Technology Foundation, Citicorp, and NSF grant IRI-94-13847. The first author would also like to thank Florida Tech for the usage of its resources.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[2] W. Buntine and R. Caruana. *Introduction to IND and Recursive Partitioning.* NASA Ames Research Center, 1991.

[3] P. Chan and S. Stolfo. Experiments on multistrategy learning by meta-learning. In *Proc. Second Intl. Conf. Info. Know. Manag.*, pages 314–323, 1993.

[4] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. on Multistrategy Learning*, pages 150–165, 1993.

[5] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. Twelfth Intl. Conf. Machine Learning*, pages 90–98, 1995.

[6] M. Craven and J. Shavlik. Learning to represent codons: A challenge problem for constructive induction. In *Proc. IJCAI-93*, pages 1319–1324, 1993.

[7] J. R. Quinlan. Induction over large data bases. Technical Report STAN-CS-79-739, Comp. Sci. Dept., Stanford Univ., 1979.

[8] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[9] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proc. AAAI-90*, pages 861–866, 1990.

[10] J. Wirth and J. Catlett. Experiments on the costs and benefits of windowing in ID3. In *Proc. Fifth Intl. Conf. Machine Learning*, pages 87–99, 1988.

[11] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[12] L. Xu, A. Krzyzak, and C. Suen. Methods of combining multiple classifires and their applications to handwriting recognition. *IEEE Trans. Sys. Man. Cyb.*, 22:418–435, 1992.

[13] X. Zhang, M. Mckenna, J. Mesirov, and D. Waltz. An efficient implementation of the backpropagation algorithm on the connection machine CM-2. Technical Report RL89-1, Thinking Machines Corp., 1989.

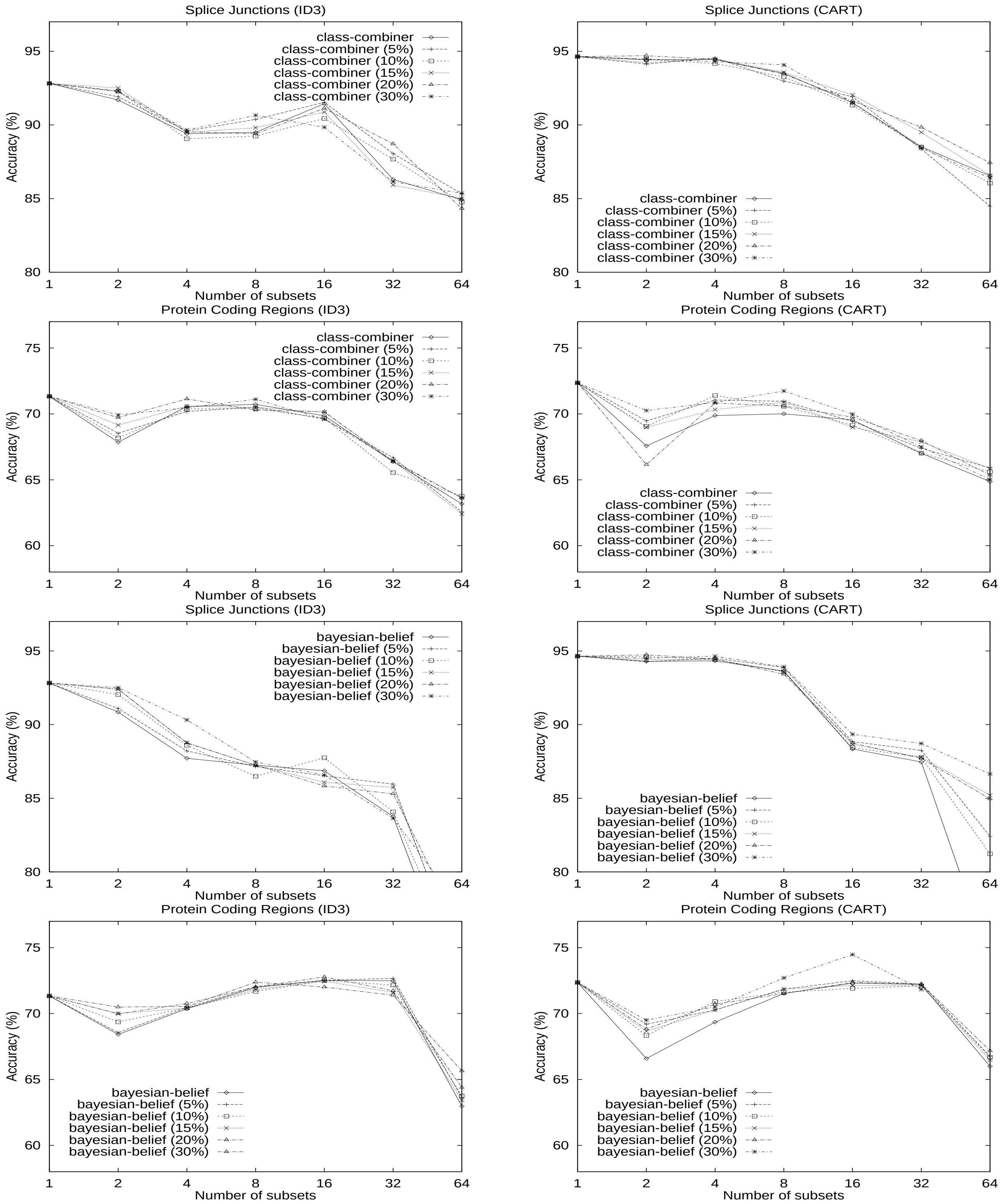

Figure 2: Accuracy for combining techniques trained over varying amounts of replicated data. Δ ranges from 0% to 30%.