Sharing Learned Models among Remote Database Partitions by Local Meta-learning

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

We explore the possibility of importing "black-box" models learned over data sources at remote sites to improve models learned over locally available data sources. In this way, we may be able to learn more accurate knowledge from globally available data than would otherwise be possible from partial, locally available data. Proposed meta-learning strategies in our previous work are extended to integrate local and remote models. We also investigate the effect on accuracy performance when data overlap among different sites.

Introduction

Much of the research in inductive learning concentrates on problems with relatively small amounts of data residing at one location. With the coming age of very large network computing, it is likely that orders of magnitude more data in databases at various sites will be available for various learning problems of real world importance. Frequently, local databases represent only a partial view of all the data globally available. For example, in detecting credit card fraud, a bank has information on its credit card transactions, from which it can learn fraud patterns. However, the patterns learned may not represent all of the fraud patterns found in transactions at other banks. That is, a bank might not know a fraud pattern that is prevalent at other banks.

One approach to solving this problem is to merge transactions from all databases into one database and locate all the fraud patterns. It is not uncommon that a bank has millions of credit card transactions; pooling transactions from all banks will create a database of enormous size. Learning fraud patterns from millions of transactions already poses significant efficiency problems; processing transactions gathered from all banks is likely infeasible. In addition, transactions at one bank are proprietary; sharing them with other banks means giving away valuable customer purchasing information. Exchanging transactions might also violate customers' privacy.

Another solution is to share the fraud patterns instead of the transaction data. This approach benefits from a significant reduction of information needed to be merged and processed. Also, proprietary customer transaction information need not be shared. You might now ask that if the data are proprietary, the fraud patterns can also be proprietary. If the patterns are encoded in programs, the executables can be treated as "black boxes." That is, by sharing the black boxes, one doesn't have to worry about giving away valuable and proprietary information. The next question is how we can merge the black boxes.

We adopted the general approach of *meta-learning* (Chan & Stolfo 1993) and developed techniques for coalescing multiple learned models. During meta-learning, the learned models are treated as black boxes so that they can use any representation and can be generated by any inductive learning algorithm. That is, our meta-learning techniques are representation- and algorithm-independent. In this paper we explore the use of meta-learning to improve the accuracy performance of locally learned models by merging them with ones imported from remote sites. That is, at each site, learned models from other sites are also available. Furthermore, we investigate the effects on local accuracy when the local underlying training data overlap with those at remote sites. This situation arises in practice (eg. a person may be a customer at several banks, and/or commit the same credit card fraud at different banks). In this paper we overview the concept of meta-learning and its techniques, followed by a discussion on how meta-learning can improve local learning. We then empirically evaluate local meta-learning and the effect of data replication on performance.

Meta-learning

Given a number of classifiers and their predictions for a particular unlabeled instance, one may combine them by picking the prediction with the largest number of votes. Our approach introduced in (Chan & Stolfo 1993) is to *meta-learn* a set of new classifiers (or *meta-classifiers*) whose training data are based on predictions of a set of underlying base classifiers. Re-

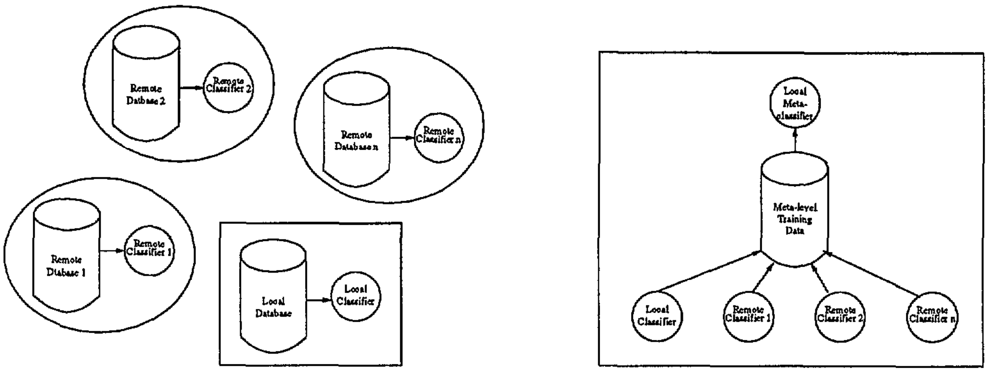


Figure 1: Local meta-learning at a site with three remote sites.

sults from (Chan & Stolfo 1995) show that our meta-learning techniques are more effective than voting-based methods.

Our techniques fall into two general categories: the *arbiter* and *combiner* schemes. We distinguish between *base classifiers* and arbiters/combiners as follows. A base classifier is the outcome of applying a learning algorithm directly to "raw" training data. The base classifier is a program that given a test datum provides a prediction of its unknown class. For purposes of this study, we ignore the representation used by the classifier (to preserve the algorithm-independent property). An arbiter or combiner, as described below, is a program generated by a learning algorithm that is trained on the predictions produced by a set of base classifiers and the raw training data. The arbiter/combiner is also a classifier, and hence other arbiters or combiners can be computed from the set of predictions of other arbiters/combiners.

An *arbiter* is learned by some learning algorithm to arbitrate among predictions generated by different base classifiers. That is, its purpose is to provide an alternate and more educated prediction when the base classifiers present diverse predictions. This arbiter, together with an *arbitration rule*, decides a final classification outcome based upon the base predictions. The arbiter is trained from examples that do not have a common prediction among the majority of the base classifiers. More details of this arbiter scheme are in (Chan & Stolfo 1995).

The aim of the *combiner* strategy is to coalesce the predictions from the base classifiers by learning the relationship between these predictions and the correct prediction. For example, a base classifier might consistently make the correct predictions for class c; i.e., when this base classifier predicts class c, it is probably correct regardless of the predictions made by the other base classifiers. In the *combiner* 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. The correct classification and predictions from the base classifiers constitute a training example in the *class-combiner* scheme. Attributes of the original example is added in the *class-attr-combiner* scheme. The details of these two schemes appear in (Chan & Stolfo 1995). 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. Based on the same composition rule, a new instance is generated from the predictions, which is then classified by the combiner. We note that a combiner computes a prediction that may be entirely different from any proposed by a base classifier, whereas an arbiter chooses one of the predictions from the base classifiers and the arbiter itself.

Local Meta-learning

Our previous work (Chan & Stolfo 1995) assumes a certain degree of "raw" data sharing. As we discussed earlier, situations might arise when data sharing is not feasible, but sharing of "black-box" learned models is possible. In this scenario a local site can "import" classifiers learned at remote sites and use them to improve local learning. The problem we face is how we can take advantage of the imported "black-box" classifiers. Our approach is to treat it as an integration problem and use meta-learning techniques to integrate the collective knowledge of the constituent classifiers.

Since only the local data set, called T_i, is available at site i, we are limited to that data set for meta-learning. A classifier, C_i, is trained from T_i locally and a set of classifiers, C_j where $j \neq i$, is imported from other sites $j, j = 1, .., n$. Using T_i, each C_j then generates predictions P_{ij} and C_i produces P_{ii}. P_{ij} and P_{ii} form the meta-level training set according to the strategies described earlier. That is, the local and remote classifiers are treated as base classifiers in our previous work. Once the meta-level training set is created, the corresponding meta-classifier is learned by applying a local machine learning algorithm to this new training set. Figure 1 depicts the relationship among various classifiers and sites during local meta-learning.

However, the predictions P_{ii} generated by the local classifier C_i on the local training set T_i will be more correct than the predictions, P_{ij}, generated by the remote classifiers because C_i was trained from T_i. As a result, during meta-learning, the trained meta-classifier will be heavily biased towards the local classifier (recall that the remote classifiers were not trained on the local data set T_i). For example, a local nearest-neighbor classifier can predict the local training set perfectly and the meta-learner will ignore all the remote classifiers. That is, we can't use the remote classifiers to improve local learning, which defeats the purpose of importing the remote classifiers.

To resolve this situation, at the local site, we partition T_i into two sets, T_{i1} and T_{i2}, from which classifiers C_{i1} and C_{i2} are trained. C_{i1} then predicts on T_{i2} and C_{i2} on T_{i1}. The union of the two sets of predictions form the predictions for the local classifier (P_{ii}). This method, called 2-fold cross validation partitioning, tries to approximate the behavior of C_i on unseen data. The process of obtaining the predictions P_{ij} from the remote classifiers remains unchanged. Now, during meta-learning, remote classifiers will not be automatically ignored since the local classifier is also judged on "unseen" data. The next section discusses our experimental evaluation of the local meta-learning approach.

Experimental Results

Four inductive learning algorithms were used in our experiments reported here: ID3 (Quinlan 1986), CART (Breiman *et al.* 1984), BAYES (described in (Clark & Niblett 1989)), and CN2 (Clark & Niblett 1989). ID3 and CART are decisions tree learning algorithms and were obtained from NASA Ames Research Center in the IND package (Buntine & Caruana 1991). BAYES is a simple Bayesian learning algorithm. CN2 is a rule learning algorithm and was obtained from Dr. Clark (Boswell 1990).

Four data sets were used in our studies. The DNA splice junction (SJ) data set (courtesy of Towell, Shavlik, and Noordewier (Towell, Shavlik, & Noordewier 1990)) contains sequences of nucleotides and the type of splice junction, if any, at the center of each sequence. *Exon-intron*, *intron-exon*, and *non-junction* are the three classes in this task. Each sequence has 60 nucleotides with eight different values per nucleotide (four base ones plus four combinations). The data set contains 3,190 training instances. The protein coding region (PCR) data set (courtesy of Craven and Shavlik (Craven & Shavlik 1993)) contains DNA nucleotide sequences and their binary classifications (*coding* or *non-coding*). Each sequence has 15 nucleotides with four different values per nucleotide. The PCR data set has 20,000 sequences. The secondary protein structure data set (SS) (Qian & Sejnowski 1988), courtesy of Qian and Sejnowski, contains sequences of amino acids and the secondary structures at the corresponding positions. There are three structures (*alpha-helix, beta-sheet, and coil*) and 20 amino acids (21 attributes, including a spacer (Qian & Sejnowski 1988)) in the data. The amino acid sequences were split into shorter sequences of length 13 according to a windowing technique used in (Qian & Sejnowski 1988). The SS data set has 21,625 sequences. The artificial (ART) data set has 10,000 instances randomly generated from a disjunctive boolean expression that has 4 symbolic (26 values) and 4 numeric (1,000 values) variables. A total of 4.6×10^{17} instances are possible.

To simulate the multiple-site scenario, we divided the training set into equi-sized subsets (each subset representing a site) and varied the number of subsets (sites) from 2 to 64. We also ensured that each subset was disjoint but with proportional distribution of examples of each class (i.e., the ratio of examples in each class in the whole data set is preserved). The *arbiter*, *class-combiner*, and *class-attribute-combiner* strategies were evaluated. The prediction accuracy on a separate test set is our primary comparison measure. The different strategies were run on the above four data sets, each with the above four learning algorithms and the results are plotted in Figure 2. Due to space limitations, only results from two data sets are shown; the rest appears in (Chan 1996). The plotted accuracy is the average accuracy of local meta-classifiers over 10-fold cross-validation runs. In each run, m sites generate m local classifiers and m local meta-classifiers, after "exchanging" all local classifiers. In the following performance graphs, *avg-base* denotes the average accuracy of the local/base classifiers as our standard base line. Statistical significance was measured by using the one-sided t-test with a 90% confidence value.

When compared to the base accuracy, at least one of the three local meta-learning strategies yields significantly higher accuracy in 13 out of the 16 cases (mostly at 4 or more subsets). Local meta-learning still has higher accuracy (not significantly) in 2 of the 3 remaining cases. Larger improvement usually occurs when the size of the local data set is smaller (the number of subsets/sites are larger). In many cases the arbiter strategy improves accuracy more than the two combiner strategies.

While many of the base classifiers drop in accuracy when the data set size gets smaller, some of the meta-learning strategies roughly maintain the same level of accuracy. One apparent example is the arbiter strategy using ID3 as the learner in the Coding Regions data set (top right graph in Figure 2). The arbiter strategy stays above 70% accuracy while the base accuracy drops to below 60%. The arbiter strategy maintains the accuracy in 8 out of 16 cases. For the Coding Regions data set, the arbiter strategy improves local learning by a wide margin using 3 of the 4 learners.

The results obtained here are consistent with those from non-local meta-learning (Chan & Stolfo 1995), where raw data can be shared among sites. Meta-learning improves accuracy in a distributed environment and the arbiter strategy is more effective than the two combiner techniques. Next, we investigate the effects on accuracy of local meta-learning when different sites posses some degree of common data.

Experimental Results on Data Replication

As we discussed previously in the introduction, different sites might have some overlapping data. To simulate this phenomenon, we allow some amount of replication in each partition of data. We prepare each learning task by generating subsets of training data for the local/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 $1...(N-1)$, 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 40%, with 10% increments. 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 same experimental setup was used as in the prior experiments. Results for the replicated data scenario using the class-combiner and class-attr-combiner strategies are plotted in Figure 3. Due to space limitations, only 8 of the 32 cases are shown, the rest appears in (Chan 1996). 7 out of 32 cases show significant accuracy improvement when the degree of replication increases; 6 of these 7 cases occur in the Coding Regions data set. 20 out of 32 cases show no significant accuracy changes across all subset sizes and degrees of replication. The remaining 5 cases have some significant accuracy improvement at certain subset sizes.

In summary, the majority does not show significant accuracy improvement when the degree of replication increases. This is contrary to one's intuition since one would expect the accuracy to increase when the local sites have a higher percentage of all the available data combined. That could imply that local meta-learning is quite effective in integrating models from remote sites without the help of replicated data. Our findings here are consistent with those from non-local meta-learning (Chan & Stolfo 1996).

Concluding Remarks

We have presented techniques for improving local learning by integrating remote classifiers through local meta-learning. Our experimental results suggest local meta-learning techniques, especially the arbiter strategy, can significantly raise the accuracy of the local classifiers. Furthermore, results from our data replication experiments suggest local meta-learning can integrate local and remote classifiers effectively without having a larger share of global data at a local site.

We are currently investigating a simplification process for reducing the complexity of the final meta-learned structures. Some classifiers could be strongly correlated and pruning some of them might not significantly change the performance of the entire structure. Finally, the meta-learning techniques reported in this paper form the basis of a system under development recently granted support by ARPA to learn fraud patterns in network-based financial information systems. The use of locally computed meta-classifiers over inherently distributed datasets of fraudulent transactions will provide an early-warning capability protecting against intruders and information warfare.

Acknowledgements

This work has been partially supported by grants from NSF (IRI-94-13847), NYSSTF, and Citicorp.

References

Boswell, R. 1990. *Manual for CN2 version 6.1.* Turing Institute. Int. Doc. IND: TI/MLT/4.0T/RAB/1.2.

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

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

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

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

Chan, P., and Stolfo, S. 1996. Scaling learning by meta-learning over disjoint and partially replicated data. In *Proc. Florida AI Research Symposium.* To appear.

Chan, P. 1996. *An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning.* Ph.D. Dissertation, Department of Computer Science, Columbia University, New York, NY. (forthcoming).

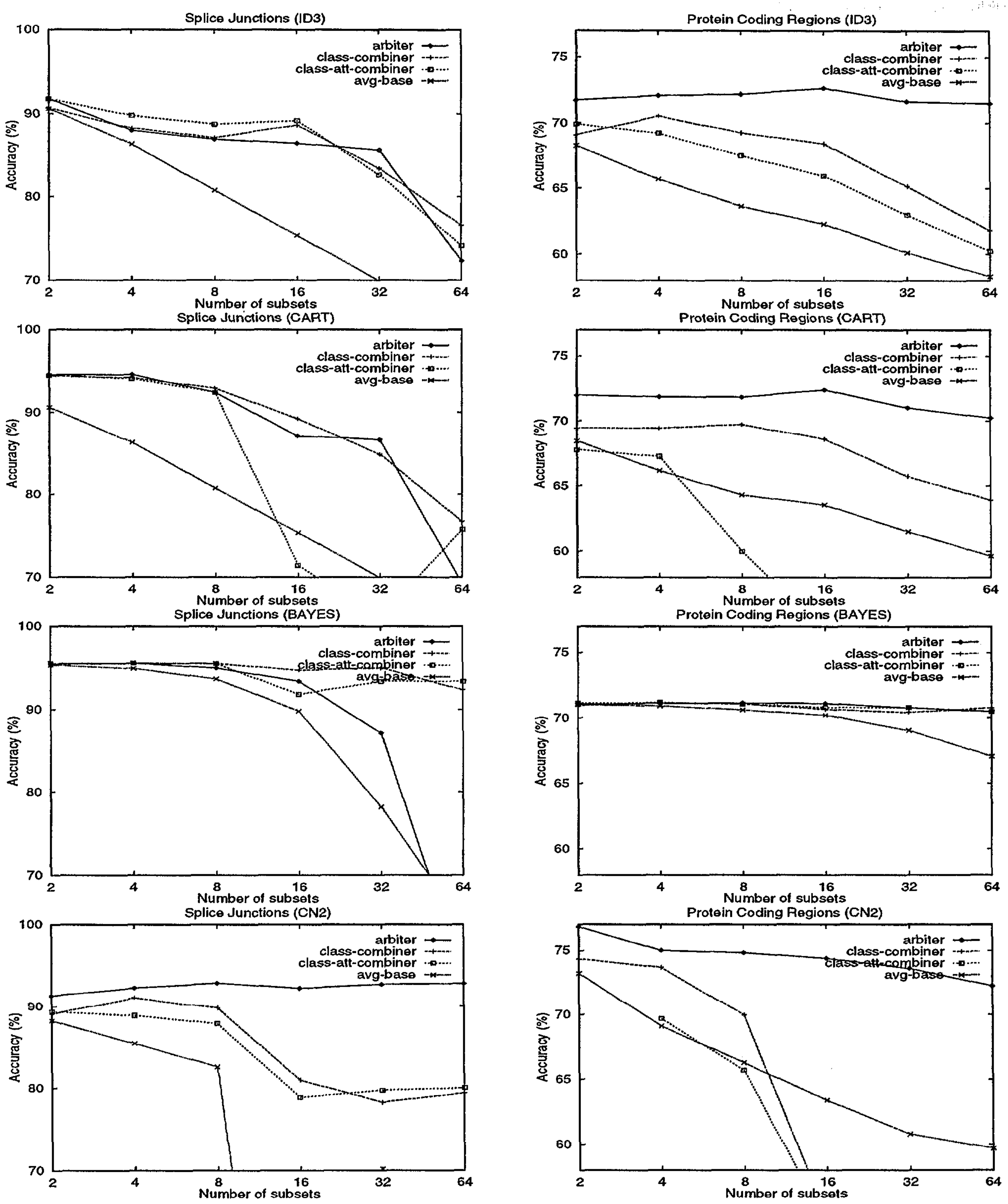
Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3:261–285.

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

Qian, N., and Sejnowski, T. 1988. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.* 202:865–884.

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

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



Splice Junctions (ID3)

Protein Coding Regions (ID3)

Splice Junctions (CART)

Protein Coding Regions (CART)

Splice Junctions (BAYES)

Protein Coding Regions (BAYES)

Splice Junctions (CN2)

Protein Coding Regions (CN2)

Figure 2: Accuracy for local meta-learning vs number of subsets

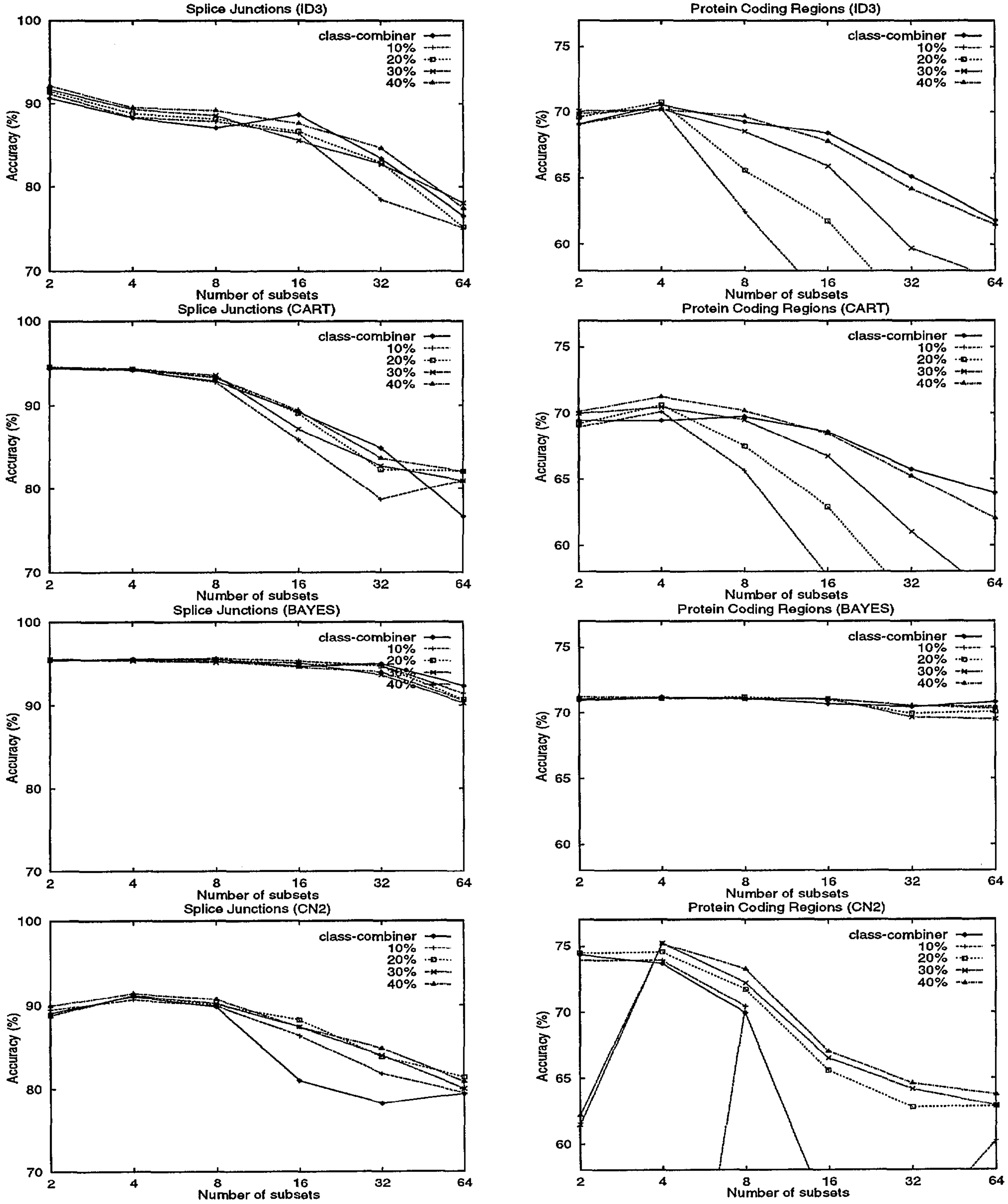


Figure 3: Accuracy for the class-combiner technique trained over varying amounts of replicated data. Δ ranges from 0% to 40%.