Bagging Is A Small-Data-Set Phenomenon

Nitesh Chawla [1], Thomas E. Moore, Jr.[1], Kevin W. Bowyer[2],
Lawrence O. Hall[1], Clayton Springer[3], and Philip Kegelmeyer[3]

[1]Department of Computer Science and Engineering
University of South Florida, Tampa, Florida 33620 USA

[2]Department of Computer Science and Engineering
University of Notre Dame, Notre Dame, Indiana 46556 USA

[3]Sandia National Laboratories, Biosystems Research Department
P.O. Box 969, MS 9951, Livermore, CA, 94551-0969

chawla, tmoore4 @csee.usf.edu, kwb@cse.nd.edu, hall@csee.usf.edu, csprin, wpk@ca.sandia.gov

Abstract

Bagging forms a committee of classifiers by bootstrap aggregation of training sets from a pool of training data. A simple alternative to bagging is to partition the data into disjoint subsets. Experiments on various datasets show that, given the same size partitions and bags, disjoint partitions result in better performance than bootstrap aggregates (bags). Many applications (e.g., protein structure prediction) involve use of datasets that are too large to handle in the memory of the typical computer. Our results indicate that, in such applications, the simple approach of creating a committee of classifiers from disjoint partitions is to be preferred over the more complex approach of bagging.

1. Introduction

Many data mining applications use data sets that are too large to be handled in the memory of the typical computer. One possible approach is to sub-sample the data in some manner [1, 2]. However, it can be difficult a priori to know how to sub-sample so that accuracy is not affected. Another possible approach is to partition the original data into smaller subsets, and form a committee of classifiers [3, 4]. One advantage of this approach is that the partition size can simply be set at whatever amount of the original data can be conveniently handled on the available system. Another advantage is that the committee potentially has better accuracy than a single classifier constructed on all the data.

In its typical form, bagging involves random sampling with replacement from the original pool of training data to create "bags" of data for a committee of thirty to one hundred classifiers. Bagging has been shown to result in improved performance over a single classifier created on all of the original data [5, 6, 7]. The success of bagging suggests that it might be a useful approach to creating a committee of classifiers for large data sets. We define *large* data sets as those which do not fit in the memory of a typical scientific computer. However, experience with bagging has primarily been in the context of "small" data sets. If the original data set is too large to handle conveniently, then creating and processing thirty or more bags will of course present even greater problems. This raises the question of which particulars of the bagging approach are essential in the context of large data sets. In this work, we show that simple partitioning of a large original data set into disjoint subsets results in better performance than creating bags of the same size.

2. Literature Review

Breiman's bagging [5] has been shown to improve classifier accuracy. Bagging basically combines models learned on different samplings of a given dataset. According to Breiman, bagging exploits the instability in the classifiers, since perturbing the training set produces different classifiers using the same learning algorithm. Quinlan experimented with bagging on various datasets and found that bagging substantially improved accuracy [6]. However, the experiments were performed on "small" datasets, the largest one being 20,000 examples.

Domingos empirically tested two alternative theories supporting bagging: (1) bagging works because it approximates Bayesian model averaging or (2) it works because it shifts the priors to a more appropriate region in the decision space [8]. The empirical results showed that bagging worked possibly because it counter-acts the inherent simplicity bias of the decision trees. That is, with M different bags, M different classifiers are learned, and together their

0-7695-1272-0/01 $10.00 © 2001 IEEE

output is more complex than that of the single learner.

Chan and Stolfo [9] compared arbiter and combiner strategies by applying a learning algorithm to disjoint subsets of data. Their experiments showed that the arbiter strategy better sustains the accuracy compared to the classifier learned on the entire data set. The combiner strategy showed a drop in accuracy with the increase in the number of subsets, which can be attributed to the lack of information content in the small subsets. However, a few cases resulted in an improvement in accuracy. We are interested in disjoint subsets of larger original data sets than in [9], and so there is reason to expect that accuracy can be maintained.

Chan and Stolfo relaxed their definition of strict disjoint subsets [10] by allowing a small amount of overlap across the subsets. On the datasets DNA Splice Junction with 3,190 examples and Protein Coding Region with 20,000 examples, it was found that overlapping did not bring any gain to their meta-learning strategy. Each classifier trained on a disjoint set is biased towards its own set, and when these classifiers are combined a protocol of knowledge sharing is established, and each individual classifier's bias is reduced. Again, we are interested in "large" data sets relative to those considered in this work.

Hall et al [11] learned decision trees using disjoint partitions of data and then combined the classifiers. It was found that using a conflict resolution strategy for combining rules, the accuracy usually did not decrease for a small number of partitions, at least on the datasets tested. Our current work is similar to this, but focuses on comparison of bagging-like approaches to simple partitioning of large datasets.

Provost et al [1] found that sub-sampling the data gave the same accuracy as the entire dataset at much lower computational cost. They analyzed "progressive sampling" methods– progressively increasing the sample size until the model accuracy was maintained. It was found that adding more training instances did not help the accuracy of the classifier, and after some number of instances the performance of the classifier plateaus. As pointed out in the Discussion, our results indicate that simple sub-sampling to produce one smaller training set is not a profitable strategy. However, more complicated sub-sampling strategies may be useful.

3. Experiments

Three sets of experiments were performed. The first uses four "small" datasets, representative of those commonly used in pattern recognition and machine learning research. It compares four approaches to creating a committee of N classifiers, with each classifier created using (1/N)-th of the training data. The performance of the approaches is also compared to that of "true bagging" – bags of the same size as the pool of training data, randomly sampled with replacement. The point of this first set of experiments is to isolate the essential factor(s) leading to good performance in the committee of classifiers. The second set of experiments uses a "moderate" size dataset of almost 300,000 examples. The same four approaches are evaluated on this data set. The point is to verify that the pattern of performance results observed with smaller data sets holds with a larger data set.

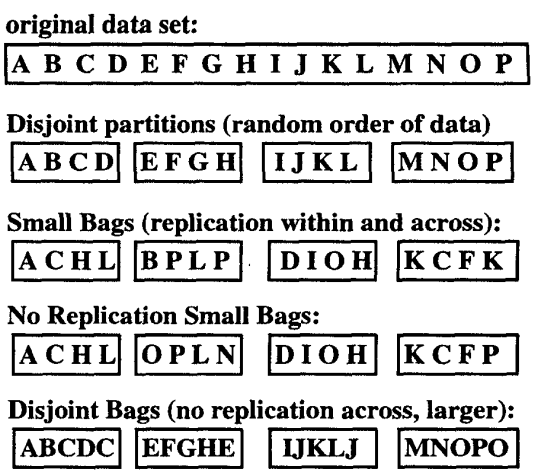

Figure 1: Four Approaches to a Committee of Classifiers.

Based on the first two sets of experiments, the disjoint partitioning approach is identified as offering the best performance for a given size of partitions/bags. It is also the simplest of the approaches considered. The last experiment uses a "large" dataset of approximately 3.6 million examples to investigate the degree of performance improvement that the disjoint partitioning approach can achieve over a classifier built on all the original data.

3.1. Variations of Partitioning & Bagging

We investigated four different approaches to creating a committee of classifiers from an original data set. See Figure 1 for an illustration. One approach is to simply partition the original data into N disjoint partitions of size (1/N)-th of the original data. Thus the union of the N training sets is identical to the original data. Results of this approach are labeled with "D" (for "disjoint") on the graphs.

The second approach is to create N bags of size (1/N)-th of the data. Each bag is created independently by random sampling with replacement, so the union of the training sets is generally not the same as the original data. This approach is labeled "SB" (for "small bags") on the graphs. Comparison of the SB performance versus that of disjoint partitions shows whether the random replication of data elements results in any inherent advantage.

The third approach is like small bags, but sampling without replacement for each individual bag. Sampling the individual bags without replacement, elements of the original

data do not repeat within a bag, but may repeat across bags. This approach is labeled "NRSB" (for "no-replication small bags") on the graphs.

The fourth approach begins with the disjoint partitions. Then, independently for each partition, a number of its elements are randomly selected with replacement to be added to the "bagged disjoint." Thus the union of the training sets is a superset of the original data; all elements of the original data appear, plus some random replications. The number of added elements is equal to the average number of repeated elements in a bag in the "small bags." Thus a bag used in this approach is slightly larger than (1/N)-th of the original data. The amount of "extra" data included decreases as the bag size decreases. Results of this approach are labeled "DB" (for "disjoint bagged") on the graphs. Comparison of the results of this approach to the results of disjoint partitions looks again at whether the random replication of data elements results in any inherent advantage.

In addition to the above four approaches, we also ran "true bagging" on each of the four "small" datasets. By "true bagging" we mean creating M bags, each of the size of the original data, independently using random sampling with replacement. True bagging is expected to out-perform committees of classifiers formed on smaller data sets, but the point is to provide a baseline performance comparison for the other approaches.

3.2. Datasets

Three of the small data sets are from the UCI repository, and one is from our own research. The "moderate" size dataset comes from the problem of predicting the secondary structure of proteins. It is the training data set used with a neural network that won the CASP-3 secondary structure prediction contest [12]. This dataset contains almost 300,000 elements. Each amino acid in a protein can have its structure labeled as helix (H), coil (C), or sheet (E). The features for a given amino acid are twenty values in the range -17 to 17, representing the likelihood of the amino acid being any one of twenty basic amino acids. Using a window of size seventeen centered around the target amino acid, gives a feature vector of size 340. (Jones [12] actually used a window size of 15 and scaled the feature values into the range [0,1].) The size and class distribution of these datasets is summarized in Table 1.

Note that the experiments include both two-class (Mammography) and multi-class (Letter, PenDigits, SatImage) datasets. They also include datasets that are approximately balanced (Letter, PenDigits) and those that are skewed (Mammography, SatImage).

The four approaches to creating a committee of classifiers, plus true bagging, were applied to each of the "small" datasets. The number of bags/partitions was varied from one to eight. Given the modest size of the datasets, creating bags/partitions of less than (1/8)-th the original size is likely to starve the classifiers for training data. For the experiments on the small and moderate size datasets, the reported results are calculated from 10-fold cross-validation.

Letter dataset (UCI) - 20,000 samples in 26 classes					
A:789	B:766	C:736	D:805	E:768	F:775
G:773	H:734	I:755	J:747	K:739	L:761
M:792	N:783	O:753	P:803	Q:783	R:758
S:748	T:796	U:813	V:764	W:752	X:787
Y:786	Z:734				
Pendigits dataset (UCI) - 10,992 samples in ten classes					
0:1,143	1:1,143	2:1,141	3:1,055	4:1,144	5:1,055
6:1,056	7:1,142	8:1,055	9:1,055		
Satimage dataset (UCI) - 6,435 samples in six classes					
1:1533	2:703	3:1,358	4:626	5:707	7:1,508
Mammography dataset - 11,183 samples in two classes					
1:10,923	2:260				
Jones' PDB dataset - 299,186 samples in three classes					
H:104,572		C:128,881		E: 65,733	
PDB dataset - 3,619,461 samples in three classes					
H:1,254,335		C:1,537,261		E:827,865	

Table 1: Data Sets Sizes and Class Distributions.

Our "large" dataset also comes from the Protein DataBase (PDB) used in the CASP contests [13]. For 18,098 protein chains taken from the PDB, there are a total of 3,679,152 amino acids for structure prediction. This training data takes from 1.3 to 30 GB to store, depending on how feature values are encoded (e.g. signed char, integer, or float). The test data for the experiments with the large dataset consists of a separate set of data. It is all protein chains entered into the PDB from July 11 2000 to July 28 2000, that are based on X-ray crystallography of three angstroms or finer. There were 146 chains entered in this time frame, made up of 38,423 amino acids. All results are reported on a per chain basis rather than per amino acid, because that is how results are reported in the CASP contest. Performance on individual amino acids is very similar to the average per chain.

3.3. Base Classifier and Computing Systems

For the experiments on the small and moderate size datasets, release 8 of the C4.5 decision tree system [14] was run on standard SUN workstations. The one run of the large dataset to produce a single classifier was done on a 64-processor SGI IRIX64 with 32 GB of main memory at Sandia National Labs, also using the standard C4.5 release 8. Creating the one decision tree on the large dataset took approximately **thirty days** on the SGI.

The experiments using partitions of the large dataset were run on the DOE's "ASCI Red" parallel supercomputer

[15]. The ASCI Red has 4,640 compute nodes, each containing two Pentium III processors sharing 256 MB of memory. The processors run a version of the UNIX operating system. The system is based on a distributed-memory mesh architecture, and is capable of 3.15 TeraFLOPS. These experiments used a version of C4.5 modified with MPI calls for parallel execution. The parallel structure of this version of C4.5 is quite simple. The disjoint partitions are loaded into the different compute node memories and each compute node independently grows a decision tree. The parallel computation to create eight decision trees on one-eighths of the large dataset takes approximately ten hours; that is, eight processors running in parallel for ten hours.

4. Results

Figures 2 through 5 summarize the experimental comparison of the different approaches on the small datasets detailed in Table 1. The plots compare the performance of two, four, six, and eight disjoint partitions (D) to that of C4.5 on the complete data set, and to classifier committees formed using the other three approaches (DB, SB, NRSB). Results are shown as the paired average difference across the ten folds in the ten-fold cross-validation, with standard error indicated.

As an example, the first cluster of four data points on the plot in Figure 2 represents the results for a committee of two classifiers on the Letter data set. The first point is the difference between a committee of two disjoint partitions and C4.5 trained on all of the data; note that the committee of two classifiers performs significantly worse. The second point is the difference between a committee formed using two disjoint partitions versus a committee using two disjoint bags (DB), the third point is two disjoint partitions versus two small bags (SB), and the fourth point is two disjoint partitions versus no-replication small bags (NRSB).

From examining the sequence of plots it is clear that disjoint partitions generally, but not always, beat small bags. It appears to make little difference whether the small bags are created by sampling with or without replacement. The "bagged disjoints" appear to generally perform slightly better than the simple disjoints, but then the training sets for the individual decision trees are slightly larger.

Because it uses constant-size bags as the number of classifiers in the committee grows larger, "true bagging" should naturally outperform any of the four approaches. Data points for "true bagging" performance are given in Table 2. However, the point is that true bagging is simply not a practical option for "large" datasets. For the example large dataset here, true bagging would require creating about fifty classifiers, each training on a data set of the same size as the original data. Recall that creating one classifier on all the data took 30 days on a substantial SGI system. When the

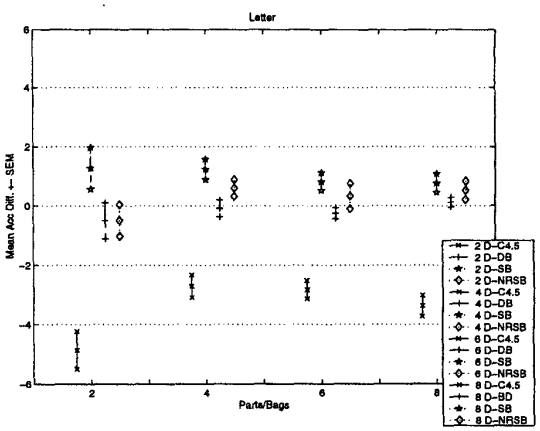

Figure 2: Comparison on Letter Dataset.

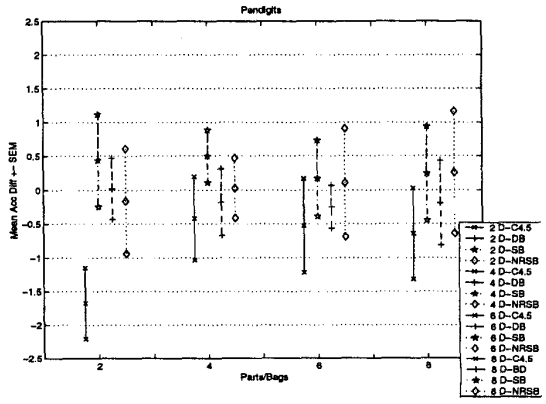

Figure 3: Comparison on PenDigits Dataset.

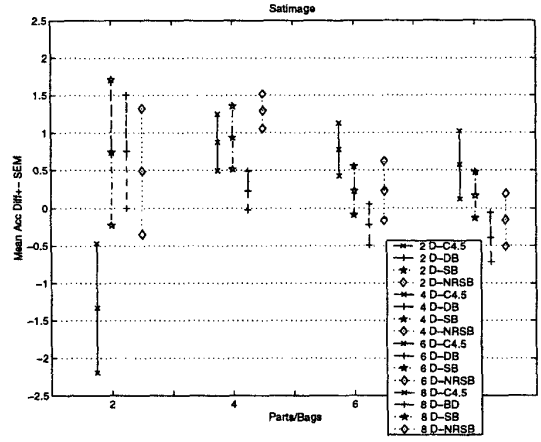

Figure 4: Comparison on SatImage Dataset.

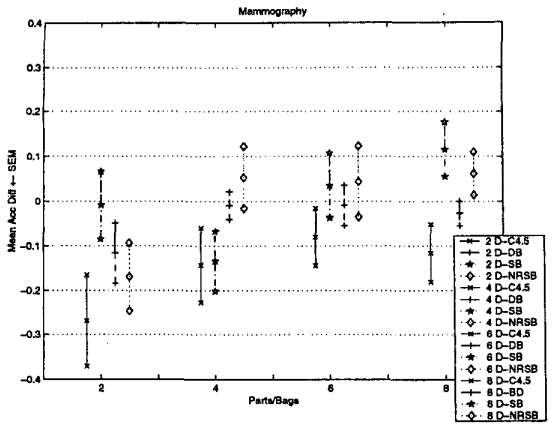

Figure 5: Comparison on Mammography Dataset.

Dataset	C4.5	50 bags	75 bags	100 bags
Satimage	86.30	90.89	90.86	90.84
Pendigits	96.57	98.42	98.43	98.36
Mammography	98.50	98.76	98.79	98.79
Letter	88.10	93.54	93.65	93.80

Table 2: Data Points for "True Bagging" Results.

dataset is too large to handle conveniently in the memory of the typical computer, the dataset must be broken into some number of practical size, though not "small," chunks. The question addressed here is whether there is any advantage in creating the practical size chunks using some bagging-like approach, or whether simple partitioning is sufficient. data.

The comparison of the four approaches on the "moderate" size dataset are shown in Figure 6. The results are plotted here as a simple trend, rather than means of paired differences. (The differences and the number of data items here are larger, and so the statistical significance of results is clearer.) Again, we see that simple disjoint partitioning offers excellent performance in comparison to the other options. In particular, the "small bags" approach performs poorly. Only the "bagged disjoints," with its slightly larger number of elements in each bag, offers any hint of performance improvement over disjoint partitions.

Figure 7 compares the results of creating one decision tree on all of the large dataset versus using a committee of N classifiers, for N = 8, 16, 24, 32, and 40. All of the committees were formed using disjoint partitions of size (1/8)-th of the large PDB dataset. This size partition just fills the memory of the compute nodes on the ASCI Red. For the committees of 16, 24, 32, and 40 classifiers, multiple different partitions of the dataset were used. For example, to create a set of sixteen classifiers, eight classifiers trained on a different eight-partition of the data were added to those

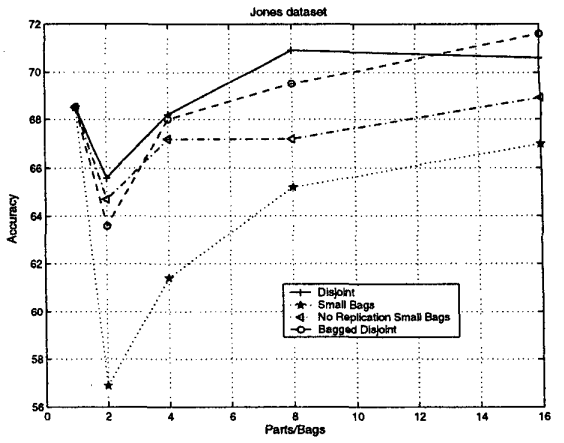

Figure 6: Results on (Jones') "Medium" PDB dataset.

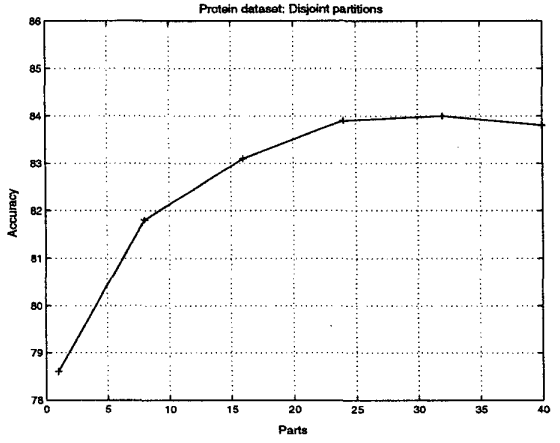

Figure 7: Partitioning Results for "Large" PDB dataset.

created on the original eight-partition.

The average accuracy of a single classifier trained on (1/8)-th of the large dataset is 74.1%. A single decision tree created using all the data performs substantially better than this, 78.6% versus 74.1%. At the same time, a committee of eight classifiers created on (1/8)-ths of the data performs substantially better than a single tree created on all the data, 81.8% versus 78.6%.

5. Conclusions and Discussion

The results support several important conclusions. The overall conclusion is that datasets too large to handle practically in the memory of the typical computer are appropriately handled by simple partitioning to form a committee of classifiers. More specifically, a committee created using disjoint partitions can be expected to outperform a commit-

tee created using the same number and size of bootstrap aggregates ("bags"). Also, the performance of the committee of classifiers can be expected to exceed that of a single classifier built from all the data.

The following considerations may provide insight into the pattern of results. Practical factors aside, one generally wants (a) each classifier in a committee to be formed using as much data as possible, and (b) the size of the committee to be as large as possible. Practical considerations typically (a) limit the amount of data that can be used in training a single classifier, and (b) limit the size of a classifier committee. If the data set is large enough, or the memory limit small enough, then partitioning into N disjoint subsets gives a reasonable size committee and this approach should suffice. If the N disjoint partitions result in too small of a committee, then the data set may be partitioned multiple times to increase committee size. As a rule of thumb, the committee size should be thirty or more. Random replication of data elements within the training set for a single classifier appears to be of value only when a (1/N)-th subset of the original data would result in an incompetent classifier.

Results obtained here seem to support the position that that bagging results depend simply on obtaining a "diverse" set of classifiers [5, 16]. Building classifiers on disjoint partitions of the data provides a set of classifiers that meet this requirement. Each individual classifier performs similarly, but correctly classifies a (partially) different set of examples.

Some researchers have suggested that many large-dataset problems can be solved using only a fraction of the data, perhaps by simple sub-sampling. Classical pattern recognition would suggest that this question is more appropriately viewed in terms of the density of training sample population in the feature space, rather than simply the size of the dataset. There is "excess" data only when (parts of) feature space are densely populated. The fact that the average (1/8)-th partition of our large dataset had performance of 74.1%, whereas a single classifier trained on all the data gave 78.6%, indicates that the original data could not be profitably sub-sampled in a simple way. Given that the problem has a 340-dimension feature space, this is perhaps not surprising, as even 3.6 million examples can result in a "sparse" population of such a space.

Acknowledgments

This work was supported in part by the United States Department of Energy through the Sandia National Laboratories LDRD program and ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789.

References

[1] F.J. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, 1999.

[2] L. Breiman. Pasting bites together for prediction in large data sets. *Machine Learning*, 36(1,2):85–103, 1999.

[3] P. Chan and S. Stolfo. Towards parallel and distributed learning by meta-learning. In *Working Notes AAAI Workshop on Knowledge Discovery in Databases*, pages 227–240, 1993.

[4] F.J Provost and D.N Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI '96*, pages 74–79, 1996.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[7] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1,2), 1999.

[8] P. Domingos. Why does bagging work? A Bayesian account and its implications. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[9] P. Chan and S. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, pages 39–44, 1995.

[10] P. Chan and S. Stolfo. Scaling learning by meta-learning over disjoint and partially replicated data. In *9th Florida Artificial Intelligence Research Symposium*, pages 151–155, 1996.

[11] L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, T.E. Moore, and C. Chao. Distributed learning on very large data sets. In *ACM SIGKDD Workshop on Distributed and Parallel Knowledge Discovery*, 2000.

[12] D.T. Jones. Protein secondary structure prediction based on decision-specific scoring matrices. *Journal of Molecular biology*, 292:195–202, 1999.

[13] Lawrence Livermore National Laboratories. Protein structure prediction center. *http://predictioncenter.llnl.gov/.*

[14] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

[15] Sandia National Labs. ASCI RED, the world's first teraops supercomputer. *http://www.sandia.gov/ASCI/Red/.*

[16] T. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.