Bagging-Like Effects for Decision Trees and Neural Nets in Protein Secondary Structure Prediction

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

[1]Department of Computer Science and Engineering, University of South Florida,
4202 East Fowler Avenue, Tampa, FL 33620 USA

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

{chawla, tmoore4, kwb, hall} @csee.usf.edu, {csprin, wpk} @ca.sandia.gov

ABSTRACT

In the Third Critical Assessment of Techniques for Protein Structure Prediction ("CASP-3") contest, the best performance was obtained with a classifier that uses neural networks, a window size of fifteen around a given amino acid, and a training set of about 299,186 amino acids. We set out to investigate the possibility of obtaining better performance by using a bagging-like committee of binary decision trees, created using an order-of-magnitude more training data. There are two main reasons to believe that it should be possible to obtain better performance in this way. One is that Jones did not use a committee of classifiers in CASP-3 (and used only a four-classifier committee in CASP-4), whereas bagging studies indicate that improvement plateaus in the range of thirty to fifty classifiers in a committee. A second is that, by using supercomputers available at the Sandia National Labs, it is feasible to use an order of magnitude more training data than was used by Jones. This paper reports on our experiences pursuing this line of research. We show that with "large" data sets and with bag size equal to partition size, simple disjoint partitioning performs at least as well as standard bagging. Given large datasets, either outperforms a single classifier built on all the data. We also show that there are subtle differences in the operation of binary decision trees and neural networks for this problem. One difference is that the neural network seems less prone to "over-learning" the "easy" subset of the training data.

Keywords

protein structure prediction, binary decision trees, neural networks, bootstrap aggregation, combination of classifiers

1. INTRODUCTION

There are many data mining applications in which the data sets are too large to fit into the memory of the typical computer [25; 9; 6; 22; 19; 3; 13; 15; 20]. One possible approach is to sub-sample the data in some manner [22; 5]. However, it can be difficult a priori to know how to subsample the data so that the accuracy of the classifier is not affected. Another possible approach is to partition the original data into smaller subsets, and use each subset to create one of a committee of classifiers [9; 6; 21; 3; 13; 15]. One advantage of this approach is that the partition size can simply be set at whatever amount of the original data can be maximally handled on the available system. Another advantage of this approach is that the committee of classifiers potentially has better accuracy than a single classifier constructed on all the data.

Bagging ("bootstrap aggregation") is a well-known technique for creating a committee of classifiers. In its typical form, it involves random sampling with replacement from the original data set to create "bags" of data of size 100% of the original data. Typically, a committee of 30 to 100 classifiers might be created in this way. Bagging has been shown to result in improved performance over a single classifier created on all of the original data [4; 24; 1].

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 data set is too large to be handled in the memory of a typical computer, then so is a bag. Similarly, creating and processing fifty or more bags of the size of the original data will likely present serious difficulties. This raises the question of which particulars of the bagging approach are essential in the context of large data sets. We show that partitioning a large original data set into N disjoint subsets is all that is needed to achieve a bagging-like effect.

We also explore the relative suitability of binary decision trees versus neural networks in the context of protein secondary structure prediction. Secondary structure prediction is an important application of "extreme" pattern recognition. It is "extreme" in the size of the learning data, which might be millions of examples requiring gigabytes of storage. It is also "extreme" in the dimensionality of the feature space, which is on the order of 300-dimensional in recent successful classifiers for this application. The basic problem is to predict one of three secondary structure labels at a given amino acid in a given protein. The feature vector might represent a window of 15 amino acids centered around the one whose structure is predicted, and contain a scaled likelihood value for each of twenty possible amino acid substitutions at each of the 15 amino acids ($15 \times 20 = 300$-dimensional).

2. LITERATURE REVIEW

Breiman's bagging [4] has been shown to improve classifier accuracy. Bagging basically combines models learned on different samplings of a given dataset. Bagging turns an order-correct learner into a nearly-optimal one. 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 [24]. However, the experiments were performed on small datasets, the largest one being 20,000 examples.

Domingos empirically tested two alternative theories supporting bagging on decision trees: (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 [10]. 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 output is more complex than what a single learned classifier produces.

The family of boosting algorithms [12] uses the entire dataset for learning. It assigns weights to the training instances, and these weight values are changed depending upon how well the associated training instance is learned by the classifier; the weights for misclassified instances are increased. Thus, re-sampling occurs based on how well the training samples are classified by the previous model. Since the training set for one model depends on the previous model, boosting requires sequential runs and thus is not readily adapted to a parallel environment. It is also going to be quite time-consuming for large disjoint subsets of data.

Combiner and arbiter strategies for combination of classifiers were evaluated in [7]. Their experiments showed that the arbiter strategy sustains the accuracy compared to the classifier learned on the entire data set. The combiner strategy experienced a drop in accuracy with the increase in the number of subsets, which can be attributed to the lack of enough information content from the small subsets. However, a few cases resulted in an improvement in the accuracy. In our work reported here, we are interested in disjoint subsets of "large" original data sets, larger than in [7], and so there is reason to expect that accuracy can be maintained.

Chan and Stolfo relaxed their definition of strict disjoint subsets [8] 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 [13; 15; 14] used disjoint partitions of data to learn on, and then combined the classifiers. The classifiers were learned using C4.5 release 8 [23]. 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 that were 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 [22] found that sub-sampling the data gave the same accuracy as learning from 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 (n_{min}) the performance of the classifier plateaus. Initial indications from our work reported here are that the large data set used **cannot** profitably be sub-sampled.

3. EXPERIMENTS ON SMALL DATASETS

One set of experiments uses five "small" public datasets to compare several different approaches for creating a committee of classifiers. The point of this set of experiments is to isolate the essential factors leading to good performance. A second set of experiments uses a true large dataset for the prediction of secondary structure of proteins, as "large" is defined in this paper, to test the basic conclusion that simple disjoint partitioning is an effective method to create a committee of classifiers for such datasets.

3.1 Variations of Partitioning & Bagging

We investigated four different approaches to creating a committee of N classifiers from an original data set. 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 original data. Each bag is created independently, by random sampling with replacement from the original data. The union of the training sets is generally not the same as the original data. Elements of the original data may be replicated within a bag or across bags, and not all elements of the original training data necessarily appear. This approach is labeled "SB" (for "small bags") on the graphs. Comparison of the performance of "small bags" versus that of disjoint partitions shows whether the random replication of elements of the original training set results in any inherent advantage.

The bagging approach blends (at least) two effects. The random sampling with replacement means that different bags may contain different elements of the original data. This may aid "diversity" of classifiers and thereby help to increase performance. The sampling with replacement also effectively works to increase the number of individual data elements that can be in a given bag, which changes their "weight." We examined two additional approaches, other than simple disjoint partitioning and small bags, to explore which elements of the bagging approach are most important. The third approach is like small bags, but sampling without replacement for each individual bag. Some authors have reported that the effect of bagging does not depend on whether the bags are sampled with or without replacement. 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 in this approach 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 approaches, we also ran "true bagging" (M bags the size of the original data, each created independently using random sampling with replacement) on each of the five "small" datasets. The point of this data is to provide a baseline performance comparison with the other approaches.

3.2 Datasets

Five small size data sets were used in experiments. Three of these are from the UCI repository [2], one is from the ELENA project[1], and one is from our own research. The size and class distribution of these datasets is summarized in Table 1. Note that the datasets include both two-class (Mammography, Phoneme) and multi-class (Letter, PenDigits, SatImage), and those that are approximately balanced (Letter and PenDigits) and those that are skewed (Mammography, Phoneme, SatImage).

The number of bags/partitions was varied from one to eight for the four approaches to partitioning the dataset. 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.

3.3 Base Classifiers and Computing Systems

The base classifier in our experiments is release 8 of the C4.5 decision tree system [23]. For the experiments on the modest size datasets, C4.5 was run on standard SUN workstations.

3.4 Results of Comparison of Approaches

Figures 1 through 5 summarize the experimental comparison of the different approaches on the small datasets detailed in Table 1. The plots compare the percentage correct of the disjoint partitions approach to each of the other three approaches, and to the performance of a single C4.5 decision tree. Results are shown for a classifier committee of size two, four, six, and eight. Results are shown as the average paired difference in accuracy across the ten folds in the ten-fold cross-validation, with standard error of the mean (SEM) indicated. Wherever the mean difference is greater than zero and the range of the standard error does not include zero, the disjoint partitions result is statistically significantly better. Wherever the mean difference is less than zero and the range of the standard error does not include zero, as is generally the case in comparing a committee of disjoint partitions to a single C4.5 decision tree, the performance of the disjoint partitions is statistically significantly worse.

[1]ftp.dice.ucl.ac.be in the directory pub/neural-nets/ELENA/databases.

From examining these plots it is clear that disjoint partitions generally, but not always, outperformed 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.

Several conclusions are evident from this first set of experiments. One important conclusion is that creating a committee of classifiers using simple disjoint partitions generally outperforms a committee created using the same number and size of bags. Disjoint partitions outperform small bags at all comparison points for three of the data sets, and at most of the points across the other two data sets. 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.

True bagging naturally out-performs any of the four approaches considered, as it uses constant-size bags as the number of classifiers in the committee grows larger. However, the point is that true bagging is simply not a practical option for "large" datasets. When the dataset is too large to handle, the dataset must be broken into some number of practical-sized, 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. These experiments show that disjoint partitions perform at least as well bags of the same size. However, with these small datasets, the committee of classifiers created on disjoint subsets of the data generally does not outperform a single classifier created using all of the training data. (The satellite image dataset shows a relatively minor improvement.)

4. MODERATE-SIZE PROTEIN DATA SET

The same four approaches investigated on the small datasets were also run on the Jones data set from CASP-3 [16]. We will refer to this as a "moderate" 299,186-item data set, standing between the "small" data sets and the "large" 3.6-million-item data set. The Jones dataset comes from the problem of predicting the secondary structure of proteins. Each amino acid in a protein can be labelled 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, we have a feature vector of size 340. This window size and amino acid encoding is similar to that used by Jones [16]; Jones actually used a window size of 15 and scaled the feature values into the range [0,1].

This experiment used a separate set of test data. The test data contains 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, containing a total of 38,423 amino acids. All results are reported on a per chain basis rather than per amino acid, as this is how results are reported in the CASP contest. The per-amino-acid performance figures and the per-chain performance figures are generally similar. These runs were performed using the

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	K: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				
Phoneme dataset (ELENA) - 5,404 samples in two classes									
0:3,818	1:1,586								
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 Used in Experiments, with Size and Class Distribution.

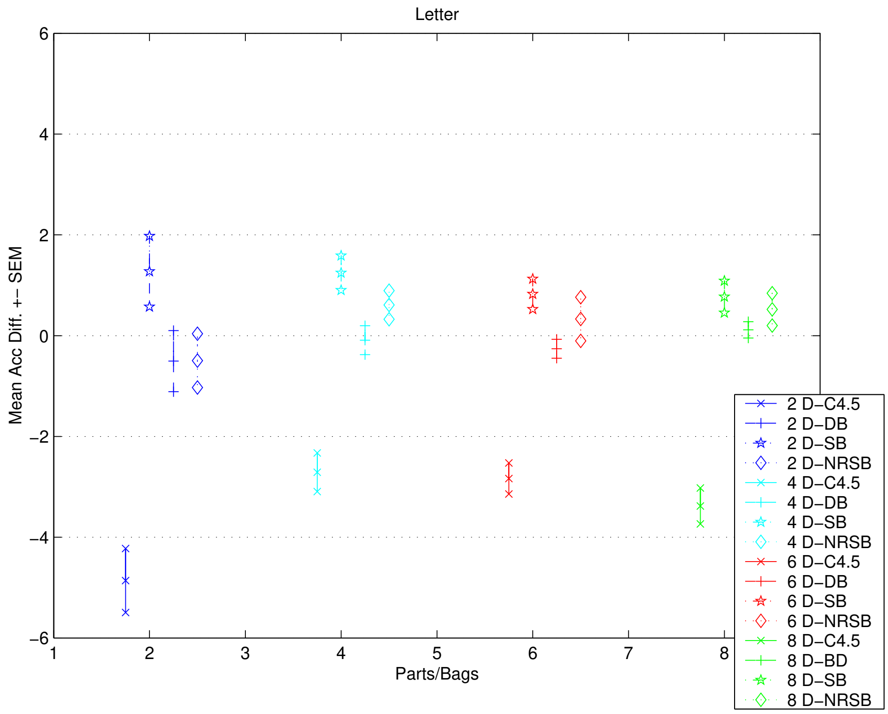

Figure 1: Comparison of Partitioning/Bagging Variations on Letter Dataset.

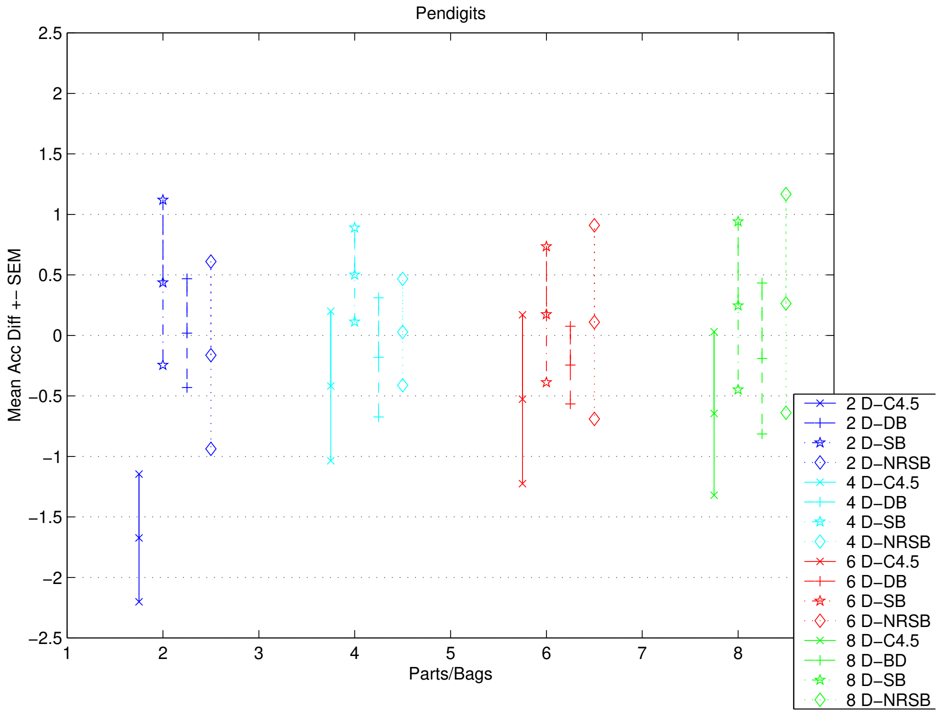

Figure 2: Comparison of Partitioning/Bagging Variations on PenDigits Dataset.

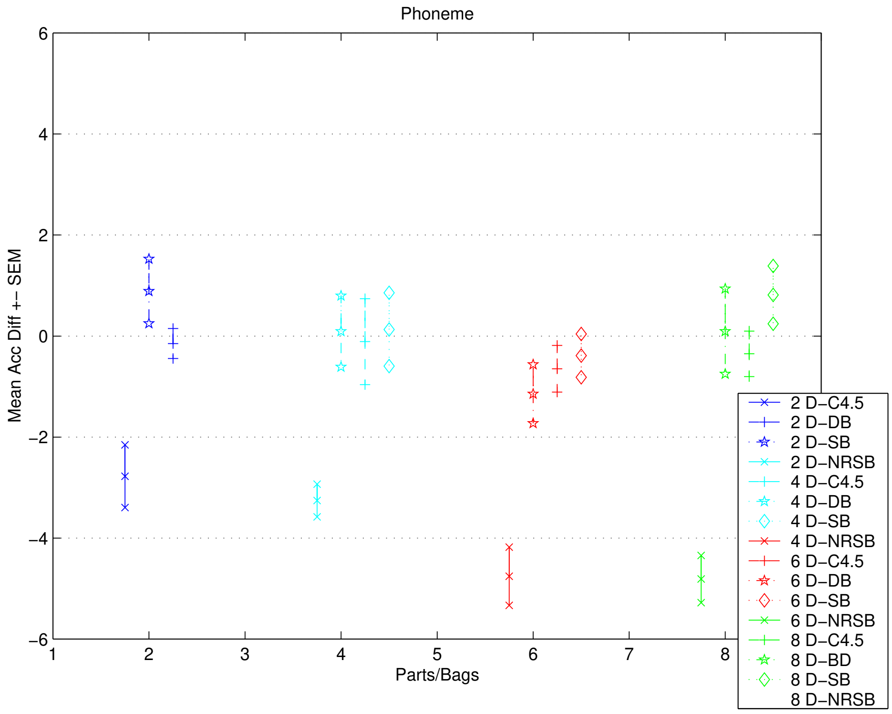

Figure 3: Comparison of Partitioning/Bagging Variations on Phoneme Dataset.

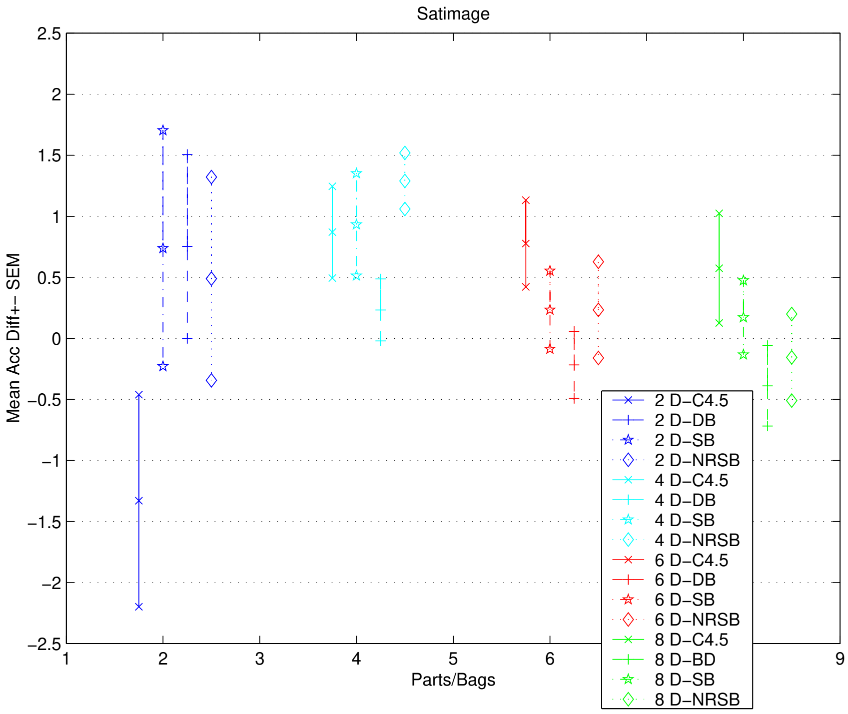

Figure 4: Comparison of Partitioning/Bagging Variations on SatImage Dataset.

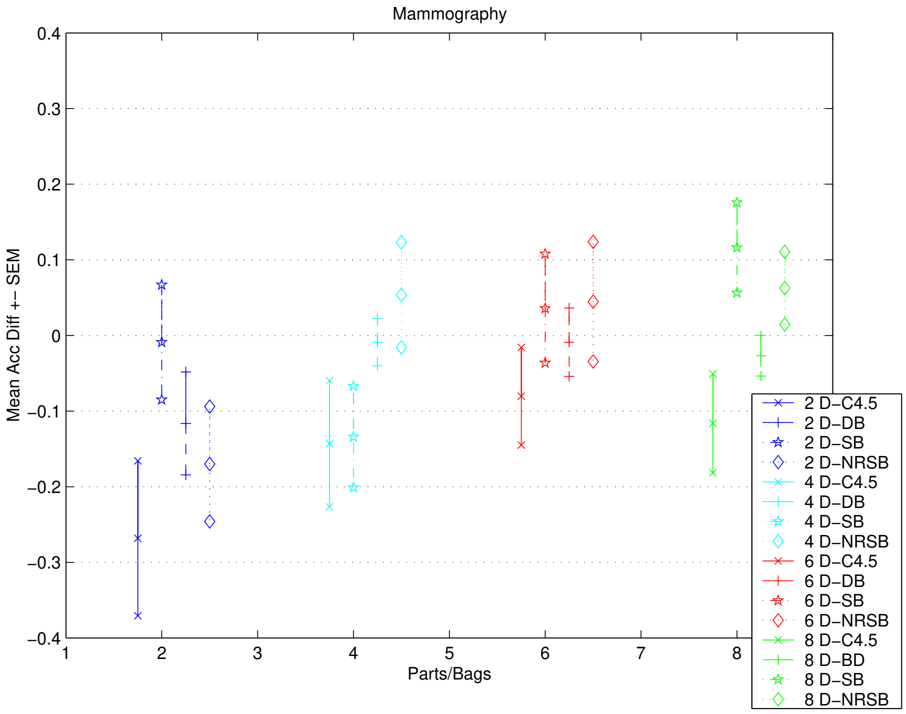

Figure 5: Comparison of Partitioning/Bagging Variations on Mammography Dataset.

standard C4.5 implementation on typical SUN workstations. The results of this experiment are plotted in Figure 6. The performance comparison of the four approaches follows the same general pattern as in the experiment with the small datasets. Disjoint partitions again offer better performance than bags of the same size. However, there is also an important difference between the results here and the results with the small datasets. With the small data sets, the committee of classifiers did not out-perform the single classifier constructed on all of the data. In the results for this "moderate-sized" data set, the committee of classifiers does improve over the single classifier constructed on all of the data.

5. COMMITTEE OF CLASSIFIERS ON A LARGE PROTEIN DATA SET

Our truly large dataset also comes from the Protein DataBase (PDB) used in the CASP contests [17]. For 18,098 protein chains taken from the PDB, we have a total of 3,679,152 amino acids for structure prediction. This training data takes approximately 1.3 GB to 30 GB depending on how the feature values are encoded (e.g., "signed char," integer, or float). The test data is again the separate set of 38,423 amino acids. Results are again reported on a per chain basis.

One run of the complete large dataset to produce a single classifier was performed on an 64-processor SGI IRIX64 IP27 with 32 GB of main memory (!) at Sandia National Labs. This run used the standard C4.5 implementation, and so ran on a single processor, but was able to keep all data structures in main memory. It required approximately 30 days to complete. The performance for the single decision tree created using all the data is 78.6%. This accuracy figure is computed as the average accuracy over 146 chains in the test set, as done for the "Q3" metric in the CASP contests.

Runs to create a committee of N decision trees using disjoint partitions of the large data set were performed on the DOE's "ASCI Red" parallel supercomputer [18]. The ASCI Red has a total of 4,640 "compute nodes," each of which contains two Pentium III processors sharing 256MB 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. The parallel structure of our modified version of C4.5 for this purpose is quite simple. The disjoint partitions are loaded into the different CPUs' memories and each CPU grows a decision tree independent of the other CPUs.

Partitioning the large data set to just fill the memories of the ASCI Red computer nodes requires the use of eight nodes. To create a set of sixteen classifiers, eight classifiers trained on a different eight partition of the data were added to those created on the original eight-partition. The same approach was used to create 24, 32 and 40 classifiers. Figure 7 summarizes results from voting 8, 16, 24, 32, and 40 classifiers created on disjoint partitions of the PDB dataset.

Note that while the performance from creating a single decision tree using all the data is 78.6%, the performance of a voting committee of eight partitions covering all of the data improves to 81.8%. With a committee of thirty-two classifiers, formed from four different eight-partitions, accuracy increases to 84.1%. The committee of classifiers actually has an even stronger effect than is evident from these numbers. The average performance of the eight individual classifiers is only 74.1%. In this experiment, we again see clearly that disjoint partitioning is a reasonable way of creating a committee of classifiers, and that the resulting committee out-performs a single classifier created using all of the data.

6. BDT VERSUS ANN DIFFERENCES

One original goal of this work was to develop a classifier to compete in the CASP 4 secondary structure prediction contest. The motivation was that it should be possible to perform well by using a truly large training data set, applying an appropriate bagging-like approach, and computing on the ASCI Red. We entered predictions for 38 of 41 "targets" in the CASP 4 contest. Our classifier was a committee of 32 decision trees, each grown on a 1/8-th partition of the 3.6 million item training set. When we measured the performance of our classifier using our own 146-chain (38,423 amino acids) validation set, our classifier out-performed Jones' on-line CASP-4 classifier. However, our CASP-4 predictions were generally not competitive with Jones' CASP-4 predictions. In the process of understanding why this is the case, we have discovered several points about performance of decision trees versus neural networks.

To make it computationally practical to run experiments to investigate various effects, we constructed a version of the Jones 299,186-item data set using a window size of five. A single cascade-correlation neural network [11] of size 75 hidden units was created on this training data, and then tested on our 146-chain (38,423 amino acid) validation set. A single C4.5 decision tree was also created on the same training set and tested on the same validation set. The decision tree was evaluated without pruning, with default pruning, and with the strongest possible pruning using the C4.5 "pessimistic pruning" (certainty factor = 1).

In the process of looking at results of the decision tree on individual proteins in the validation set, it was noticed that the decision tree performance was nearly perfect on some proteins and then quite poor on other proteins. Thus the decision tree effectively divided the data into an "easy" subset (chains from the validation set on which the accuracy was over 90%) and a "hard" subset. In comparison, this effect was much smaller with the neural network. While the decision tree accuracy with maximum pruning and use of the feature to indicate zero-padding was only 1.0% lower than the neural network accuracy overall, it was 7.0% lower on the "hard" subset and 5.5% higher on the "easy" subset. This effect hurt the decision tree entries in CASP-4, as the "hard" subset of our validation set was much more representative of the CASP-4 targets than the "easy" subset. Thus while the decision tree performed well in comparison to Jones' on-line classifier on our validation set, it's corresponding performance on the CASP-4 targets would be substantially worse.

Interestingly, the hard/easy gap in performance is strongly affected by the degree of pruning, even though the overall accuracy is not as greatly affected. There are different pruning strategies available for use with decision trees, and a "reduced-error" strategy might allow stronger pruning. We are currently implementing reduced-error pruning for use with C4.5 decision trees, in order to more completely explore the effects of pruning.

Another interesting effect is seen when the feature vector for an amino acid is augmented with an extra feature to

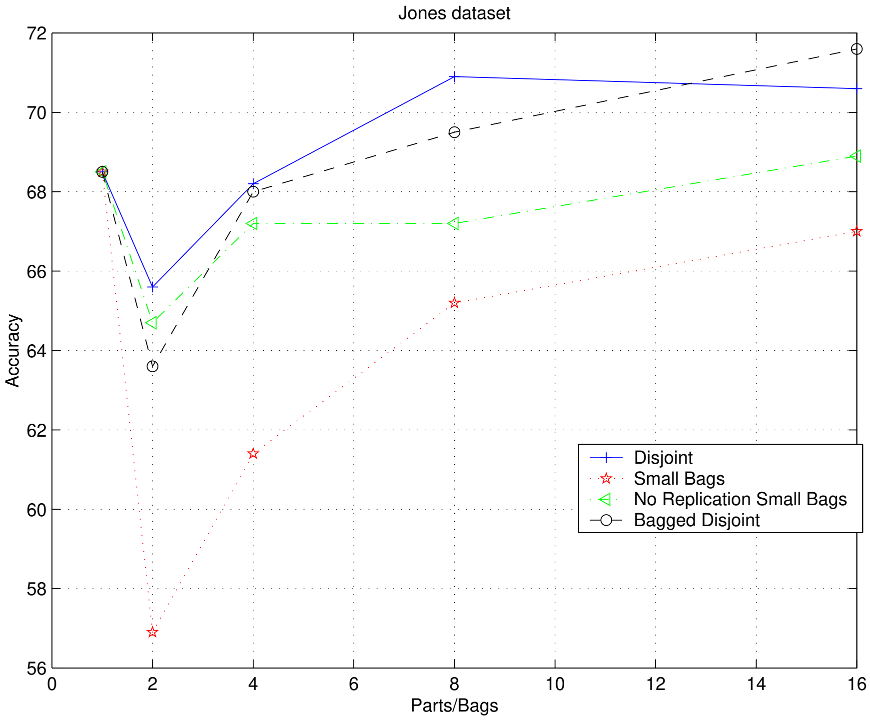

Figure 6: Results Jones' PDB dataset.

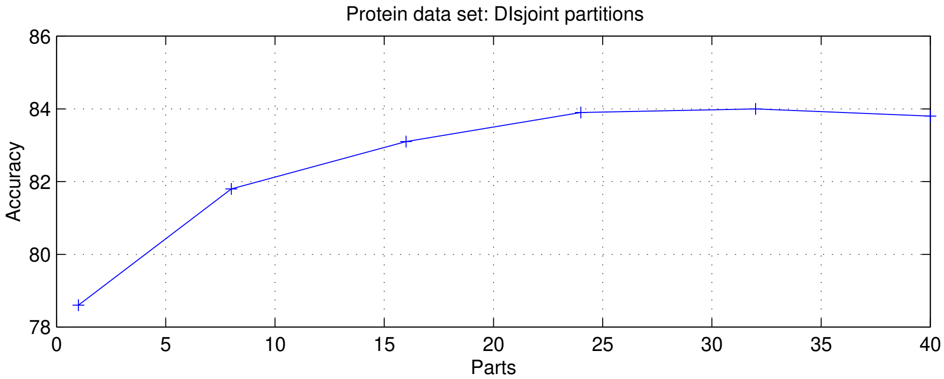

Figure 7: Results of Partitioning on PDB dataset.

classifier	overall accuracy	"hard" subset	"easy" subset
cascade correlation NN: 75 hidden units, no extra bit	69.7%	68.4%	71.2%
cascade correlation NN: 75 hidden units, extra bit	71.3%	69.3%	73.6%
C4.5 BDT: no pruning, no extra bit	68.0%	57.0%	80.0%
C4.5 BDT: no pruning, extra bit	68.9%	58.0%	80.6%
C4.5 BDT: default pruning, no extra bit	68.4%	58.0%	79.7%
C4.5 BDT: default pruning, extra bit	69.2%	58.5%	80.9%
C4.5 BDT: max pruning, no extra bit	68.8%	61.4%	79.6%
C4.5 BDT: max pruning, extra bit	70.3%	62.3%	79.1%
Jones' on-line classifier	80.1%	79.0%	81.3%
C4.5 BDT committee, 17-window	84.1%	72.6%	96.6%

Table 2: Performance Comparison of Individual DTs and ANNs, With Reference Points.

indicate whether or not the window centered at that amino acid lies within the chain. For amino acids within half the window size of the end of the protein, part of the window centered at that position falls off the end of the protein. Feature values for the amino acid positions that fall off the end of the protein are padded with zeroes. The "extra bit" indicated with some entries in Table 2 is an added feature for each amino acid that indicates whether part of its feature vector consists of zero-padding. Performance is consistently better by a small amount when this feature is used. Note that the percentage of the training data affected by zero-padding increases as the window size increases.

The last two rows in Table 2 compare the performance of a committee of thirty-two C4.5 decision tree classifiers with the performance of Jones' on-line classifier. The training set is the large dataset in Table 1. The dataset is divided into eight disjoint partitions in four different ways. This gives the thirty-two classifiers. The test set is our set of 146 chains (38,423 amino acids). Note how the relative performance of Jones' neural network classifier and our decision tree classifier differ. The committee of decision tree classifiers was able to outperform Jones' classifier on the overall data set, but lost to Jones' classifier by a substantial amount on the hard subset of the data. The individual decision trees in this committee of classifiers were unpruned. We are currently running additional experiments to explore the effect of using heavily pruned decision trees for the classifiers in the committee.

7. CONCLUSIONS AND DISCUSSION

The results presented here support several important conclusions. The primary 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 voting committee of N classifiers. In addition, for large data sets, the performance of the resulting committee of classifiers can be expected to exceed that of a single classifier built from all the data. In the context of the protein structure prediction problem, the performance of the decision tree is subtly influenced by the degree of pruning. Thus, it will be important to tune decision tree performance by optimizing the degree of pruning. By itself, this is not necessarily a drawback in comparison to neural networks, because neural networks must be tuned by adjusting the structure of the network and various training parameters.

In contrast to decision trees, neural networks do not seem to be as susceptible to learning a hard/easy split of the training data. We conjecture that this is an intrinsic reflection of how decision trees and neural networks operate. The decision tree can effectively split off a subset of the training data (a region of feature space) to be handled differently, whereas all the training examples influence all the weights in the neural network.

Some researchers have suggested that many large-data-set 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. This should not really be surprising. With a 340-dimension feature space, even 3.6 million examples can result in a "sparse" population.

8. ACKNOWLEDGEMENTS

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, and by NSF grant EIA-9729904 (research equipment grant).

9. REFERENCES

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

[2] C. Blake and C. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1998.

[3] K. Bowyer, N. Chawla, T. Moore, L. Hall, and W. Kegelmeyer. A parallel decision tree builder for mining very large visualization datasets. In *Proceedings of IEEE-System, Man and Cybernetics*, 2000.

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

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

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

[7] 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.

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

[9] J. Darlington, Y. Guo, J. Sutiwaraphun, and H. To. Parallel induction algorithms for data mining. In *Advances in Intelligent Data Analysis Reasoning about Data, Second International Symposium, IDA-97. Proceedings*, pages 437–445, 1997.

[10] 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.

[11] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, 1990.

[12] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[13] L. Hall, K. Bowyer, W. Kegelmeyer, T. Moore, and C. Chao. Distributed learning on very large data sets. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[14] L. Hall, N. Chawla, and K. Bowyer. Decision tree learning on very large data sets. In *IEEE Conference on Systems, Man and Cybernetics*, pages 2579–2584, 1998.

[15] L. Hall, N. Chawla, K. Bowyer, and W. Kegelmeyer. Learning rules from distributed data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

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

[17] L. L. N. Laboratories. Protein structure prediction center. *http://predictioncenter.llnl.gov/*.

[18] S. N. Labs. ASCI RED, the world's first teraops supercomputer. *http://www.sandia.gov/ASCI/Red/*.

[19] A. W. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

[20] T. Oates and D. Jensen. Large datasets lead to overly complex models: an explanation and a solution. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. August 1998*, 1998.

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

[22] F. 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.

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

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

[25] J. Shafer, R. Agrawal, and M. Mehta. Sprint: A scalable parallel classifier for data mining. In *Proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India*, pages 1–12, 1996.