ELSEVIER

Pattern Recognition Letters xxx (2002) xxx–xxx

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Distributed learning with bagging-like performance

Nitesh V. Chawla [a], Thomas E. Moore [a], Lawrence O. Hall [a,*], Kevin W. Bowyer [b], Philip Kegelmeyer [c], Clayton Springer [c]

[a] *Department of Computer Science and Engineering, University of South Florida, 4202 East Flower Avenue, Tampa, FL 33620 USA*
[b] *Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA*
[c] *Sandia National Laboratories, Biosystems Research Department, P.O. Box 969, MS 9951, Livermore, CA 94551-0969, USA*

Received 11 December 2001; received in revised form 3 June 2002

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 with decision tree and neural network classifiers on various datasets show that, given the same size partitions and bags, disjoint partitions result in performance equivalent to, or better 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. Hence, bagging with samples the size of the data is impractical. Our results indicate that, in such applications, the simple approach of creating a committee of n classifiers from disjoint partitions each of size $1/n$ (which will be memory resident during learning) in a distributed way results in a classifier which has a bagging-like performance gain. The use of distributed disjoint partitions in learning is significantly less complex and faster than bagging.

Keywords: Distributed learning; Bagging; Large data sets; Ensembles; Multiple classifiers

1. Introduction

Many data mining applications use data sets that are too large to be handled in the memory of the typical computer (Shafer et al., 1996; Darlington et al., 1997; Chan and Stolfo, 1993; Provost et al., 1999; Moore and Lee, 1998; Bowyer et al., 2000; Hall et al., 1999, 2000; Oates and Jensen, 1998). One possible approach is to subsample the data in some manner (Provost et al., 1999; Breiman, 1999). However, it can be difficult a priori to know how to subsample so that accuracy is not affected. Also, recent work by Perlich et al. (in press) has shown that classifier accuracy tends to increase with more training data even for large data sets. Another possible approach is to partition the original data into smaller subsets, and form a committee of classifiers (Chan and Stolfo, 1993; Provost and Hennessy, 1996). One advantage of this approach is that the partition size can simply be set at whatever amount of the original

* Corresponding author. Tel.: +813-974-4195; fax: +813-974-5456.
E-mail addresses: chawla@csee.usf.edu (N.V. Chawla), tmoore4@csee.usf.edu (T.E. Moore), hall@csee.usf.edu (L.O. Hall), kwb@cse.nd.edu (K.W. Bowyer), wpk@ca.sandia.gov (P. Kegelmeyer), csprin@ca.sandia.gov (C. Springer).

0167-8655/02/$ - see front matter © 2002 Published by Elsevier Science B.V.
PII: S0167-8655(02)00269-6

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 almost always result in equal or (usually) improved performance over a single classifier created on all of the original data (Breiman, 1996; Quinlan, 1996; Bauer and Kohavi, 1999). The success of bagging suggests that it might be a useful approach to create accurate 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 will not fit in the main memory of the typical computer, then none of the thirty or more bags one might create will fit. This raises the question of which particulars of the bagging approach are essential in the context of large data sets. We consider the question of whether simple partitioning of the training data into tractable-size subsets can produce an ensemble of classifiers with accuracy equal to that of a single classifier built on all of the data. 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. In other words, it is the committee of classifiers that is essential and bootstrap aggregation to form the individual classifiers is not essential. Further, it is straightforward to create one or several different disjoint partitions of data and the process is rapid.

Classifiers can be built in a distributed way on disjoint partitions. Each of n classifiers can be learned on a separate processor in parallel. Further, for large enough partitions (bags) the performance of the resulting classifier will meet or exceed that of a classifier built on all the data. The time required to build an ensemble of n classifiers which are learned independently on n processors without communication will be the longest time required to learn a single classifier on a processor.

In Section 2, we discuss related work. In Section 3, we describe our experimental setup for small data sets from the UC Irvine repository (Blake and Merz, 1998), as well as the setup for experiments with a mid-size and large data set from our own research. We also describe the base classifiers and computing systems used in the experiments. Section 4 contains the experimental results. Finally, Section 5 contains a discussion of the results and conclusions that can be drawn.

2. Literature review

Bagging (Breiman, 1996) 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 (1996) experimented with bagging on various datasets and found that bagging substantially improved accuracy. However, the experiments were performed on *small* datasets, the largest one being 20,000 examples.

Domingos (1997) 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. 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 that of the single learner.

In (Street and Kim, 2001) an ensemble of classifiers are built from training data which is treated as a stream. Each classifier is trained on a fixed amount of data from the stream. The size of the ensemble is fixed at 25 classifiers. Classifiers "compete" for entry into the ensemble based on their accuracy and diversity. This approach allows an ensemble classifier to be built from an unlimited amount of training data. It also facilitates building an ensemble classifier which might be built on data with temporal dependencies, where the concept to

be modeled may vary over time. In their experiments, the ensemble classifier was usually slightly less accurate than a classifier which was built with as much data as the current ensemble had used. In our work larger training sets are used.

In (Breiman, 1999) classifiers were built on small randomly chosen subsets of an overall training set. This approach can deal with extremely large training datasets, but requires many classifiers because it uses somewhere around 800 examples per classifier in the discussed experiments. Also, the process of continually selecting small subsets can be computationally problematic for very large datasets. This is different from the approach we are looking at, in which the number of training samples may be as large as main memory space available.

Chan and Stolfo (1995) compared arbiter and combiner strategies by applying a learning algorithm to disjoint subsets of data. An arbiter scheme uses a learned representation of which classifier to choose given an example and a combiner takes classifier outputs of a test example as input and produces a classification. The described experiments showed that the arbiter strategy can sometimes better sustain 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 (Chan and Stolfo, 1995) and so there is reason to expect that accuracy can be maintained.

Chan and Stolfo (1996) relaxed their definition of strict disjoint subsets by allowing a small amount of overlap across the subsets. On the datasets DNA Splice Junction with 3190 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.

Domingos (1996) describes how a specific-to-general rule induction system (RISE) was sped up by applying it to disjoint training sets. This allowed the time required for learning to become linear in the number of examples. The resulting rule based classifiers were voted (with some weighting) in an approach very similar to bagging. The major difference was that the size of each training data set was much smaller than the original. On a set of seven data sets from the Irvine repository using disjoint partitions of between 100 and 500 examples they found that the resulting voting performance was generally as good as or better than applying RISE to all the data.

Dietterich (2000) describes how an ensemble of decision trees was built from a single unmodified training set. Diversity in the trees was obtained by randomly choosing a test at each internal node from among the top k tests (ranked by information gain, with k typically 20). Each tree was generally suboptimal, but when voted as an ensemble they provided a bagging-like and sometimes better accuracy gain. This result seems to suggest that bagging-like performance can be obtained from a set of diverse classifiers (in the sense that they make errors on different examples) which may each be suboptimal (in the sense that they are not as good as a classifier built on all the data without any manipulation), but similar in accuracy.

Hall et al. (2000) learned decision trees using disjoint partitions of data and then combined the classifiers. It was found that when 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. (1999) found that subsampling 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 no longer improved. 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 later in the discussion section, our

results indicate that simple subsampling to produce one smaller training set is not a profitable strategy for the larger datasets that we consider. However, more complicated subsampling strategies may be useful.

3. Experiments

Three sets of experiments were performed. The first uses five 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 mid-size dataset of almost 230,000 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.

Based on the first two sets of experiments, the disjoint partitioning approach is identified as offering equivalent performance for a given size of partition/bag. 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 and bagging

We investigated four different approaches to creating a committee of classifiers from an original data set (see Fig. 1 for an illustration). One approach is to simply randomly 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.

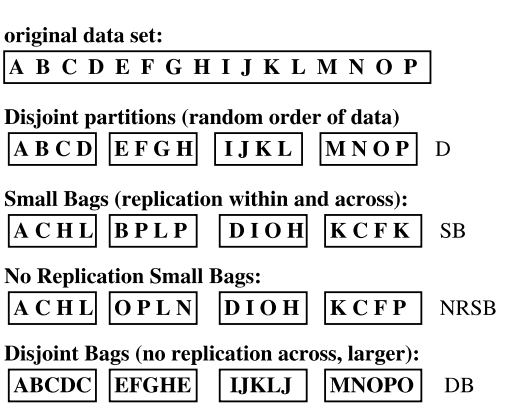

Fig. 1. Four approaches to a committee of classifiers.

The second approach is to create N bags of size $(1/N)$th of the data. Each bag is created independently, by random sampling is done 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. When 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 partition to form a "disjoint bagged" training set. 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, but through the effect of allowing larger bag size.

In addition to the above four approaches, we also ran "true bagging" on each of the five 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 using smaller bags, but the point is to provide a baseline performance comparison for the other approaches.

For each experiment in which voting is used, ties are broken by assigning the example to the largest class (in the training set) participating in the tie.

3.2. Datasets

Three of the small data sets are from the UCI repository (Blake and Merz, 1998), one is from the ELENA project, [1] and one is from our own research. The mid-size dataset comes from the problem of predicting the secondary structure of proteins. It is part of the training data set used with a neural network that won the CASP-3 secondary structure prediction contest (Jones, 1999). We concatenated his train set one and test set one as our overall training set. This dataset contains almost 230,000 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 log likelihood of the amino acid being any one of twenty basic amino acids. Using a window of size 15 centered around the target amino acid and an extra bit for each window to indicate an N or C terminus (beginning or end of chain) gives a feature vector of size 315.

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

Our large dataset also comes from the protein database (PDB) (Berman et al., 2000) used in the CASP contests (Lawrence Livermore National Laboratories, 2001). For 18,098 protein chains taken from the PDB, there are a total of 3,679,152 amino acids for structure prediction. Using a window of size seventeen centered around the target amino acid (without an extra N/C terminus bit), we have a feature vector of size 340. 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 crystal structures with resolution 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 amino acid basis.

The size and class distribution of the datasets are summarized in Table 1. Note that the experiments include both two-class (Mammography, Phoneme) and multi-class (Letter, PenDigits, SatImage) datasets. They also include datasets that are approximately balanced (Letter, PenDigits) and those that are skewed (Mammography, Phoneme, 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 appears to begin to starve the classifiers for training data. For the experiments on the small and mid-size datasets, the reported results are calculated from 10-fold cross-validation.

3.3. Base classifiers and computing systems

For the experiments on the small and mid-size datasets, release eight of the C4.5 decision tree system (Quinlan, 1992) and the cascade correlation neural network learning code (Fahlman and Lebiere, 1990) were run on standard SUN workstations and a 24-node Beowulf cluster computer (Mimir). Mimir consists of 900 Mhz Athlon pro-

Table 1
Data sets sizes and class distributions

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				
Phoneme dataset (ELENA)—5404 samples in two classes					
0:3818	1:1586				
PenDigits dataset (UCI)—10,992 samples in ten classes					
0:1143	1:1143	2:1141	3:1055	4:1144	5:1055
6:1056	7:1142	8:1055	9:1055		
SatImage dataset (UCI)—6435 samples in six classes					
1:1533	2:703	3:1358	4:626	5:707	7:1508
Mammography dataset—11,183 samples in two classes					
1:10,923	2:260				
Jones' PDB dataset—227,260 samples in three classes					
H:75,455		C:100,909		E:50,896	
PDB dataset—3,619,461 samples in three classes					
H:1,254,335		C:1,537,261		E:827,865	

cessors each with 512 Mb of memory and processors are connected with 100 Bt Ethernet. The neural network was only applied to the small data sets due to its extremely long training times and larger memory requirements.

The one run of the large dataset to produce a single classifier was done on a single node of a 64-processor SGI IRIX64 with 32 GB of main memory at Sandia National Labs, also using the standard C4.5 release eight. Creating the one decision tree on the large dataset took approximately *thirty days* on the SGI and was not attempted with the much slower neural network learning code.

The experiments using partitions of the large dataset were run on the DOE's "ASCI Red" parallel supercomputer (Sandia National Labs, 1998). The ASCI Red has 4640 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 8 h; that is, eight processors running in parallel for 8 h. It is possible to create 32 distributed trees built on 1/8 size partitions of the data in approximately 10 h.

4. Results

4.1. C4.5 on small data sets

Figs. 2–6 summarize the experimental comparison of the different approaches on the small datasets detailed in Table 1 for C4.5. 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 average paired difference across the 10 folds in the 10-fold cross-validation, with standard error indicated. Note the zero mean accuracy mid-line; if a point is above

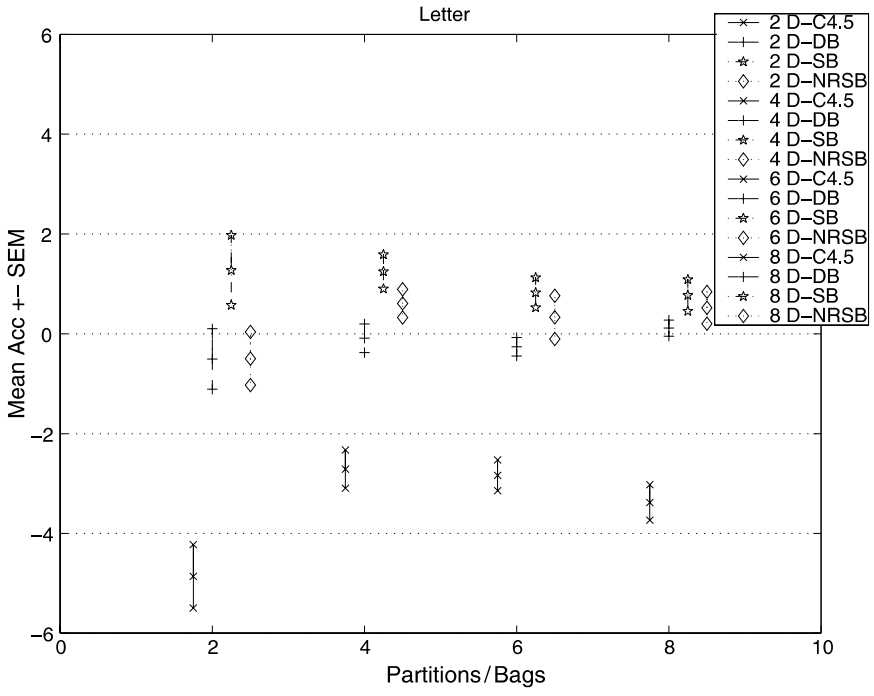

Fig. 2. Comparison on Letter dataset with C4.5.

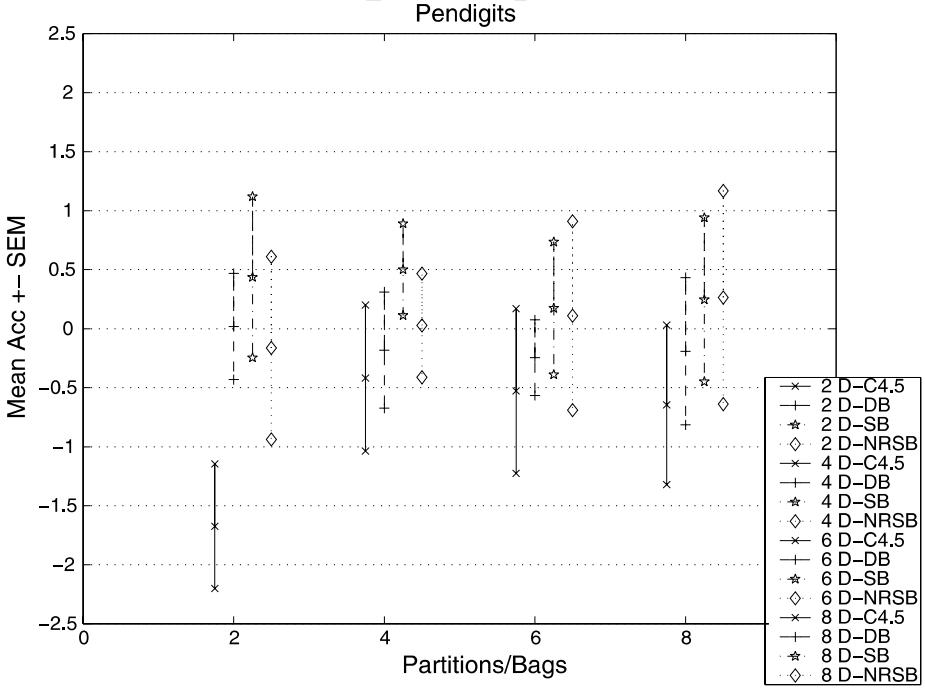

Fig. 3. Comparison on PenDigits dataset with C4.5.

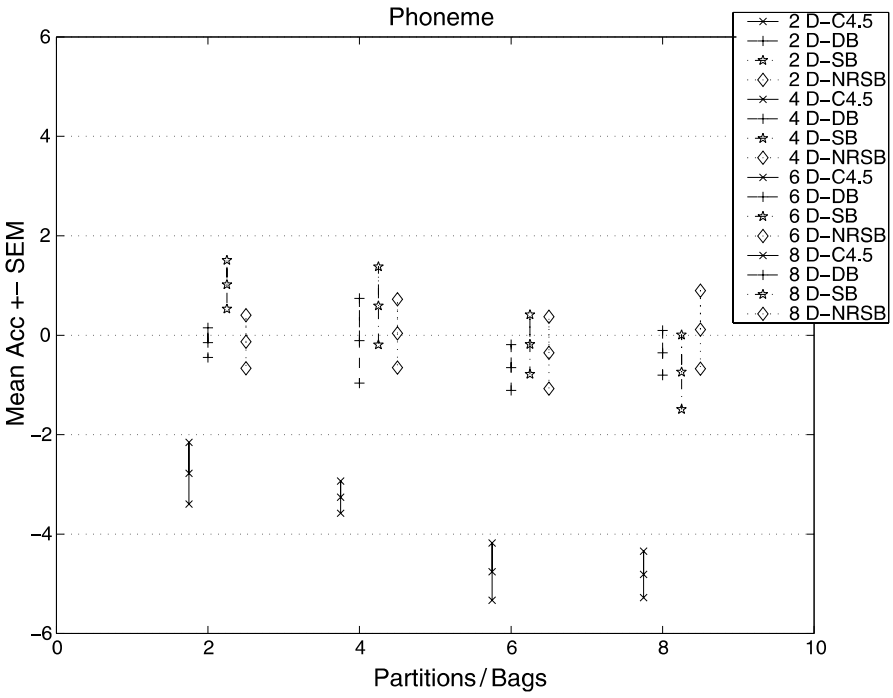

Fig. 4. Comparison on Phoneme dataset with C4.5.

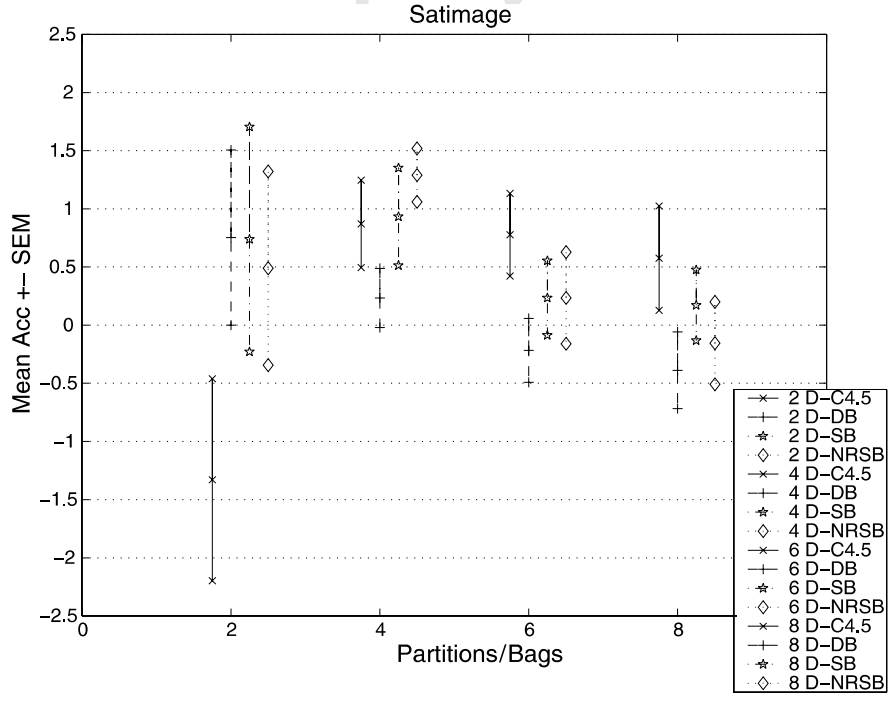

Fig. 5. Comparison on SatImage dataset with C4.5.

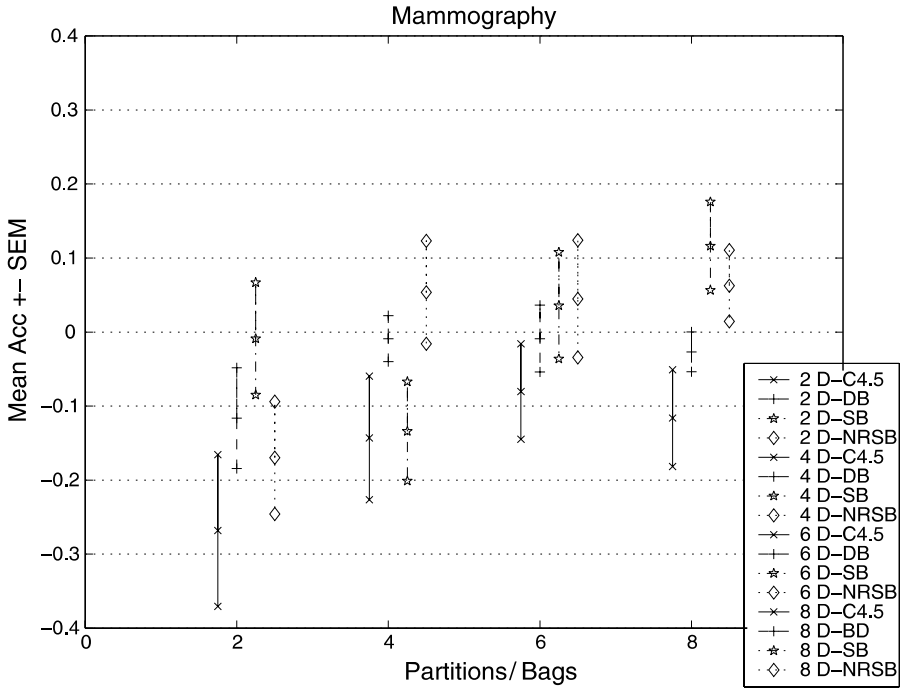

Fig. 6. Comparison on Mammography dataset with C4.5.

the line, then the first of the alternatives examined is superior. If the point is below, then the second was best. And the extent to which the error bar spans the mid-line is the extent to which the results are inconclusive.

As an example, the first cluster of four data points on the plot in Fig. 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 in a number of instances beat small bags. It appears to make little difference whether the small bags are created by sampling with or without replacement. The ensembles created from "bagged disjoints" appear to generally perform slightly better than those created from simple disjoint partitions, but then the training sets for the individual decision trees have repeated examples which give some examples greater "weight".

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. Tables 3 and 4 show the accuracy results obtained by learning four classifiers each built from 1/4 of a

Table 2
Data points for "true bagging" results

Dataset	C4.5	50 bags	75 bags	100 bags
Phoneme	86.50	89.15	89.02	88.15
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 3
Four partitions/bags accuracy in percentage using C4.5

Dataset	D	SB	DB	NRSB
Phoneme	83.25	82.66	83.36	83.22
SatImage	87.24	86.31	87.00	85.95
PenDigits	96.15	95.65	96.33	96.12
Mammography	98.37	98.51	98.38	98.32
Letter	85.44	84.19	85.525	84.83

Table 4
Four partitions/bags accuracy in percentage using CC

Dataset	D	SB	DB	NRSB
Phoneme	83.75	84.06	83.92	83.59
SatImage	89.43	88.97	89	89.26
PenDigits	90.89	91.01	91.73	90.86
Mammography	97.78	98.19	98.43	98.02
Letter	82.15	81.88	81.96	82.17

disjoint four partition utilizing decision trees and neural networks, respectively. Each value in the table is an average over a 10-fold cross-validation. The results from a four partition are representative, as more partitions cause greater "data starvation" in some of the small domains. Bagging with 50–100 bags clearly outperforms the small ensembles. 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 50 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 dataset is too large to handle conveniently in the memory of the typical computer, the dataset must be broken into some number of practically sized, though not "small," chunks. The question addressed here is whether there is any advantage in creating the practically sized chunks using some bagging-like approach, or whether simple partitioning is sufficient.

4.2. Cascade correlation on small datasets

Figs. 7–11 summarize the experimental comparison of the different approaches on the small

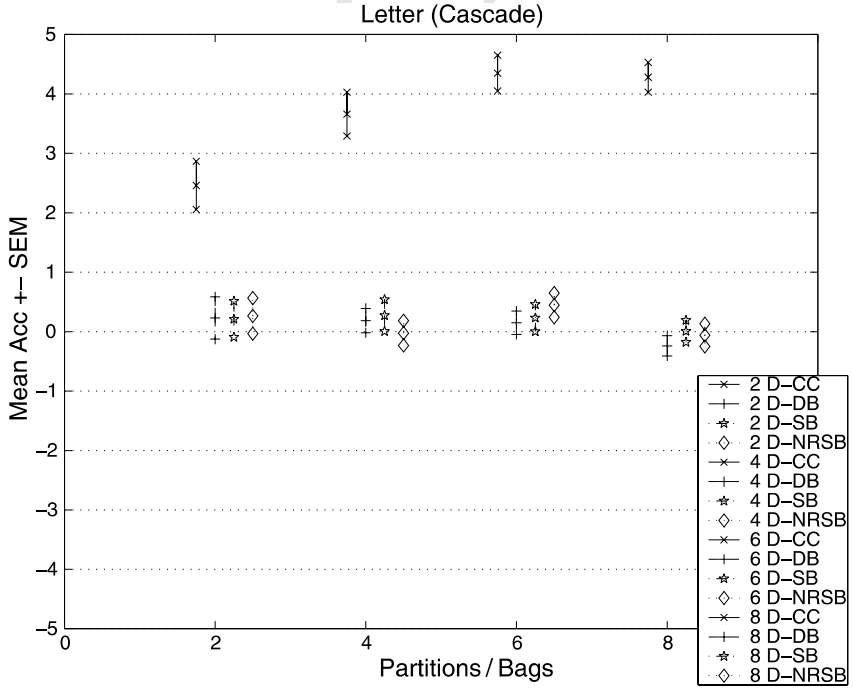

Fig. 7. Comparison on Letter dataset with cascade correlation.

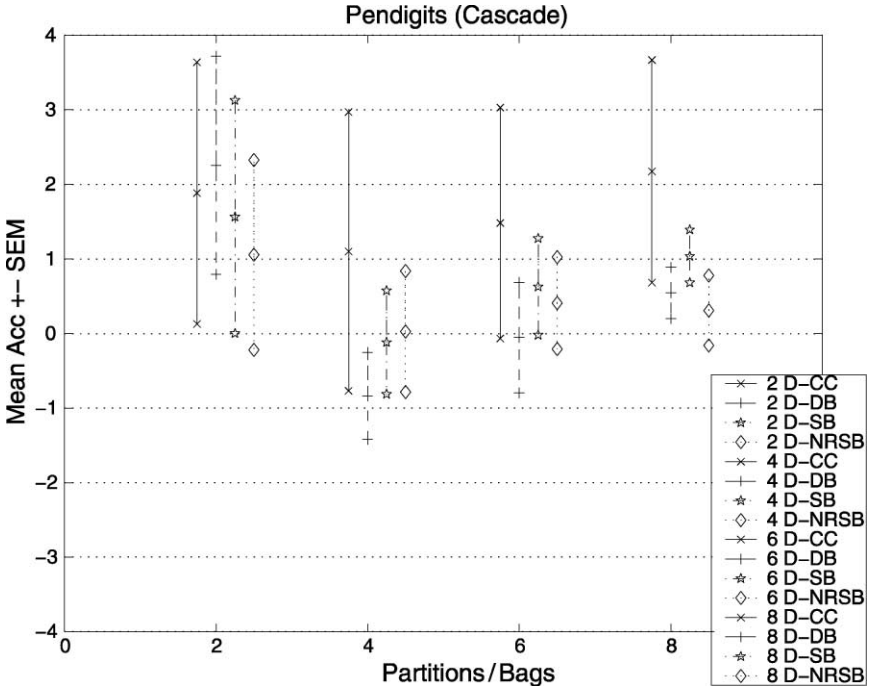

Fig. 8. Comparison on PenDigits dataset with cascade correlation.

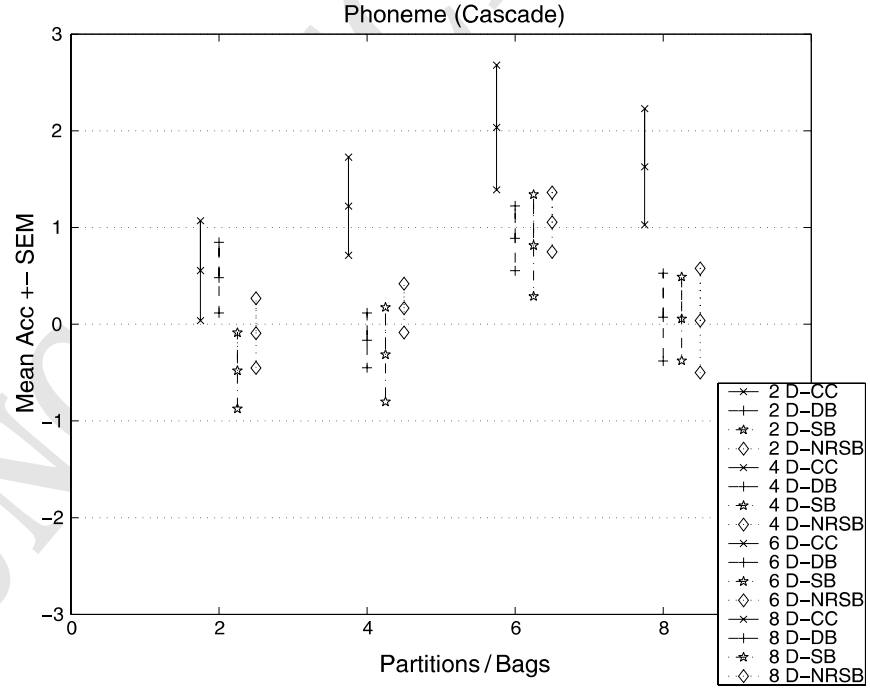

Fig. 9. Comparison on Phoneme dataset with cascade correlation.

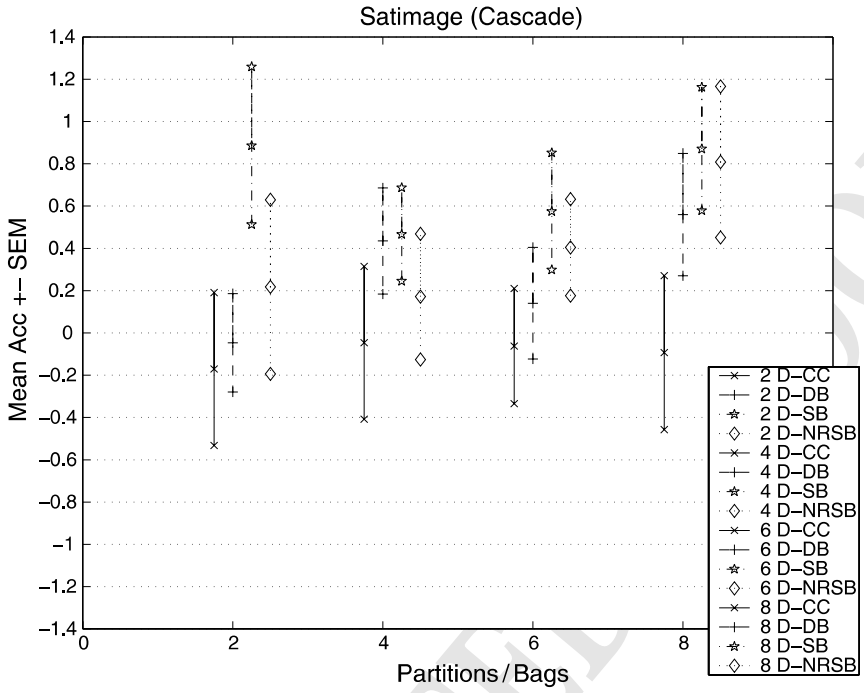

Fig. 10. Comparison on SatImage dataset with cascade correlation.

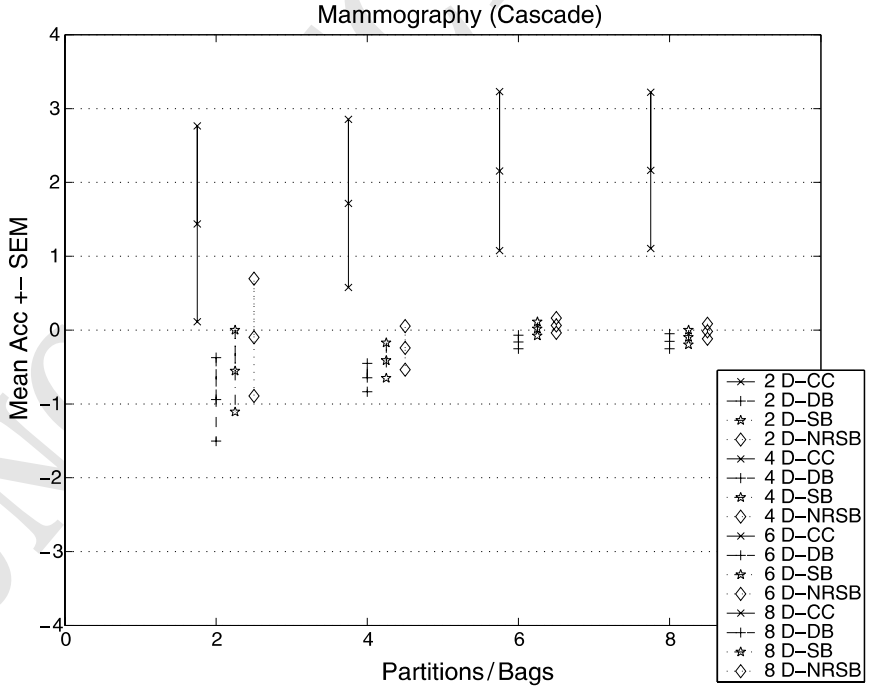

Fig. 11. Comparison on Mammography dataset with cascade correlation.

datasets detailed in Table 1 using cascade correlation. The data was not normalized. The results are similar to those obtained with C4.5. The disjoint partitions are generally as good or better than small bags. The exception is that they lose to small bags for a four partition on the Mammography dataset and to DB at every partition in the Mammography data set. The Mammography dataset is highly skewed with a small minority class and the repeated examples in the disjoint bag data sets seemed to make an important, if minor, difference in accuracy. The other exception is on the PenDigits dataset where the DB approach is better for a four partition.

It is interesting that the ensemble of neural networks is generally no worse than learning from all the data even when partitioning up the small data into eight subsets. In fact, the ensemble is always better than a single neural network for the Letter data set. In general, the ensembles built from disjoint partitions are no worse than those built from bags of the same size for cascade correlation neural networks.

4.3. Results on mid-size data set

The comparison of the four approaches on the mid-size dataset are shown with C4.5 in Fig. 12. The results are the means of paired differences and the standard error from a 10-fold cross-validation from 4 to 16 disjoint partitions. In all cases the ensemble classifier learned from disjoint partitions resulted in a significantly better classifier than applying C4.5 to all of the data.

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 examples in each bag, offers any hint of performance improvement over disjoint partitions. The "bagged disjoints" have the same number of distinct examples with a few repeat examples which essentially gives the repeat examples greater weight.

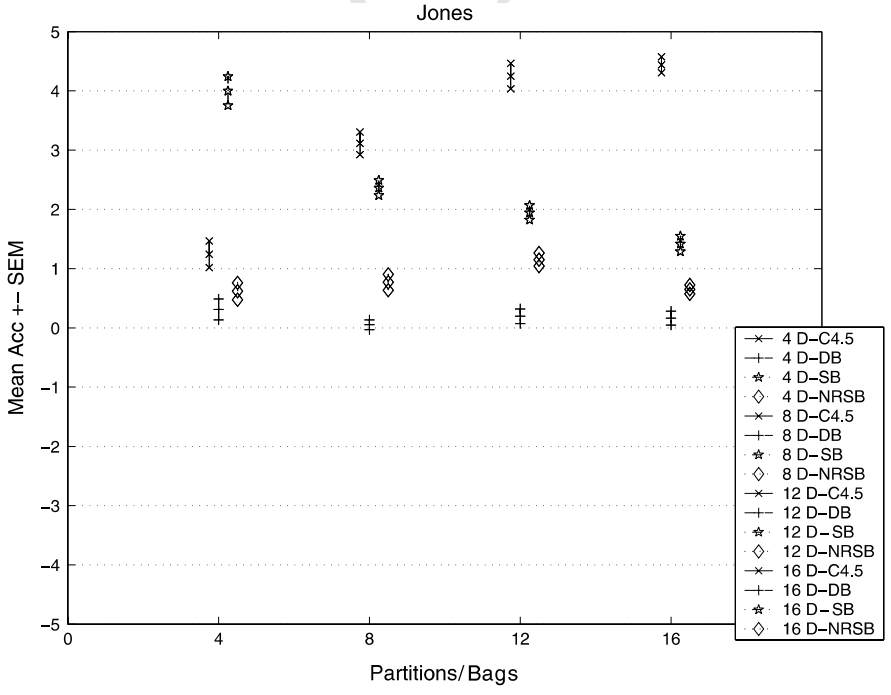

Fig. 12. Results on (Jones') "medium" PDB dataset.

4.3.1. Timing

We did a detailed comparison of the time required to create an ensemble on the Beowulf cluster for the Jones data set. Table 5 shows the average and standard deviation over 10 trials. For disjoint partitioning for a four partition, there is an average of a 9.4 times speed up over learning a decision tree on all the data, which increases to 26.6 times for an eight partition, 46.1 times for a 16 partition and 71.3 times for a 24 partition. While the speed gain is impressive, it is actually true that for very large data sets that do not fit in main memory all the data cannot be practically used unless a distributed approach is pursued.

Table 5
Jones dataset: average and standard deviation of CPU time (longest time taken to learn a decision tree on a partition/bag by a processor) in seconds for 10-folds

Approach	Average time	Standard deviation
C4.5	9094.38	137.11
4-D	963.61	14.42
4-SB	990.61	17.04
4-NRSB	967.96	21.22
8-D	342.172	8.46
8-SB	352.04	8.98
8-NRSB	339.51	5.45
12-D	197.19	5.52
12-SB	202.55	3.89
12-NRSB	196.98	4.72
16-D	127.48	3.6
16-SB	130.62	1.32
16-NRSB	128.50	2.7

Convention = D: disjoint; SB: small bag; NRSB: no-replication small bag. For instance, four-dimensional means four disjoint partitions.

4.4. Partitioning results on large dataset

Fig. 13 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, and 32. 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. The $N = 8$ point represents a straightforward disjoint partitioning, as was used with the smaller datasets. For

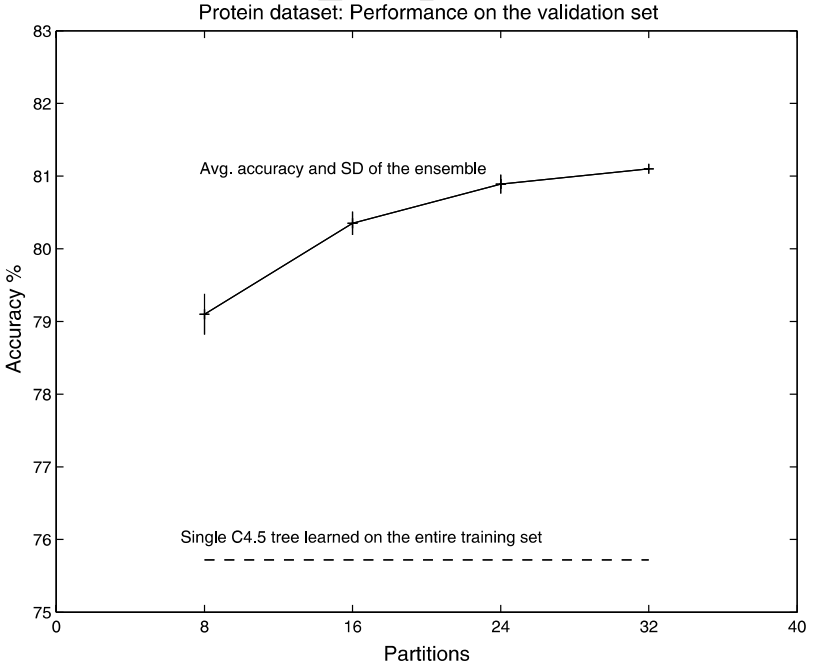

Fig. 13. Partitioning results for large PDB dataset.

the committees of 16, 24, 32, and 40 classifiers, we varied the earlier methodology to make use of multiple different partitions of the dataset. 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 created on the original eight-partition. We report an average and standard deviation over five trials of different ensembles of trees of the appropriate size.

The average accuracy per amino acid of a single classifier trained on (1/8)th of the large dataset is 71.21%. A single decision tree created using all the data performs substantially better than this, 75.72% versus 71.21%. At the same time, an average committee of eight classifiers created on (1/8)ths of the data performs substantially better than a single tree created on all the data, 79.1% versus 75.72%.

It took approximately 30 days to create a single tree from all the data and an average of 10 h to create an ensemble of from 8 to 32 trees. We did some experiments to determine the relative speeds of the SGI machine vs. an ASCI Red processor and found the ratio to be 1.92 for the same size decision tree learning problem. So, the Red processor is significantly faster. Normalizing for processor speed, the distributed learner can be created approximately 37 times faster than learning from all the data.

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 perform at least as well as a committee 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 we did in Section 4.4. A typical result from partitioning the data multiple times is shown in Table 6 for the ''small'' Letter data set. The data set is broken into a two partition five different ways. A 10-fold cross-validation is done, meaning that the test set is left out and then the training set partitioned into halves (eight different ways). From the table, it is clear that accuracy continues to increase and with 16 partitions we are approaching the accuracy of pure bagging (93.8% for 100 bags), considerably outperforming a C4.5 generated decision tree from the whole data set.

Results obtained here seem to support the position that bagging results depend simply on obtaining a diverse set of classifiers (Breiman, 1996; Chawla et al., 2001; Dietterich, 2000; Domingos, 1996). 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.

Table 6
Multiple two partitions on the Letter data set for a 10-fold cross-validation using C4.5

Number of two partitions	Accuracy	Standard deviation
1	83.645	1.078
2	89.225	0.522
3	91.21	0.589
4	91.885	0.461
5	92.26	0.427

C4.5 is 88.1% accurate in a 10-fold CV.

Some researchers have suggested that many large-data-set problems can be solved using only a fraction of the data, perhaps by simple subsampling. However, recent work has suggested that all the data (even for very large training data sets) may result in the maximal accuracy classifier (Perlich et al., in press). 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 71.21%, whereas a single classifier trained on all the data gave 75.72%, indicates that the original data could not be profitably subsampled 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.

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. A portion of the work was presented at CVPR 2001. Thanks to Robert Banfield for his help with experimental work. Thanks also to the referees for their insightful comments and guidance.

References

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

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E., 2000. The protein data bank. Nucleic Acids Research 28, 235–242, Available from <http://www.pdb.org/>.

Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases. Available from <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

Bowyer, K.W., Chawla, N.V., Moore Jr., I.E., Hall, L.O., Kegelmeyer, W.P., 2000. A parallel decision tree builder for mining very large visualization datasets. In: IEEE System, Man, and Cybernetics Conference, pp. 1888–1893.

Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

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

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

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

Chan, P., Stolfo, S., 1996. Scaling learning by meta-learning over disjoint and partially replicated data. In: Ninth Florida Artificial Intelligence Research Symposium, pp. 151–155.

Chawla, N., Moore Jr., T.E., Bowyer, K.W., Hall, L.O., Springer, C., Kegelmeyer, W.P., 2001. Bagging is a small-data-set phenomenon. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 684–689.

Darlington, J., Guo, Y., Sutiwaraphun, J., To, H., 1997. Parallel induction algorithms for data mining. In: Advances in Intelligent Data Analysis Reasoning about Data, Second Internat. Symposium, IDA-9 7. Proc., pp. 437–445.

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

Domingos, P., 1996. Using partitioning to speed up specific-to-general rule induction. In: Proc. AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press, Portland, OR, pp. 29–34.

Domingos, P., 1997. Why does bagging work? A Bayesian account and its implications. In: Proc. Third Internat. Conf. on Knowledge Discovery and Data Mining.

Fahlman, S.E., Lebiere, C., 1990. The cascade-correlation learning architecture. In: Advances in Neural Information Processing Systems 2. Morgan Kaufmann, Los Altos, CA.

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

Hall, L.O., Chawla, N.V., Bowyer, K.W., Kegelmeyer, W.P., 1999. Learning rules from distributed data. In: ACM SIGKDD Workshop on Large-Scale Parallel Data Mining Systems.

Jones, D.T., 1999. Protein secondary structure prediction based on decision-specific scoring matrices. Journal of Molecular Biology 292, 195–202.

Lawrence Livermore National Laboratories, 2001. Protein structure prediction center. Available from <http://predictioncenter.llnl.gov/>.

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

Oates, T., Jensen, D., 1998. Large datasets lead to overly complex models: an explanation and a solution. In: Proc. Fourth Internat. Conf. on Knowledge Discovery and Data Mining, August.

Perlich, C., Provost, F., Simonoff, J., 2002. Tree induction vs. logistic regression: a learning-curve analysis. J. Machine Learning Res., in press.

Provost, F.J., Hennessy, D.N., 1996. Scaling up: Distributed machine learning with cooperation. In: Proc. 13th National Conf. on Artificial Intelligence, AAAI'96, pp. 74–79.

Provost, F.J., Jensen, D., Gates, T., 1999. Efficient progressive sampling. In: Proc. Fifth ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, pp. 23–32.

Quinlan, J.R., 1996. Bagging, boosting, and C4.5. In: Proc. Thirteenth National Conf. on Artificial Intelligence, pp. 725–730.

Quinlan, J.R., 1992. C4.5: Programs for Machine Learning. Morgan Kaufinann, San Mateo, CA.

Sandia National Labs, 1998. ASCI RED, the world's first teraops supercomputer. Available from <http://www.sandia.gov/ASCI/Red/>.

Shafer, J., Agrawal, R., Mehta, M., 1996. Sprint: A scalable parallel classifier for data mining. In: Proc. 22nd VLDB Conf., Mumbai (Bombay), India, pp. 1–12.

Street, W.N., Kim, Y., 2001. A streaming ensemble algorithm (SEA) for large-scale classification. In: Provost, F., Srikant, R. (Ed.), KDD'01, San Francisco, CA, pp. 377–382.