Journal of Machine Learning Research 5 (2004) 421-451 Submitted 3/03; Revised 11/03; Published 4/04

Learning Ensembles from Bites: A Scalable and Accurate Approach

Nitesh V. Chawla* NITESH.CHAWLA@CIBC.CA
Customer Behavior Analytics, CIBC
Commerce Court East, 11th Floor
Toronto, ON M5L 1A2, Canada

Lawrence O. Hall HALL@CSEE.USF.EDU
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620, USA

Kevin W. Bowyer KWB@CSE.ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
384 Fitzpatrick Hall
Notre Dame, IN 46556, USA

W. Philip Kegelmeyer WPK@CA.SANDIA.GOV
Sandia National Labs, Biosystems Research Department
P.O. Box 969, MS 9951
Livermore, CA 94551-0969, USA

Editor: Claude Sammut

Abstract

Bagging and boosting are two popular ensemble methods that typically achieve better accuracy than a single classifier. These techniques have limitations on massive data sets, because the size of the data set can be a bottleneck. Voting many classifiers built on small subsets of data ("pasting small votes") is a promising approach for learning from massive data sets, one that can utilize the power of boosting and bagging. We propose a framework for building hundreds or thousands of such classifiers on small subsets of data in a distributed environment. Experiments show this approach is fast, accurate, and scalable.

Keywords: ensembles, bagging, boosting, diversity, distributed learning

1. Introduction

The last decade has witnessed a surge in the availability of massive data sets. These include historical data of transactions from credit card companies, telephone companies, e-commerce companies,

*. This is the author to whom correspondence should be addressed.

and financial markets. The relatively new bioinformatics field has also opened the doors to extremely large data sets such as the Protein Data Bank (Berman et al., 2000). The availability of very large databases has constantly challenged the machine learning and data mining community to come up with fast, scalable, and accurate approaches (Provost and Kolluri, 1999; Fayyad et al., 1996). As Neal Leavitt notes (Leavitt, 2002, p. 22):

> "The two most significant challenges driving changes in data mining are scalability and performance. Organizations want data mining to become more powerful so that they can analyze and compare multiple data sets, not just individual large data sets, as is traditionally the case."

Very large data sets present a challenge for both humans and machine learning algorithms. Machine learning algorithms can be inundated by the flood of data, and become very slow in learning a model or classifier. Moreover, along with the large amount of data available, there is also a compelling need for producing results *accurately* and *fast*. Efficiency and scalability are, indeed, the key issues when designing data mining systems for very large data sets.

The machine learning community has essentially focused on two directions to deal with massive data sets: data subsampling (Musick et al., 1993; Provost et al., 1999), and the design of parallel or distributed approaches capable of handling all the data (Chan and Stolfo, 1993; Provost and Hennessy, 1996; Hall et al., 1999; Chawla et al., 2000). The subsampling approaches build on the assumption that "we don't really need all the data." The KDD-2001 conference (ACM, 2001) conducted a panel on subsampling, which overall offered positive views of subsampling. However, given 100 gigabytes of data, subsampling at 10% can itself pose a challenge. Other pertinent issues with subsampling are: What subsampling methodology to adopt? What is the right sample size? To do any intelligent subsampling, one might need to sort through the entire data set, which could take away some of the efficiency advantages. Also, some of the data mining systems are concerned with identifying interesting patterns in a large database (Hall et al., 2000; Provost and Kolluri, 1999). In such scenarios, it could be important to have enough instances of each salient case so that the learner can identify those patterns. A lot of business analysts want to identify interesting customer patterns in the data sets, so taking a subsample might not help in such a scenario. Dan Graham, IBM's director of business-intelligence solutions, notes (Leavitt, 2002, p. 22), "Data mining yields better results if more data is analyzed." While subsampling a massive data set can simplify the learning task, it can also degrade accuracy (Perlich et al., 2003). However, one can essentially build an ensemble of subsamples and observe an improvement in accuracy (Eschrich et al., 2002).

The second of the two approaches to handling massive data is to bypass the need for loading the entire data set into the memory of a single computer. Our claim is that distributed data mining can address, to a large extent, the scalability and efficiency issues presented by massive training sets. The data sets can be partitioned into a size that can be efficiently managed on a group of processors. Partitioning the data sets into random, disjoint partitions will not only overcome the issue of exceeding memory size, but will also lead to creating an ensemble of diverse and accurate classifiers, each built from a disjoint partition, but the aggregate processing all of the data (Chawla et al., 2002b,a). This can result in an improvement in performance that might not be possible by subsampling. Chawla et al. (2002b) show that it is possible to create multiple disjoint partitions of both small and very large data sets, and approach classification accuracies achievable by popular ensemble techniques such as bagging, which is normally an approach suited for small data sets.

In this paper we show that it is also possible to learn an ensemble of classifiers from each of the random disjoint partitions of data, and combine predictions from all those classifiers to achieve high classification accuracies; in some cases similar to or better than boosting or distributed boosting (Lazarevic and Obradovic, 2002). We utilize Breiman's algorithms for pasting small votes: *Ivote* and *Rvote* (Breiman, 1999). In pasting Rvotes, small random training sets are constructed from the data set and classifiers are learned. In pasting Ivotes, each subsequent small training set is constructed by importance sampling based on the quality of classifiers constructed so far. That is, the misclassified cases are given a higher probability of selection than the correctly classified cases over the learning iterations. The classifiers learned are then uniformly voted for prediction. These approaches are sequential in operation, although Rvoting can be easily implemented in a distributed environment.

We propose a distributed setting for pasting of small votes, which can also be applicable in scenarios where data is already distributed at sites, and collecting data at one location is a costly procedure. Our approach is essentially "divide and conquer": we divide the training set into n disjoint partitions, and then *Rvote* or *Ivote* on each of the disjoint subsets independently. We call our distributed approaches to pasting Ivotes and Rvotes, *DIvote* and *DRvote* respectively (Chawla et al., 2002a).

The organization of the rest of the paper is as follows. We discuss the related work in Section 2. We present the distributed paradigm of pasting Ivotes, Rvotes, and experimental details in Sections 3 to 7. We ran the distributed learning experiments on a 24-node Beowulf cluster and the ASCI Blue Supercomputer (Livermore National Laboratories), using C4.5 release 8 decision tree (Quinlan, 1992) and Cascade Correlation neural network (Fahlman and Lebiere, 1990) as the base classifiers. We show that DIvote and Ivote give very comparable accuracies, while achieving significantly better classification accuracies than a single classifier in most cases. One major advantage of DIvote is a *significant reduction in training time as compared with Ivote*. Section 7 also includes comparisons to distributed boosting. Section 8 contains the κ plots to highlight the diversity trends of DIvote and DRvote. Section 9 includes empirical evidence of applicability of Ivote with a stable method of learning such as Naive Bayes classifier (Duda et al., 2001; Good, 1965). We present the main conclusions from our work in Section 10.

2. Related Work

One popular approach towards tackling very large training sets is "divide and conquer", which can be set up in a *distributed learning* paradigm. An advantage of this approach is that the partition size of the learning task can be adjusted to fit the available computational resources. One can easily imagine a divide and conquer approach in which a data set is divided across a group of processors and a classifier is learned on each processor concurrently. The classifiers are transmitted to a central processor. The central processor can then process the predictions of the independent classifiers learned. It has been shown that combining classifiers learned on random (smaller) disjoint partitions of data can achieve the classification accuracies of the popular ensemble technique of bagging (Chawla et al., 2002b).

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 weighting) in an approach 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 7 data sets from the UCI repository using disjoint partitions of between 100 and 500 examples, they found that the resulting classification accuracy was generally as good or better than applying RISE to all the data.

Street and Kim (2001) built an ensemble of classifiers from training data 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 accuracy and diversity. This approach allows an ensemble of classifiers to be built from an unlimited amount of training data. It also facilitates the building of an ensemble of classifiers on data with temporal dependencies, where the concept to be modeled may vary over time. In their experiments the ensemble of classifiers was usually slightly less accurate than a single classifier built with as much data as the current ensemble used.

Chawla et al. (2000) studied various partition strategies and found that an intelligent partitioning methodology — clustering — is generally better than simple random partitioning, and generally results in a classification accuracy comparable to learning a single C4.5 tree on the entire data set. They also found that applying bagging to the disjoint partitions, and making an ensemble of many C4.5 decision trees on each partition, can yield better results than building a decision tree by applying C4.5 on the entire data set.

Bagging, boosting, and their variants have been shown to improve classifier accuracy (Freund and Schapire, 1996; Breiman, 1996; Bauer and Kohavi, 1999; Dietterich, 2000; Latinne et al., 2001). According to Breiman, bagging exploits the instability in the classifiers, since perturbing the training set produces different classifiers using the same algorithm. However, creating 30 or more bags of 100% size can be problematic for massive data sets (Chawla et al., 2002b). We observed that for data sets too large to handle practically in the memory of a typical computer, a committee created using disjoint partitions can be expected to outperform 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 (Chawla et al., 2002b).

Latinne et al. (2001) proposed a combination of bagging and random feature subsets: "Bagfs". They generated bootstrap replicates (B) of a given training set, and for each such *bag* they randomly chose features without replacement F times, resulting in F feature subsets. Thus, they had $B \times F$ new learning sets. Using the McNemar test of significance they found that Bagfs never performed worse than bagging and Multiple Feature Subsets (MFS).

Boosting (Freund and Schapire, 1996) also creates an ensemble of classifiers from a single data set by utilizing different training set representations of the same data set, focusing on misclassified cases. Boosting is essentially a sequential procedure applicable to data sets small enough to fit in a computer's memory. Lazarevic and Obradovic (2002) proposed a distributed boosting algorithm to deal with massive data sets or very large distributed homogeneous data sets. In their framework, classifiers are learned on each distributed site, and broadcast to every other site. The ensemble of classifiers constructed at each site is used to compute the hypothesis, $h_{j,t}$, at the jth site at iteration t. In addition, each site also broadcasts a vector with a sum of local weights, reflecting its prediction accuracy. Each site j maintains the local distribution $\Delta_{j,t}$ and local weights $w_{j,t}$. They try to emulate the global distribution D_t by communicating the sums of weights from each site. Each site also maintains a local copy, $D_{j,t}$, of the global distribution D_t, and its local distribution Δ_j, for each boosting iteration t. The training set at site j, for each boosting iteration, is sampled according to $D_{j,t}$. At the end of all boosting iterations, hypotheses $h_{j,t}$ from different sites are combined into a final hypothesis h_{fn}. They achieved the same or slightly better prediction accuracy than standard boosting, and they also observed a reduction in the costs of learning and the memory requirements for their data sets (Lazarevic and Obradovic, 2002). In Section 7, we compare DIvote with distributed boosting.

3. Pasting Votes

Breiman (1999) proposed pasting votes to build many classifiers from small training sets or “bites” of data. He proposed two strategies of pasting votes: Ivote and Rvote. In Ivote, the small training set (bite) of each subsequent classifier relies on the combined hypothesis of the previous classifiers, and the sampling is done with replacement. The sampling probabilities rely on the out-of-bag error, that is, a classifier is only tested on the instances not belonging to its training set. This out-of-bag estimation gives good estimates of the generalization error (Breiman, 1999), and is used to determine the number of iterations in the pasting votes procedure. Ivote is, thus, very similar to boosting, but the “bites” are much smaller in size than the original data set. Thus, Ivote sequentially generates training sets (and thus classifiers) by importance sampling.

Rvote creates many random bites, and is a fast and simple approach. Breiman found that Rvote was not competitive in accuracy to Ivote or Adaboost. The detailed algorithms behind both the approaches are presented in Section 4 as part of DIvote and DRvote.

Using CART, Breiman found that pasting Ivotes gave an accuracy comparable to running Adaboost on the whole data set (Breiman, 1999). Pasting Ivotes does not require the entire data set to be in memory for learning; instances are drawn from the pool of training data to form much smaller training sets. However, sampling from the pool of training data can entail multiple random disk accesses, which could swamp the CPU times. So Breiman proposed an alternate scheme: a sequential pass through the data set. In this scheme, an instance is read and checked to see if it will make the training set for the next classifier in the aggregate. This is repeated in a sequential fashion until all N instances (size of a bite) are accumulated. The terminating condition for the algorithm is a specified number of trees or epochs, where an epoch is one sequential scan through the entire data set. However, the sequential pass through the data set approach led to a degradation in accuracy for a majority of the data sets. Breiman also pointed out that this approach of sequentially reading instances from the disk will not work for highly skewed data sets. Thus, one important component of the power of pasting Ivotes is random sampling with replacement.

The memory requirement to keep track of which instance belongs in which small training set and the number of times the instance was given a particular class is $2JN_B$ bytes, where J is the number of classes and N_B is the number of instances in the entire data set. Let us assume we have a data set of 10^8 records(N_B), 1000 features (equivalent to 4 bytes each), the training set of size 10^5, and $J = 2$ (Breiman, 1999). The memory requirement will be close to a gigabyte, as follows:

$$\begin{aligned}
A &= 2 * J * N_B = 2 * 2 * 10^8 = 0.4 \text{ gigabytes;} \\
B &= \text{1000 features at 4 bytes each for } 10^5 \text{records} \\
&= 1000 * 4 * 10^5 = 0.4 \text{ gigabytes;} \\
C &= \text{memory required by trees stored in memory in megabytes.} \\
\text{Total} &= A + B + C \approx 1 \text{ gigabytes.}
\end{aligned}$$

In our distributed approach to pasting small votes, we divide a data set into T disjoint subsets, and assign each disjoint subset to a different processor. On each of the disjoint partitions, we follow Breiman's approach of pasting small votes. We randomly sample with replacement, as we can load the entire disjoint subset of data in a processor's memory. Thus, the number of disjoint partitions can be dictated by the amount of memory available on each processor. We combine the predictions of all the classifiers by majority vote. Again, using the above framework of memory requirement, if we break up the data set into T disjoint subsets, the memory requirement will decrease by a factor of $1/T$, which is substantial. So, DIvote can be more scalable than Ivote in memory. One can essentially divide a data set into subsets easily managed by the computer's main memory.

4. Pasting DIvotes and DRvotes

The procedure for DIvote is as follows:

1. Divide the data set into T disjoint subsets.

2. Assign each disjoint subset to a unique processor.

3. On each processor build the first bite of size N by sampling with replacement from its subset, and learn a classifier.

4. Compute the out-of-bag error, $e(k)$, and the probability of selection, $c(k)$, as follows (Breiman, 1999):

$$\begin{aligned} e(k) &= p \times e(k-1) + (1-p) \times r(k). \\ p &= 0.75. \text{ We used the same p value as used by Breiman.} \\ k &= \text{number of classifiers in the aggregate or ensemble so far.} \\ r(k) &= \text{error rate of the kth aggregated classifiers on a T disjoint subset.} \\ c(k) &= e(k)/(1-e(k)), \text{ probability of selecting a correctly classified instance.} \end{aligned}$$

5. For the subsequent bites on each of the processors, an instance is drawn at random from the resident subset of data. If this instance is misclassified by a majority vote of the out-of-bag classifiers (those classifiers for which the instance was not in the training set), then it is put in the subsequent bite. Otherwise, put this instance in the bite with a probability of c(k). Repeat until N instances have been selected for the bite (Breiman, 1999).

6. Learn the $(k+1)th$ classifier on the bite created by step 5.

7. Repeat steps 4 to 6, until the out-of-bag error estimate plateaus, or for a given number of iterations, to produce a desired number of classifiers. We ran our experiments for a given number of iterations.

8. After the desired number of classifiers have been learned, combine their predictions on the test data using a voting mechanism. We used simple majority voting.

Pasting DRvotes follows a procedure similar to DIvotes. The only difference is that each bite is a bootstrap replicate of size N. Each instance through all iterations has the same probability of being selected. DRvote is very fast, as the intermediate steps of DIvote — steps 4 and 5 in the above algorithm — are not required. However, DRvote does not provide the accuracies achieved by DIvote. This agrees with Breiman's observations on Rvote and Ivote.

Pasting DIvotes or DRvotes has the advantage of not requiring any communication between the processors, unlike the distributed boosting approach by Lazarevic and Obradovic (2002). Thus, there is no time lost in communication among processors, as trees are built independently on each processor. Furthermore, dividing the data set into smaller disjoint subsets can also mitigate the need for larger memories. Also, if the disjoint subset size is small compared to the main memory on a computer, the entire data set can be loaded in the memory. This will speed up the sampling from the data sets. Lastly, DIvote reduces the data set size on each processor, hence less examples must be tested by the aggregate classifiers during training, which also reduces the computational time.

5. Experimental Setup

We evaluated DIvote and DRvote by experiments on six small to moderate sized data sets, and two large data sets. We performed 10-fold cross-validation for almost all of our data sets, and used two-tailed paired-t-tests at the 95% confidence level to evaluate statistical significance of our results, where applicable. We also provide error bars (accuracy +/- standard error) for the Ivote, DIVote, Rvote, and DRvote in the plots. The y-axis in the plots indicates the accuracy, and the x-axis indicates the number of iterations. Please note that the number of iterations does not necessarily equate to number of classifiers. For the distributed approaches, there are n*$iterations$ classifiers, where n is the number of disjoint partitions. For the sequential approaches, the number of iterations is equal to the number of classifiers in the ensemble.

5.1 Data Sets

The size of these data sets is summarized in Table 1. DNA, satimage, LED, pendigits, letter, waveform and covtype are available from the UCI repository. We used three of the four Statlog (D. Michie and Taylor, 1994) project data sets — DNA, satimage, and letter — used by Breiman (1999). We did not use the shuttle data set, as 10-fold cross-validation with C4.5 already gives accuracies of around 99.5%, and any improvement on it will be miniscule. Moreover, we have previously observed that simple disjoint partitioning is able to achieve that accuracy (Chawla et al., 2000). We used a subset of 60 features (features 61 — 120) for the DNA data set (D. Michie and Taylor, 1994).

We procured the training and testing set partitions of waveform, LED and covtype from Lazarevic and Obradovic (2002) to allow direct comparisons. One of our large data sets comes from the problem of predicting the secondary structure of proteins (Jones, 1999). The "train and test set one" were used in developing and validating, respectively, a neural network that won the CASP-3 (Livermore National Laboratories, 1999) secondary structure prediction contest. The Protein data set (called the "Jones data set" in the rest of the paper) contains 209,529 elements in the training set and 17,731 elements in the testing set. 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 per amino acid to signify where the window spans either the N or C terminus of the protein chain, gives a feature vector of size 315 (Jones, 1999). The testing set is specially constructed to avoid

Data Set	Data Set Size	Classes	Number of Attributes
DNA	3,186	3	60
Satimage	6,435	6	36
LED	Training = 6,000; Testing = 4,000	10	7
Pendigits	10,992	10	16
Letter	20,000	26	16
Waveform	Training = 40,000; Testing = 10,000	3	21
Covtype	Training = 149,982; Testing = 431,030	7	21
Jones	Training = 209,529; Testing = 17,731	3	315

Table 1: Data set sizes, number of classes, and attributes.

any homology to the training set; this makes it a hard prediction problem for the classifier, and an appropriate test for the new protein secondary structure predictions.

We performed 10-fold cross-validation for seven of our data sets: DNA, satimage, pendigits, letter, waveform, LED, and the Jones data set. For the waveform and LED data sets, we combined the training and testing sets to perform 10-fold cross-validation. We combined the training and testing sets to increase the overall data set size for the learning procedures. For the Jones data set the 10-folds were constructed per-chain from the training set, so that non-homogenity is maintained between our 10-fold training and testing sets. This maintained the level of difficulty for the learning algorithm. For the other large data set, covtype, we only ran on the separate training and testing sets of sizes 149,982 and 431,030, respectively, as done by Lazarevic and Obradovic (2002). The size of the entire covtype data set was too large (581,102) to perform a 10-fold cross-validation in a reasonable time, given the number of approaches and classifiers.

5.2 Base Classifiers and Computing Systems

We used the C4.5 release 8 decision tree, the Cascade Correlation neural network, and Naive Bayes classifiers for our experiments. The sequential Rvote and Ivote experiments were run on a 1.4 GHz Pentium 4 Linux workstation with two gigabytes of memory, and an 8-processor Sun-Fire-880 with 32 gigabytes of main memory. We ran most of our DIvote and DRvote experiments on a 24-node Beowulf cluster. Each node on the cluster has a 900 MHz Athlon processor and 512 megabytes of memory. The cluster is connected with 100Bt ethernet. We also ran some of our DIvote experiments on the ASCI Blue Supercomputer. There are 1296 compute nodes on the ASCI Blue. The 4-CPU nodes have: 1.5 gigabyte memory, 332 MHz PowerPC 604e chip, 83 Mhz memory bus, and a compute node peak performance of 2.6 GFLOP/s (Livermore National Laboratories).

6. Experiments with the C4.5 Decision Tree Learning Algorithm

We ran 10-fold cross-validation experiments using C4.5 decision trees for all the small and moderate sized data sets. Section 6.1 contains the results on the small and moderate data sets. For the large data sets, we used separate training and testing splits. However, for one of the large data sets–Jones–we also report the 10-fold cross-validation result. Section 6.2 contains the results on the large data sets. We also present a timing analysis and comparison between DIvote and Ivote in Section 6.3.

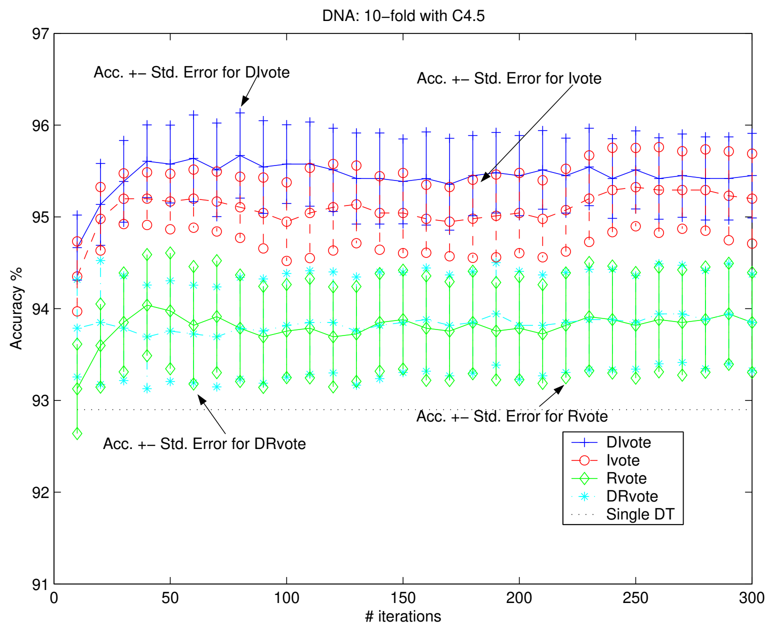


Figure 1: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for DNA.

6.1 Small and Moderate Sized Data Sets

We partitioned our small and moderate sized data sets, with the exception of DNA, into four disjoint partitions and used bites of size 800 to learn classifiers. Due to the smaller size of DNA, we partitioned it into three disjoint partitions and used bites of size 400. We used unpruned trees for all decision tree experiments as the ensemble of classifiers will override the overfitting (Breiman, 1999). Figures 1 to 5 show the 10-fold accuracy trends for six of our data sets. Each of the distributed approaches is comparable in its classification accuracy to its sequential counterpart. DIvote/Ivote achieve significantly higher classification accuracies than a single C4.5 classifier, and the simplistic approaches of DRvote/Rvote. LED is an exception, as all the approaches are comparable to each other. We believe that DIVote/IVote could be a victim of the feature noise present in the LED data set, as each feature has a 10% probability of having its value inverted (Blake and Merz, 1998).

The letter data set differed from other data sets as both DRvote and Rvote achieved significantly worse classification accuracies than C4.5. However, for letter also, DIvote and Ivote were comparable, and significantly better than Rvote, DRvote, and C4.5, in their classification accuracies. The letter data set has 26 classes with 16 dimensions; each 800 sized bite will contain an average of 30 instances (on an average) for each class. Thus, the 30 training instances may not mimic the real distribution of the data set. To test this model, we created 100% bags on each disjoint partition (Chawla et al., 2000) of the letter data set, and found that the classification accuracy increased significantly as compared to DRvote or Rvote, but was still not better than DIvote and Ivote. This shows that 100% random bags constructed on each disjoint partition of the letter data set are introducing more coverage, better individual classifiers, and diversity[1] compared to the DRvote bites (800 instances).

1. We visit the diversity issue in Section 8.

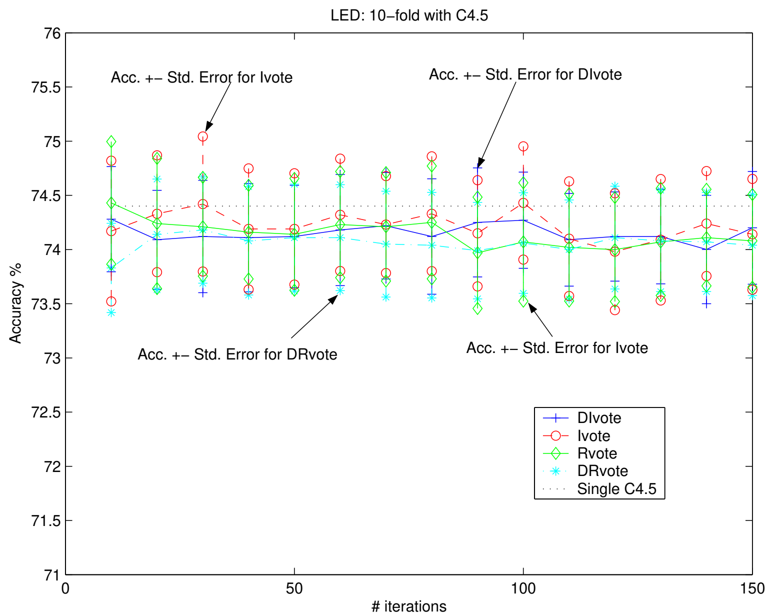


Figure 2: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for LED.

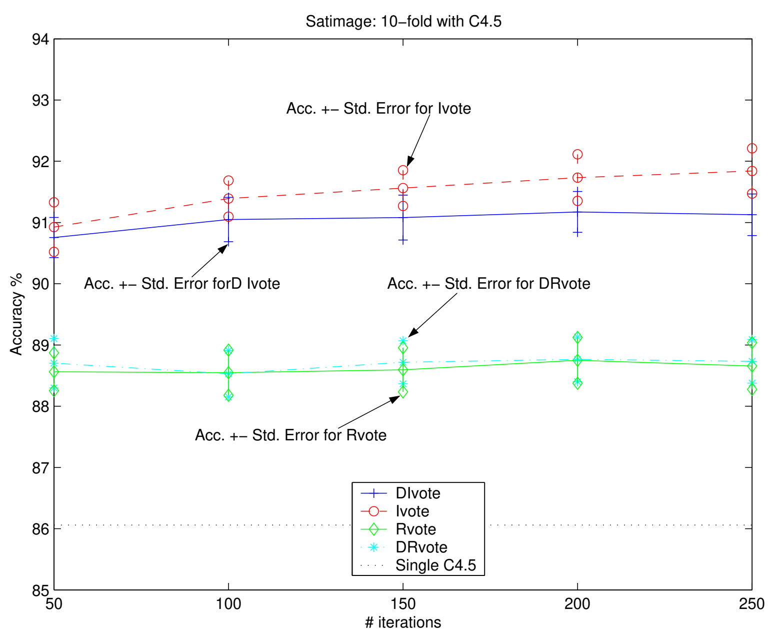


Figure 3: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for satimage.

Since DIvote and Ivote sample heavily from misclassified instances, after each iteration or series of iterations they focus on potentially different instances.

The results on the small data sets are encouraging as for almost all the data sets, DIvote is similar to Ivote in its classification accuracy, while surpassing a single C4.5 decision tree learned on

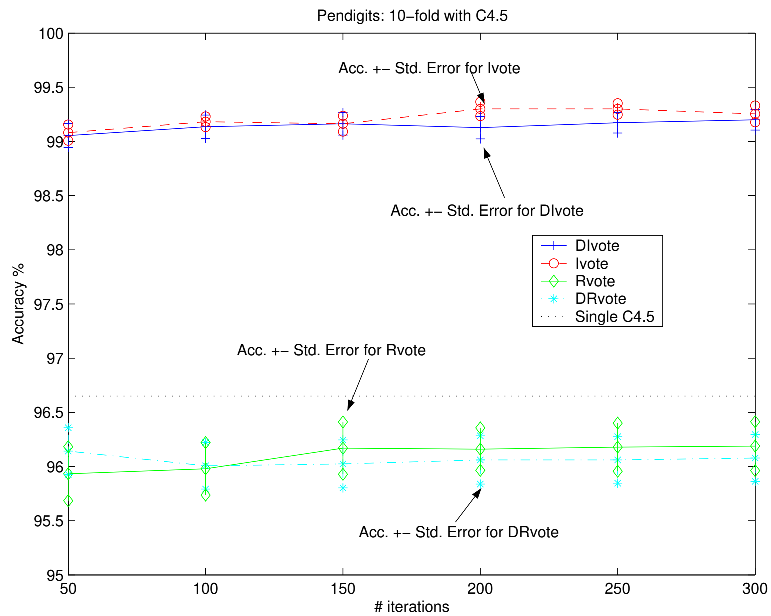


Figure 4: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for pendigits.

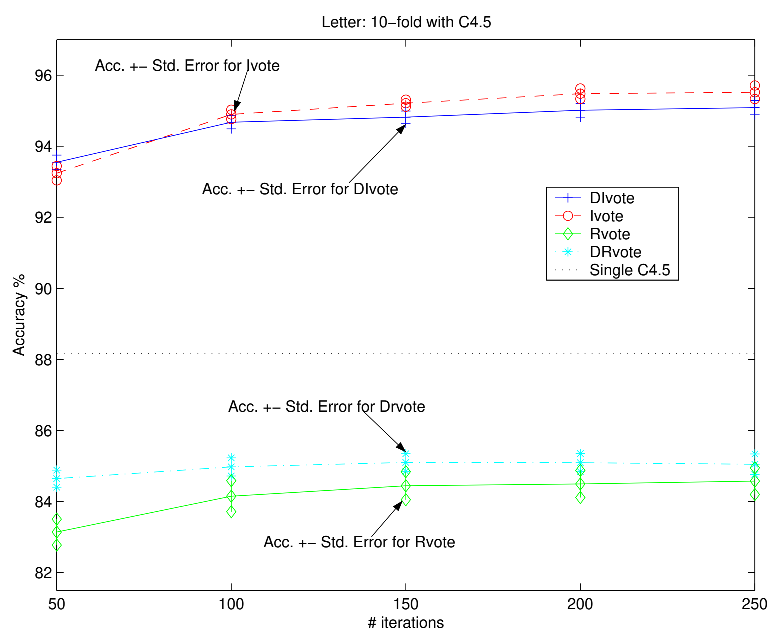


Figure 5: C4.5 Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for letter.

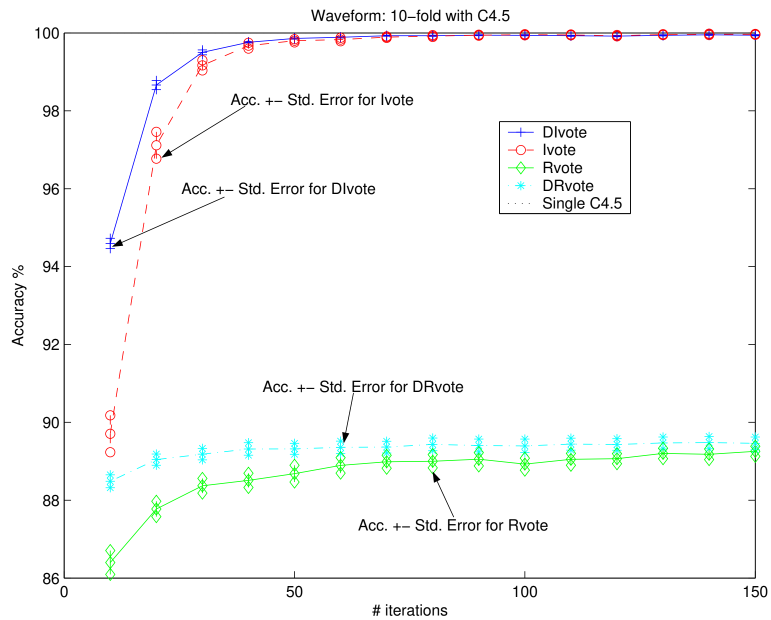


Figure 6: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for waveform.

the entire data set. DIvote is also significantly better than DRvote. Another important observation is that for almost all the data sets, 100 iterations of DIvote or Ivote achieve close-to-maximum accuracies, and after that the increase in accuracy is very slow compared to the addition of iterations. Given the small bite size, 100 iterations are very fast using any algorithm as the training set size is only 800 (or 400).

6.2 Large Data Sets

Due to the distinct nature of analysis on both the large data sets, we treat them separately in the subsequent subsections.

6.2.1 Jones Data Set

We set up two different experiments on the Jones data set. One evaluates the DIvote, Ivote, DRvote, and Rvote approaches by learning and testing on the provided train and test sets. The other, 10-fold cross-validation experiment, was to understand and report statistical differences between DIvote and Ivote.

We divided the training set into 24 random disjoint partitions, as we had 24 nodes of the Beowulf cluster available for this experiment. To understand the effect of bite size on DIvote for this data set, we also set up bites at sizes 1/256 (818 instances), 1/128 (1636 instances) and 1/64 (3272 instances) of the entire training set size of 209,529. Figure 7 shows the classification accuracies of the various approaches using the given train/test split. As is evident from the figures, all the distributed approaches more accurate than the sequential approaches.

We also observe that increasing the bite sizes has no major effect on the classification accuracies of the ensemble. The experiment with increasing bite sizes is computationally cheap, as the bite size

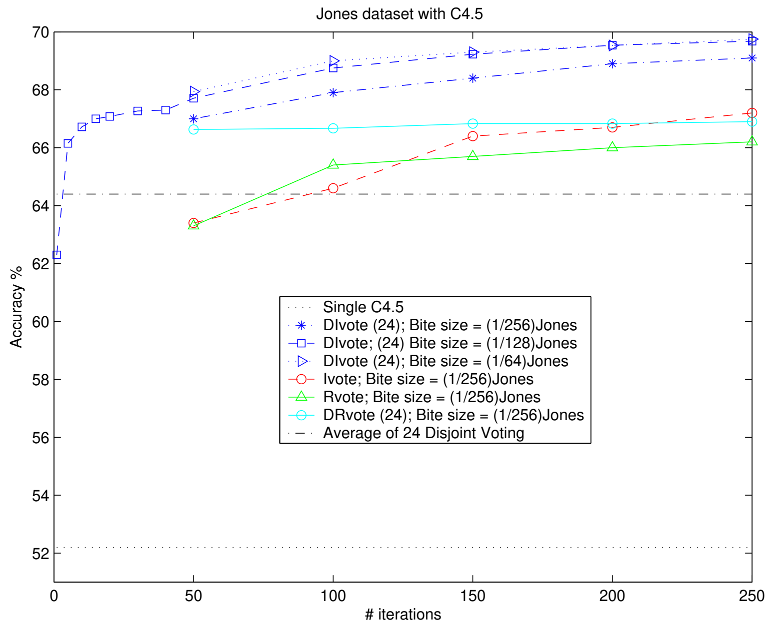


Figure 7: Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for the Jones data set.

remains very small (from approximately 800 to 3200). In addition, we also show the classification accuracy achieved by voting classifiers learned on each of the random disjoint partitions. Note that adding a meta-layer of DIvote on the disjoint partitions provides an absolute improvement of at least 4% in accuracy. We would also like to point out the accuracy achieved by learning a single C4.5 decision tree on the entire data set: 52.2%. It is remarkable that all the ensemble approaches, reported in the Figure 7, provide an absolute gain of at least 12% on this data set. The non-homologous nature of the testing set makes it a particularly hard problem for C4.5 with its axis parallel splits (Chawla et al., 2001). The decision tree overfits the training set, achieving a very high resubstitution accuracy of 96.7%. The ensemble of decision trees counters the effect of overfitting and improves generalization, which is very important for the protein domain (Eschrich et al., 2002).

It is also interesting to note that the average accuracy of a decision tree learned on bites of size 1/256th of the Jones data set (for DIvote) is below 50%, and the aggregate of all the not-so-good classifiers gives a performance in the range of 66% to 70% for 50 or more learning iterations. A classifier, learned on 1/256th of the entire training set, with an accuracy of $< 50\%$ is not very surprising when juxtaposed with the accuracy of the single classifier learned on the entire training set. Figure 8 shows the accuracy trends for 4800 decision trees learned in the DIvoted ensemble for the Jones data set. These plotted accuracies are sorted in an increasing order, and are not in the same order as the construction of bites.

To further evaluate the behavior of DIvote and Ivote for the large Jones data set, we report the 10-fold averaged cross-validation results in Figures 9 and 10. As with the previous experiment, we divided each training fold into 24 random disjoint partitions. Let us examine Figure 9 first. DIvote achieves significantly higher classification accuracies than Ivote for 200 iterations. However, a point of contention could be that DIvote is constructing 200*24 (4800) classifiers versus 200 classifiers for Ivote, albeit in less time (which is an advantage in itself of DIvote). So, on the ASCI Blue

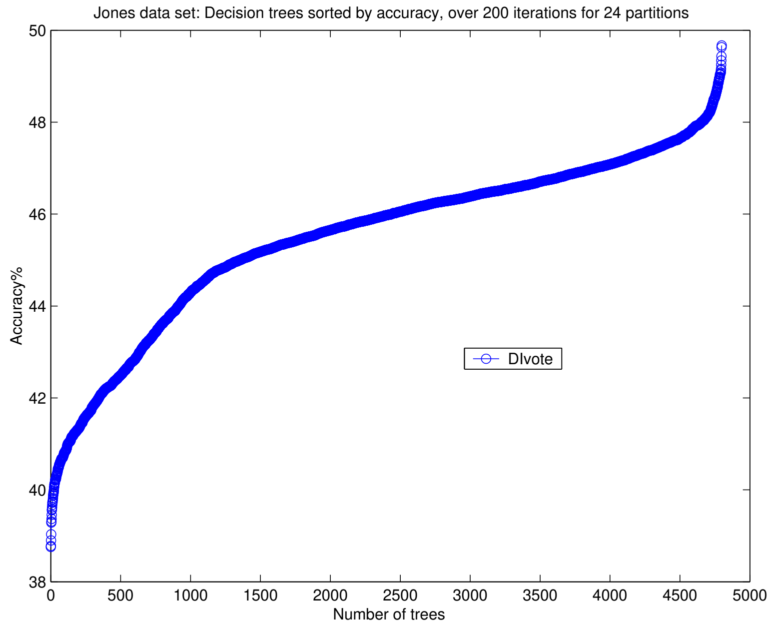


Figure 8: Individual performances on the Jones data set.

supercomputer, we let Ivote run up to 1200 iterations or six hours (per fold) of allowed compute time on a node. We then ran 50 (1200/24) iterations of DIvote, taking only six minutes (per fold), which means that we constructed only 50 decision tree classifiers on each of the 24 disjoint partitions. This gave us an ensemble size of 1200 (50*24), which is equivalent to Ivote. On performing a two-tailed paired-t-test at 95% confidence interval, we find that an ensemble of 1200 classifiers of Ivote is significantly better (in classification accuracy) than the ensemble of 1200 classifiers of DIvote.

Now, the ensemble of DIvote classifiers is really under-constructed this time, and we can still use the hours of compute time utilized by Ivote to construct the ensemble. For instance, consider Figure 10, if we compare 200 DIvote classifiers that took 30 minutes and 1200 (maximal possible, given the computation bottleneck) Ivote classifiers that took six hours, we observe that DIvote and Ivote are comparable to each other. This highlights the significant advantage of DIvote in terms of time spent on computation.

6.2.2 COVTYPE

Our second large data set, covtype, contains 149,982 instances in the training set. This data set also has a very high skew in its class distribution as reported in Table 2. We divided the training set into eight disjoint partitions, as done in the distributed boosting paper. We ran DIvote, Ivote, Rvote, and DRvote experiments using a bite size of 800. Figure 11 shows one of the results on this data set. Each of the ensemble approaches for covtype results in significantly worse classification accuracy than learning a single C4.5 decision tree on the entire training set. This is contrary to our previous observations. We believe the high skew in the class distribution is coming into play, as each of the classes is, possibly, not getting a sufficient representative coverage and interaction in the bites.

We further investigated the coverage issue by increasing the bite size for DIvote and Ivote. As one would do in subsampling (Provost et al., 1999), we tried the following different bite sizes: 800,

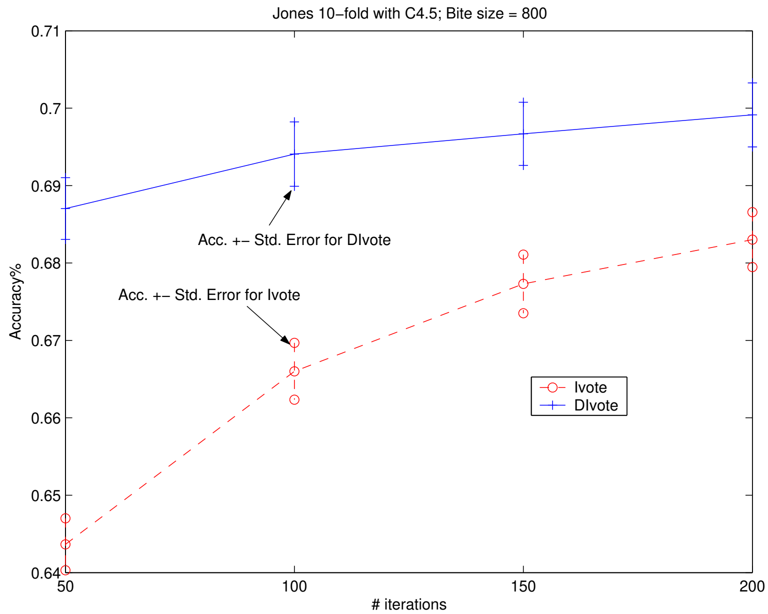


Figure 9: 10-fold averages of up to 200 iterations of Ivote and DIvote.

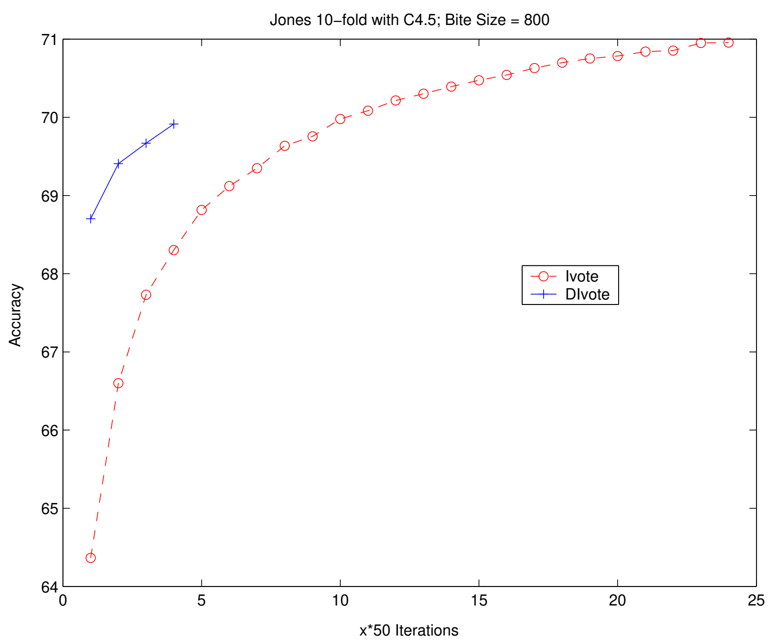


Figure 10: 10-fold averages of up to 1200 iterations of Ivote and 200 iterations of DIvote.

Class 3	670
Class 4	2,414
Class 5	4,544
Class 6	5,433
Class 2	8,971
Class 0	54,433
Class 1	73,517

Table 2: Class distribution for the covtype data set, in ascending order.

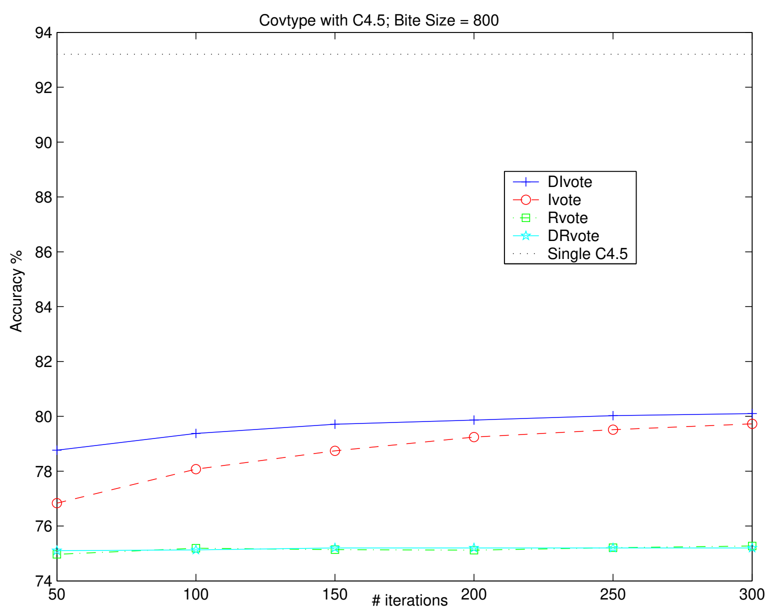


Figure 11: Covtype data set with C4.5.

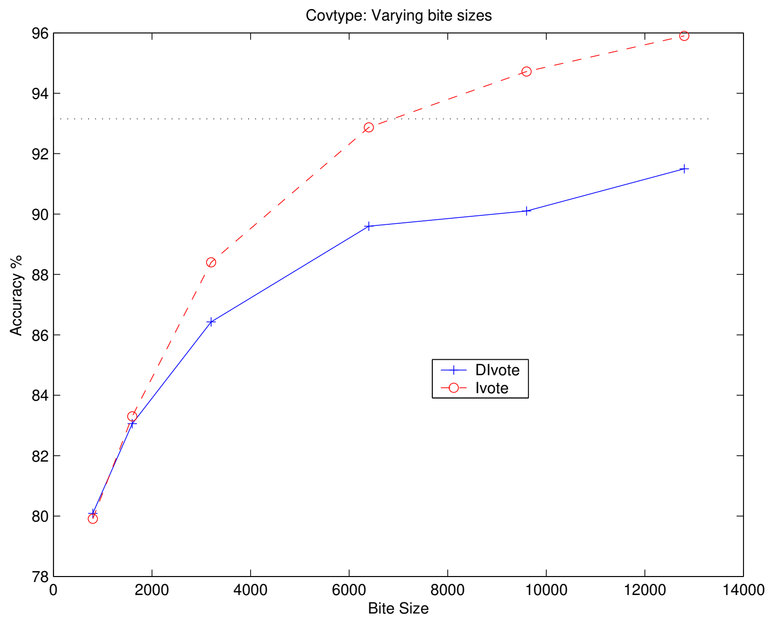


Figure 12: Effect of varying bite sizes on the covtype data set for Ivote and DIvote.

1600, 3200, 6400, 9600, 12800. We would like to point out that the bite size of 12,800 is still less than 10% of the training set of size 149,982. In the sequential domain, when searching for a single appropriate subsample of a given data set, the error (or accuracy) curve at varying subsample sizes is constructed, where each (x, y) on the curve signifies the error y at a particular subsample size x. So in the ensemble setting, each (x, y) represents the voted accuracy, y, of the ensemble constructed from subsamples (bites) of size x. This allowed us to make an accuracy (or error) curve for the improvement observed with varying bite sizes. Figure 12 summarizes the result. We constructed 300 classifiers at each bite size. Increasing the bite size improves the coverage for the classes, and leads to a significant improvement in the prediction accuracy for both DIvote and Ivote. However, Ivote achieves higher classification accuracies than DIvote for this data set, and also surpasses the accuracy achieved by the single C4.5 decision tree — 93.2%. The discrepancy in Ivote and DIvote performance is possibly arising from the limited coverage of classes in each bite from the 8 disjoint partitions; for example, each disjoint partition has, on an average, only 83 examples of class 3.

6.3 Timing

Table 3 shows the timing (user and system time during training) ratios of DIvote to Ivote, and DRvote to Rvote on the Beowulf cluster for some of our data sets using C4.5 decision trees. We report times for some of our small and large data sets, and with only C4.5, as the rest should follow similar trends. The experimental parameters were: number of iterations = 100; bite size $N = 800$ for the small data sets, and bite size N = (1/256) * (Jones data set size) for the Jones data set.

The time taken for DIvote and DRvote reflects the average of the time taken for 100 iterations on each of the T ($T = 4$ for the small data sets; $T = 24$ for the large data set) nodes of the cluster. For fair timing comparisons to DRvote and DIvote, we also ran 100 iterations of Ivote and Rvote on a single cluster node. It is noteworthy that we are able to build T*100 DIvote classifiers simultane-

Data Set	DIvote/Ivote	DRvote/Rvote
Satimage	0.36	0.91
Pendigits	0.29	0.95
Letter	0.23	0.89
Jones	0.048	0.90

Table 3: Ratio of time taken by DIvote to Ivote, and DRvote to Rvote, on a cluster node.

ously. One significant advantage of the proposed DIvote approach is that it requires less time than Ivote, particularly for very large data sets. Since we divide the original training set into T disjoint subsets, during training the aggregate DIvote classifiers on a processor test many fewer instances than aggregate Ivote classifiers (for the Jones data set each disjoint partition has only 8730 instances, compared to 209,529 in the entire data set). Also, a reduction in the training set size implies that the data set can be more easily handled in main memory. This is a big gain as the multiple random disk accesses can then be avoided during sampling. We pointed out earlier that Breiman found that sequential disk access was not giving as good a classification accuracy as multiple random disk access. So, to maintain the advantage of random sampling, each processor can load the relatively smaller training set partition into its main memory. Table 3 shows that as the overall training data set size increases, the ratio of DIvote time to Ivote time decreases. That is, Ivote becomes more computationally expensive than DIvote as the data set increases.

It is not surprising that the timings for DRvote and Rvote are very similar, as both the approaches essentially build many small bags from a given training set, without the intermediate book-keeping. Nevertheless, DRvote builds T times as many Rvote classifiers in less time.

We would like to note that the same software performs DIvote and Ivote, except that for the distributed runs we wrote an MPI program to load our pasting votes software on different nodes of the cluster, and collect results. So, any further improvement of the software would be applicable across the board, although it might decrease the ratios of time taken by DIvote to time taken by Ivote. Nonetheless, it is fair to assume that Ivote will hit a memory bottleneck (with an increasing data set size) faster than DIvote irrespective of the implementation.

7. Experiments with the Cascade Correlation Neural Network

To investigate whether our results generalize to other sorts of classifiers, we performed 10-fold cross-validation experiments on the small to moderate sized data sets using the Cascade Correlation (CC) neural network. We also compared the DIvote approach with the distributed boosting algorithm, which is presented in Section 7.2.

7.1 10-fold Cross-Validation Experiments

Figures 13 to 18 summarize the results of 10-fold cross-validation with CC. Using two-tailed paired-t-tests at the 95% confidence interval, we find that DIvote and Ivote are significantly better than DRvote, Rvote for satimage, letter, waveform, and pendigits, and comparable for DNA. DIvote and IVote are significantly better than a single CC for DNA, satimage, letter, waveform, and pendigits. DIvote and Ivote achieve comparable accuracies (no significant difference) for all the data sets,

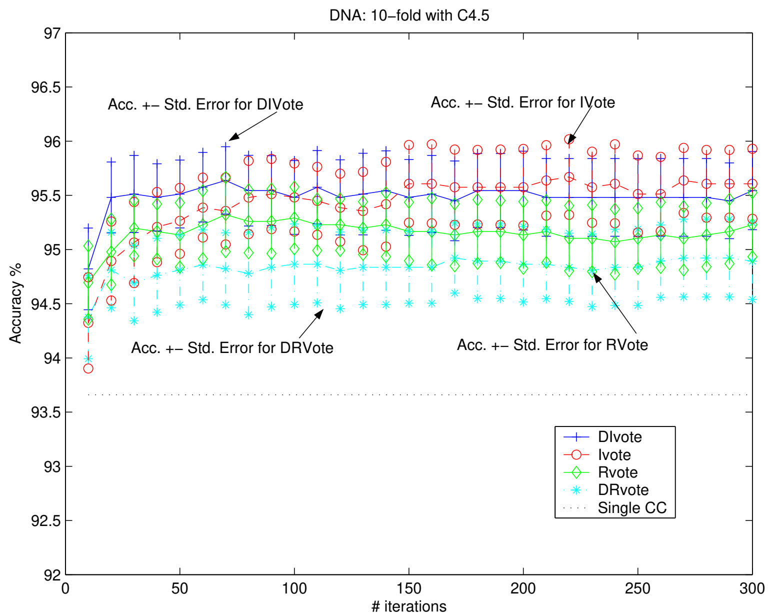


Figure 13: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for DNA.

but for pendigits. DIvote is significantly better than Ivote for pendigits. LED, again, provides an exception: all approaches are similar to each other in their classification accuracies.

We required fewer learning iterations with CC than with C4.5. Moreover, we observe that with CC, 40 to 50 iterations are sufficient, and the accuracy increases very slowly, if at all, thereafter. This is noteworthy as although neural networks are slower to learn when compared to decision trees, they require even fewer learning iterations. Essentially, an important conclusion to be drawn from all these experiments is that using importance sampling not only mitigates the need for many iterations but also limits the amount of data needed for learning.

7.2 Comparison to Distributed Boosting

We ran experiments on the four publicly available data sets used by Lazarevic and Obradovic (2002) to facilitate direct comparisons. We divided the training sets of covtype, LED and waveform into four disjoint partitions and the training set of pendigits into 6 disjoint partitions as done by Lazarevic and Obradovic (2002). We learned an ensemble of CC classifiers by running 100 iterations of DIvote with a bite size of 800 for each of the data sets. We evaluate the ensemble on the separate testing sets to compare with the distributed boosting approach.

Table 4 reports the accuracies achieved by DIvote and distributed boosting on the separate testing sets. We report the distributed boosting accuracy for the "Simple Majority" voting scheme using $p = 0$ directly from (Lazarevic and Obradovic, 2002). The table shows that using neural networks we can achieve accuracies comparable to (or better than) the distributed boosting algorithm. The inherent and compelling advantage of the DIvote approach is that it requires no inter-processor communication during learning. Although with DIvote we are learning hundreds of classifiers, we never need to learn a single classifier on a complete given partition of data. The bite size always remains

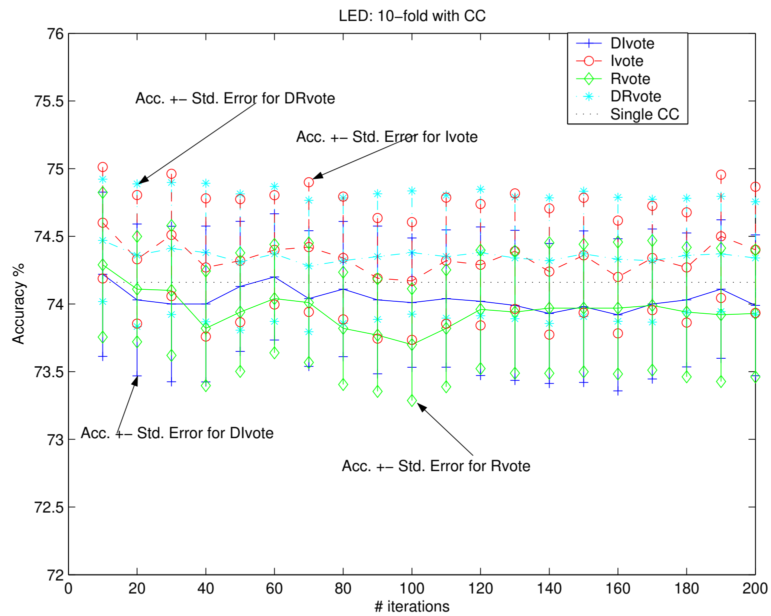


Figure 14: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for LED.

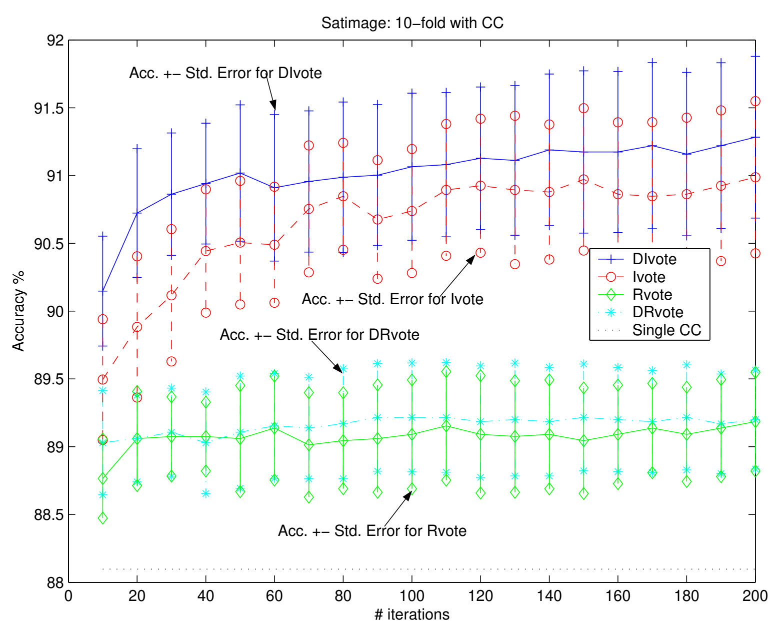


Figure 15: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for satimage.

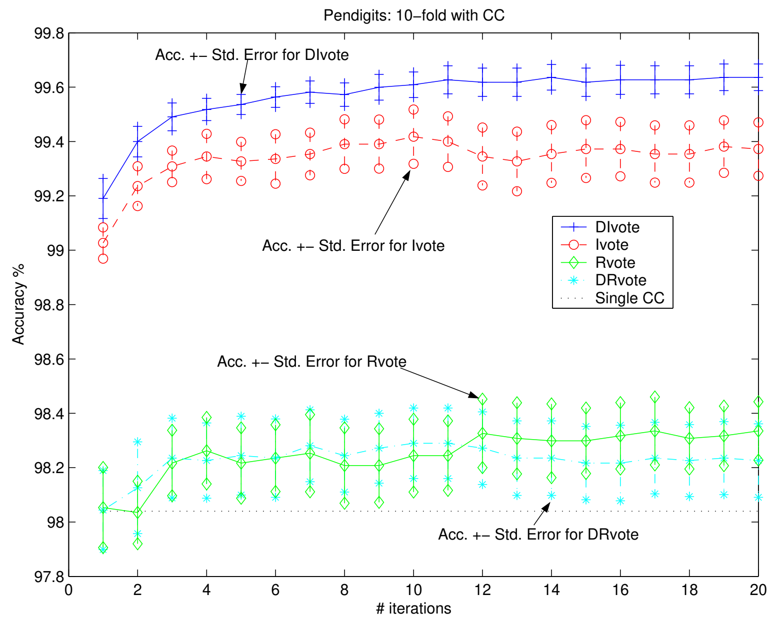


Figure 16: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for pendigits.

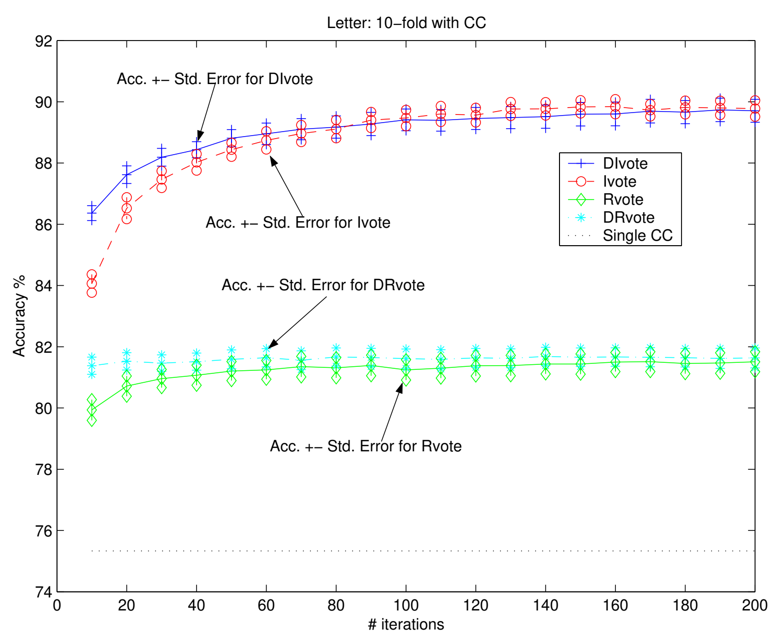


Figure 17: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for letter.

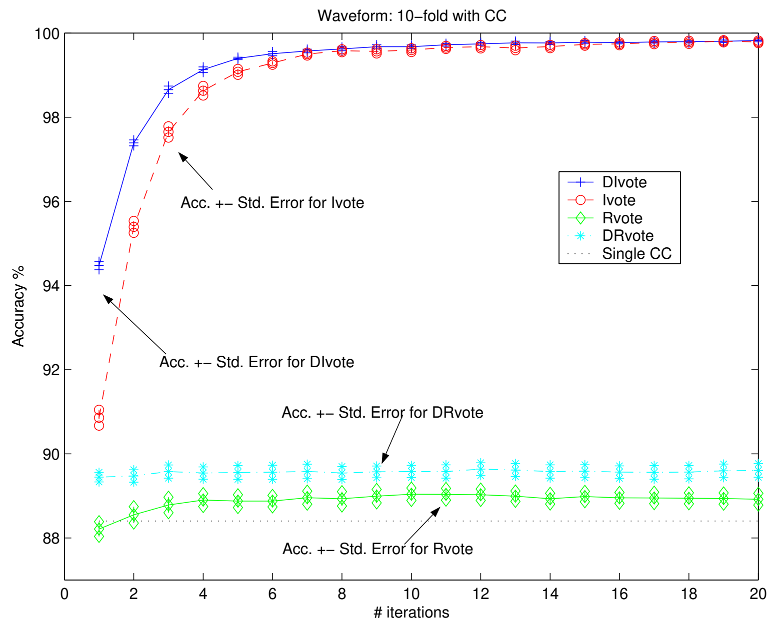


Figure 18: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for waveform.

Data Set	DIvote	Distributed Boosting (Lazarevic and Obradovic, 2002)
Pendigits	96.4%	96.5%
LED	74.0%	73.4%
Waveform	87.2%	87.0%
Covtype	78.2%	72.6%

Table 4: Classification accuracy comparisons for DIvote and distributed boosting.

very small compared to the size of the entire training set or even the size of the disjoint partition. Thus, given any partition size, the learning task for a single classifier always remains small.

8. Diversity of Classifiers in the Ensemble

The diversity of classifiers in an ensemble is considered a key issue in the design of an ensemble (Kuncheva and Whitaker, 2003). Classifiers are considered diverse if they disagree on the examples for which they err. Diversity, thus, is a property of a group of classifiers with respect to a data set. The classifiers might be reporting similar accuracies, but be disagreeing on their errors. Diversity is an important aspect of the ensemble techniques — bagging, boosting, and randomization (Dietterich, 2000). There has been a significant amount of research on defining measures for diversity, and evaluating the reasons behind the success of ensemble techniques (Giacinto and Roli, 2001; Kuncheva and Whitaker, 2003; Kuncheva et al., 2000; Skalak, 1996; Ho, 1998; Banfield et al., 2003). Diversity is an important metric in addition to accuracy when evaluating and constructing an ensemble of classifiers. Breiman notes that the success of random forests lies in the interplay of the "strength[2]" and "correlation[3]" of classifiers. Ideally, one wants lower correlation and higher strength of classifiers Breiman (2001). To understand the behavior of our ensemble of classifiers, we implemented the κ metric given by Dietterich (2000), and defined by

$$\begin{aligned}
\Theta_1 &= \frac{\sum_{i=1}^{T} C_{ii}}{m}, \\
\Theta_2 &= \sum_{i=1}^{T} \left(\sum_{j=1}^{T} \frac{C_{ij}}{m} \cdot \sum_{j=1}^{T} \frac{C_{ji}}{m} \right), \\
\kappa &= \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}.
\end{aligned}$$

T is the number of classes, C is the $T * T$ square array, such that C_{ij} signifies the number of examples assigned to class i by the first classifier and to class j by the second classifier. Θ_1 gives the degree of agreement, and Θ_2 is the degree of agreement expected at random. κ is the statistic measuring diversity. κ equals zero when the two classifiers agree only by chance and κ equals one when the two classifiers agree for every example. Dietterich (2000) produced a scatter plot of κ and mean error of a pair of classifiers to show the spread of κ values against error. The κ plots are constructed by plotting the κ value against the mean error (or accuracy) of the n random pairs of decision tree classifiers. Each classifier in an ensemble (or a random subset of the ensemble) is combined with every other classifier to compute the mean pair-wise error and κ value. The lower the κ values, the higher the disagreement amongst the classifiers.

We built κ plots for C4.5 decision tree ensembles constructed by DIvote and DRvote. We wished to shed some light on the classification accuracy improvements observed with DIvote by virtue of more diversity in the design of the ensemble of classifiers. We only show the κ plots for two of the data sets: letter and Jones. Figure 19 shows the κ plots for C4.5 decision trees constructed with DIvote and DRvote for the letter data set. We get a broader range of κ values and mean error with DIvote, compared to DRvote, thus implying that DIvoted classifiers are essentially more diverse with each other than the DRvoted classifiers. This ties in with our observation that the ensemble constructed via DIvote is more accurate than the ensemble constructed via DRvote.

Figure 20(a) shows the diversity plots for DIvote on the Jones data set. We randomly chose 600 decision trees from the ensemble of 4800 decision trees for constructing the κ plots; the bite size was 1636 or 1/128th of the Jones data. We chose only 600 trees due to the computational infeasibility of doing pairwise comparisons for all 4800 decision trees. As is evident from the figures, the DIvoted classifiers produce a lower κ value, that is, there is a higher disagreement in the classifiers' predictions. Thus, DIvote is able to boost the classification accuracy of a multiple classifier system of weak and unstable classifiers, by inducing a high diversity amongst the classifiers. Figure 20(b) shows the diversity plot for DRvote on the Jones data set. We again selected 600 trees at random from our ensemble of 4800 decision trees constructed from bites of size 1636. As shown in the Figure 20(b), there is a high degree of disagreement amongst the classifiers, but there is still a more compact set of points in the plot as compared to the DIvoted scatter plot. There is a broader range in error as well as κ values for the DIvote classifiers. We observed that the ensemble constructed

2. Strength is the accuracy of the individual classifiers.
3. Correlation is the dependence amongst classifiers.

Data Set	Single Naive Bayes Classifier	300 Ivote	300 Rvote	50 Bags
Letter	65.58%	**68.82**%	67.02%	66.1%
Pendigits	81.1%	**91.29**%	81.36%	81.04%

Table 5: Naive Bayes classifier on the letter and pendigits data sets. The highest accuracy is given in bold.

with DRvote classifiers is able to provide an accuracy boost of at least 14% over learning a single decision tree classifier (see Figure 7). Thus, the high diversity amongst the classifiers aids the voting ensemble.

9. Stable Classifier and Ivote/DIvote

So far, we have shown that DIvoting/Ivoting unstable classifiers, such as decision trees and neural networks, achieves better classification accuracies than learning a single unstable classifier on the entire data set. Ensemble methods have been usually cited to work better with unstable classifiers due to the sensitivity of the unstable classifiers to the data presented for learning. We believe that the core issue in generating a diverse set of classifiers is how you generate the ensemble, not the base classification method. For instance, boosting has been shown to work well with a stable classifier like Naive Bayes (Bauer and Kohavi, 1999).

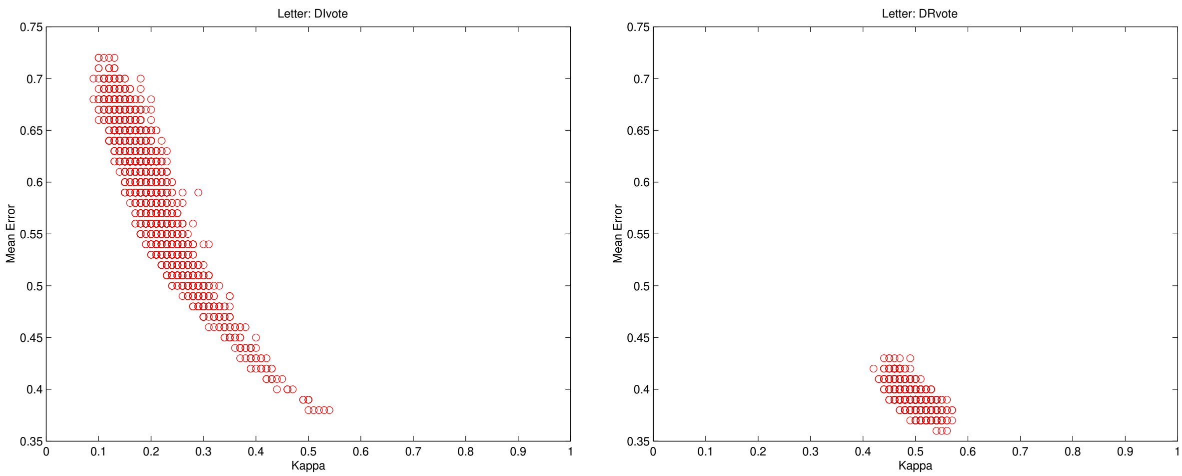
We saw in the diversity plots in the previous section, that DRvote and Rvote produce a compact set of κ values in spite of being constructed from unstable classifiers. And DIvote and Ivote produced a bigger spread of κ values. Hence, it is the importance sampling in DIvote or Ivote, which is helping the learned ensemble. To show this we used a stable method of learning a classifier, Naive Bayes[4], in conjunction with Ivote, Rvote, and bagging. For comparison purposes, we evaluated the sequential versions — Ivote, Rvote and bagging — with Naive Bayes on two of our data sets: pendigits and letter. We used the separate training and testing sets, since we were more interested in comparing the diversity trend between importance sampling and random sampling methodologies than in statistical validation of accuracies' improvement.

Table 5 shows the classification accuracies obtained from learning Naive Bayes classifiers on the letter and pendigits data sets. Figures 21 to 22 highlight the κ results on both the pendigits and letter data sets. As is evident from the figures, Ivoted classifiers are more diverse (and accurate) than both Rvoted and bagged classifiers. This result is not very surprising, as Ivote/DIvote classifiers are learned on bites derived from importance sampling, and each of these bites can look very different due to the heavier sampling from more difficult examples. Although a spread of κ values is observed for Rvote with the pendigits data set, this spread is prevalent at $0.7 \leq \kappa \leq 1.0$. Higher κ values indicate a higher degree of agreement between the different pairs of classifiers.

10. Conclusion

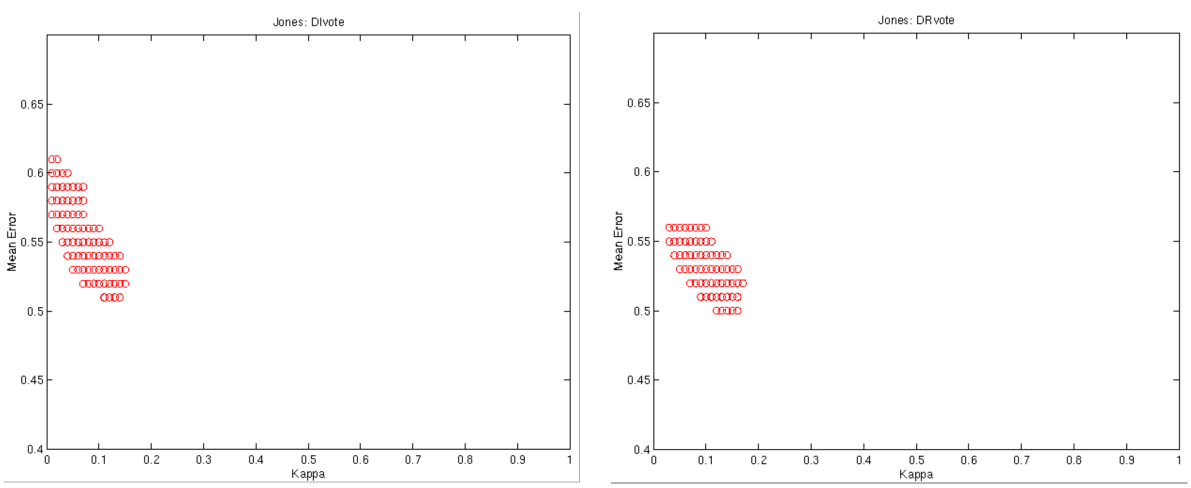
We proposed a distributed framework for pasting Ivotes by dividing a training set into n disjoint partitions and building hundreds of classifiers (using very small training sets) on each disjoint partition. The main conclusion of our work is that pasting DIvotes is a promising approach for very

4. The source code was downloaded from http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html.



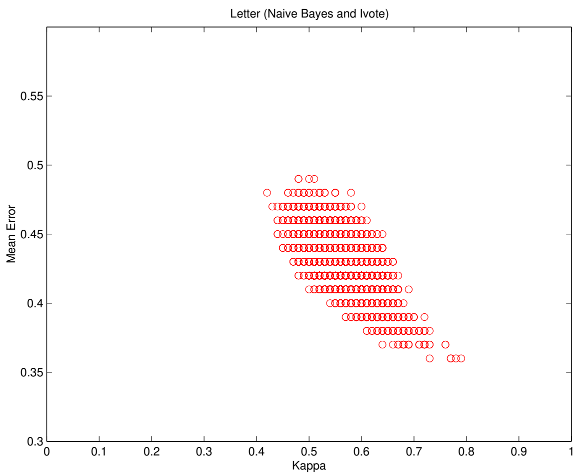
(a) (b)

Figure 19: a) κ Plot for DIvote with C4.5 on the letter data set. b) κ Plot for DRvote with C4.5 on the letter data set.

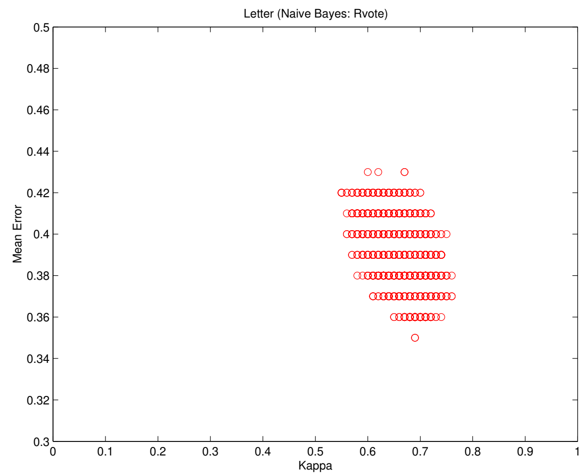


(a) (b)

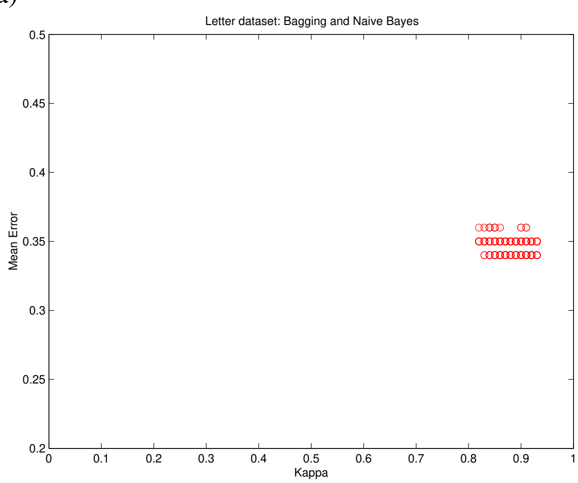
Figure 20: a) κ plot for DIvote and Jones data set. b) κ plot for DRvote and Jones data set.



(a)

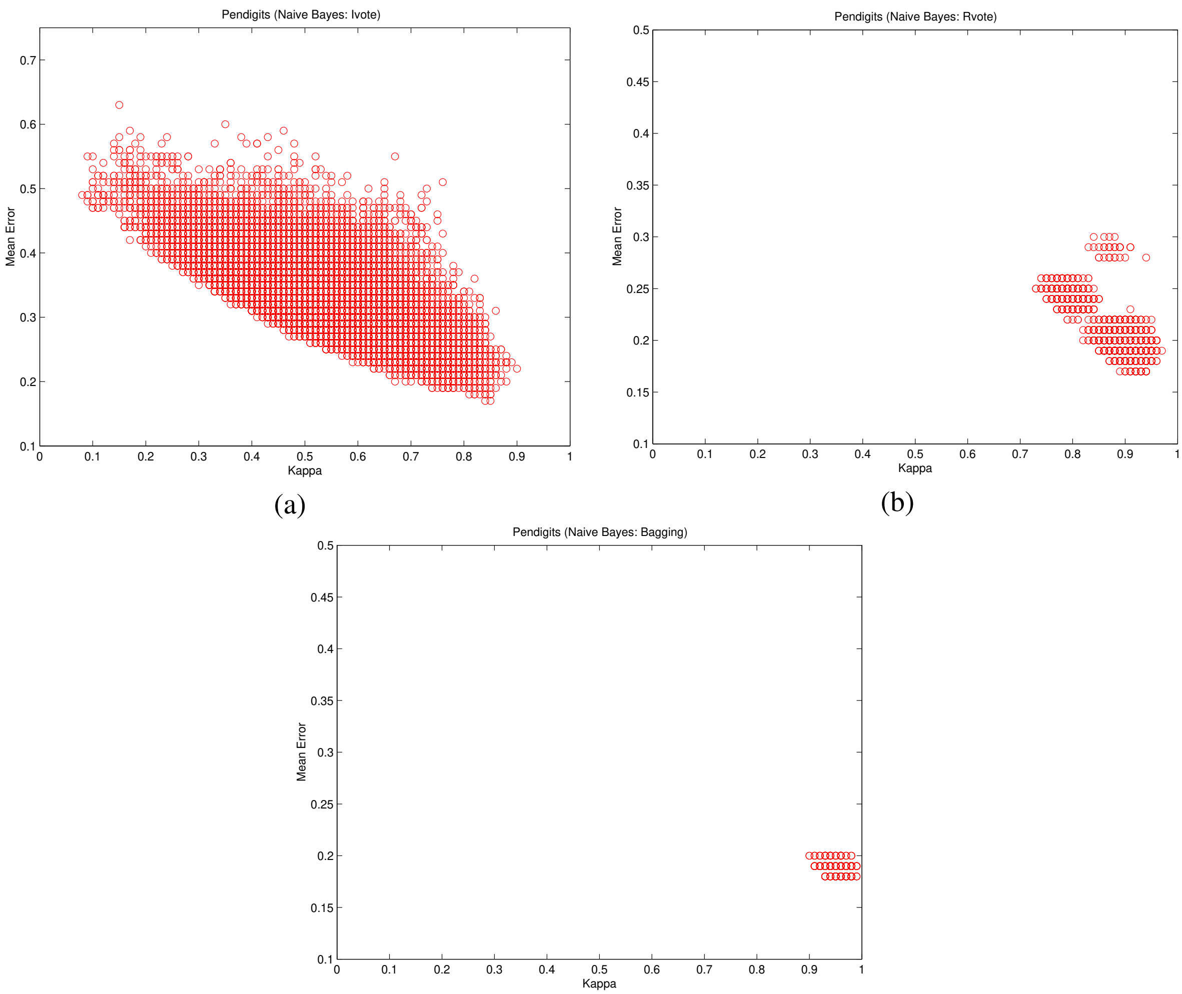


(b)



(c)

Figure 21: a) κ plot for Naive Bayes and Ivote on the letter data set. b) κ plot for Naive Bayes and Rvote on the letter data set. c) κ plot for Naive Bayes and bagging on the letter data set.



(a) (b) (c)

Figure 22: a) κ plot for Naive Bayes and Ivote on the pendigits data set. b) κ plot for Naive Bayes and Rvote on the pendigits data set. c) κ plot for Naive Bayes and bagging on the pendigits data set.

large data sets. Data sets too large to be handled practically in the memory of a typical computer are appropriately handled by simple partitioning into disjoint subsets, and adding another level of learning by pasting DIvotes or DRvotes on each of the disjoint subsets. We show that for almost all the data sets, using decision trees and neural networks, DIvote achieves classification accuracy comparable to Ivote (sometimes better), and almost always better than learning a single classifier. We evaluated DIvote and Ivote on data sets coming from various domains, and for almost all the cases they achieve significantly better classification accuracies than a single classifier.

DIvote is scalable as it never requires the learning set size to be greater than a very small proportion of the entire training set. Each processor works independently, without requiring communication at any stage of learning. The end result is an ensemble of hundreds of DIvote classifiers. We also conclude that pasting DIvotes is more accurate than pasting DRvotes. We believe that the combined effects of diversity, good coverage, and importance sampling are helping DIvote and Ivote. It is significant that DIvote does much better than the simplistic version of pasting small votes in a distributed scenario. Moreover, we observe that with DIvote and Ivote, the accuracy grows fast initially during learning, and then slowly plateaus. Particularly, with neural networks fewer iterations of DIvoting are needed to vastly improve over learning a single classifier from the entire training set. Neural networks, particularly, can get slowed down significantly in the training phase when learning from very large data sets. DIvote is a promising approach to reduce the learning time with neural networks even on very large data sets.

Using κ (diversity) plots, we support the theory that given an ensemble of diverse classifiers, an improvement in the accuracy can be observed. We note that DIvote provides the most significant improvement of almost 18% (relative improvement of 34%) in the difficult, non-homogeneous Protein Prediction problem over learning a single classifier using C4.5. Thus, the ensemble of diverse DIvote classifiers counters the effect of overfitting and improves the generalization.

Another important contribution of our work is comparison to the distributed boosting approach. We conclude that DIvote achieves classification accuracies similar to the distributed boosting approach, with no inter-processor communication at any stage of learning.

The DIvote framework is naturally applicable to the scenario in which data sets for a problem are already distributed. At each of the distributed sites multiple classifiers can be built, and the only communication required is the learned classifiers at the end of training. Inter-processor communication can become a bottleneck if an exchange of data is required across various nodes or processors. Moreover, one can easily conceptualize the applicability of DIvote on a cluster of workstations in a lab, where each workstation independently works on a part of the problem in its main memory.

Acknowledgments

This work was supported in part by the United States Department of Energy through the Sandia National Laboratories ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789. This work was also supported in part by National Science Foundation under grant EIA-0130768. We would like to thank the reviewers for their useful comments. We would like to thank Robert Banfield for his help with the experiment on ASCI Blue Supercomputer. We also thank Aleksander Lazarevic for providing us the training and testing sets of the data sets used in the distributed boosting paper.

References

Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001. ACM.

R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A new ensemble diversity measure applied to thinning ensembles. In *Multiple Classifier Systems Workshop*, pages 306–316, Surrey, UK, 2003.

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

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

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, 1998.

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

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

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

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

N. V. Chawla, S. Eschrich, and L. O. Hall. Creating ensembles of classifiers. In *First IEEE International Conference on Data Mining*, pages 581 – 583, San Jose, CA, 2000.

N. V. Chawla, L. O. Hall, K. W. Bowyer, T. E. Moore, and W. P. Kegelmeyer. Distributed pasting of small votes. In *Third International Workshop on Multiple Classifier Systems*, pages 52 – 61, Cagliari, Italy, 2002a.

N. V. Chawla, T. E. Moore, L. O. Hall, K. W. Bowyer, C. Springer, and W. P. Kegelmeyer. Distributed learning with bagging-like performance. *Pattern Recognition Letters*, 24(1-3):455 – 471, 2002b.

N. V. Chawla, T. E. Moore, Jr., L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, and C. Springer. Investigation of bagging-like effects and decision trees versus neural nets in protein secondary structure prediction. In *ACM SIGKDD Workshop on Data Mining in Bio-Informatics*, San Francisco, CA, 2001.

D. J. Spiegelhalter D. Michie and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

T. Dietterich. An empirical comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning*, 40(2):139 – 157, 2000.

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

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

S. Eschrich, N. V. Chawla, and L. O. Hall. Learning to predict in complex biological domains. *Journal of System Simulation*, 2:1464 – 1471, 2002.

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

U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*, chapter From data mining to knowledge discovery: An overview. AAAI Press, Menlo Park, CA, 1996.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996.

G. Giacinto and F. Roli. An approach to automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22:25 – 33, 2001.

I. J. Good. *The Estimation of Probabilities: An essay on modern Bayesian methods*. MIT Press, 1965.

L. O. Hall, K. W. Bowyer, N. V. Chawla, T. E. Moore, and W. P. Kegelmeyer. Avatar: Adaptive Visualization Aid for Touring and Recovery. Technical Report SAND2000-8203, Sandia National Labs, 2000.

L. O. Hall, N. V. Chawla, K. W. Bowyer, and W.P Kegelmeyer. Learning rules from distributed data. In *Workshop of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 1999.

T. Ho. Random subspace method for constructing decision forests. *IEEE Transactions on PAMI*, 20(8):832 – 844, 1998.

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

L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181 – 207, 2003.

L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin. Is independence good for combining classifiers? In *Proceedings of 15th International Conference on Pattern Recognition*, pages 168 – 171, Barcelona, Spain, September 2000.

P. Latinne, O. Debeir, and C. Decaestecker. Limiting the number of trees in random forests. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, Second International Workshop*, pages 178 – 187, Cambridge, UK, 2001. Springer.

A. Lazarevic and Z. Obradovic. Boosting algorithms for parallel and distributed learning. *Distributed and Parallel Databases Journal, Special Issue on Parallel and Distributed Data Mining*, 11:203 – 229, 2002.

N. Leavitt. Data mining for the corporate masses. In *IEEE Computer*. IEEE Computer Society, May 2002.

Lawrence Livermore National Laboratories. ASCI Blue Pacific. *http://www.llnl.gov/asci/platforms/bluepac*.

Lawrence Livermore National Laboratories. Protein Structure Prediction Center. *http://predictioncenter.llnl.gov/*, 1999.

R. Musick, J. Catlett, and S. Russell. Decision theoretic subsampling for induction on large databases. In *Proceedings of Tenth International Conference on Machine Learning*, pages 212 – 219, Amherst, MA, 1993.

C. Perlich, F. Provost, and J. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255, 2003.

F. 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, Portland, Oregon, 1996.

F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

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, San Diego, CA, 1999.

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

D. B. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *AAAI Integrating Multiple Learned Models Workshop*, Portland, Oregon, 1996.

W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of seventh International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001. San Francisco, CA.