Distributed Pasting of Small Votes

N.V. Chawla[1], L.O. Hall[1], K.W. Bowyer[2], T.E. Moore, Jr.[1], and W.P. Kegelmeyer[3]

[1] Department of Computer Science and Engineering, University of South Florida
4202 E. Fowler Avenue, Tampa, Florida 33620, USA
{chawla,hall,tmoore4}@csee.usf.edu
[2] Department of Computer Science and Engineering, University of Notre Dame
384 Fitzpatrick Hall, Notre Dame, IN 46556, USA
kwb@cse.nd.edu
[3] Sandia National Labs, Biosystems Research Department, P.O. Box 969, MS 9951,
Livermore, CA 94551-0969, USA
wpk@ca.sandia.gov

Abstract. Bagging and boosting are two popular ensemble methods that achieve better accuracy than a single classifier. These techniques have limitations on massive datasets, as the size of the dataset can be a bottleneck. Voting many classifiers built on small subsets of data ("pasting small votes") is a promising approach for learning from massive datasets. Pasting small votes can utilize the power of boosting and bagging, and potentially scale up to massive datasets. 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 to massive datasets.

1 Introduction

The last decade has witnessed a surge in the availability of massive datasets. These include historical data of transactions from credit card companies, telephone companies, e-commerce companies, and financial markets. The relatively new bioinformatics field has also opened the doors to extremely large datasets such as the Protein Data Bank [16]. The size of these important datasets poses a challenge for developers of machine learning algorithms and software — how to construct accurate *and* efficient models. The machine learning community has essentially focused on two directions to deal with massive datasets: data subsampling [14,17], and the design of parallel or distributed algorithms capable of handling all the data [5,15,10,6]. The latter approaches try to bypass the need for loading the entire dataset into the memory of a single computer by distributing the dataset across a group of computers.

Evaluating these two directions leads to the compelling question: "Do we really need all the data?" The KDD-2001 conference [20] conducted a panel on subsampling, which overall offered positive views of subsampling. However, given 100 GB of data, subsampling at 10% can itself pose a challenge. Other pertinent

F. Roli and J. Kittler (Eds.): MCS 2002, LNCS 2364, pp. 52–61, 2002.

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 dataset, which could take away some of the efficiency advantages. Ideally, one might want to use the existing computational resources to handle the flood of training data.

Our claim is that distributed data mining can mitigate, to a large extent, the scalability issues presented by massive training sets. The datasets can be partitioned into a size that can be efficiently managed on a group of processors. Partitioning the datasets into random, disjoint partitions will not only overcome the issue of exceeding memory size, but will also lead to creating diverse classifiers (each built from a disjoint partition, but the aggregate processing all of the data) [7]. This can result in an improvement in performance that might not be possible by subsampling.

To implement this idea, we divide the training set into n disjoint partitions, and then paste *Rvote*[1] or *Ivote*[2] respectively [4] on each of the disjoint subsets independently. We call our distributed approaches of pasting Ivotes and Rvotes *DIvote* and *DRvote* respectively. Breiman has shown that pasting together Ivotes gives accuracy comparable to Adaboost. Our experimental results with Ivote using C4.5 release 8 [18] agree with Breiman's results using CART. We also show that DIvote is comparable to Ivote for small or moderate datasets, while performing better for a large dataset. One major advantage of DIvote is a *significant reduction in training time as compared to Ivote*. We ran the distributed experiments on a 24-node Beowulf cluster, though DIvote and DRvote can be easily applied on a cluster of workstations. Each workstation could build classifiers on disjoint subsets of data at the same time.

2 Related Work

The machine learning community has generally addressed the problem of massive datasets by a "divide and conquer approach" — break a dataset into subsets, learn models on the subsets, and combine them. Chawla et al. [6] studied various partition strategies and found that an intelligent partitioning methodology — clustering — is generally better than simple random partitioning, and generally performs as well as C4.5 learning on the entire dataset. They also found that applying bagging to the disjoint partitions, and making an ensemble of many C4.5 decision trees, can yield better results than building a decision tree by applying C4.5 on the entire dataset.

Bagging, boosting, and their variants have been shown to improve classifier accuracy [9,3,1,8,12]. 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 datasets [7]. We observed that for datasets too large to handle practically in the memory of a typical computer, a committee created

[1] In pasting Rvotes, each small training set is created by random sampling.
[2] In pasting Ivotes, each small training set is created by importance sampling.

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 [7].

Boosting [9] also creates an ensemble of classifiers from a single dataset by looking at different training set representations of the same dataset, focusing on misclassified cases. Boosting is essentially a sequential procedure applicable to datasets small enough to fit in a computer's memory. Lazarevic and Obradovic proposed a distributed boosting algorithm to deal with massive datasets or very large distributed homogeneous datasets [13]. In their framework, classifiers are learned on each distributed site, and broadcast to every other site. The ensemble of classifiers constructed from 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 comprising a sum of local weights, reflecting its prediction accuracy.

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 datasets [13].

Our work is built on Breiman's pasting votes approach [4], which will be discussed further in the following sections.

3 Pasting Votes

Breiman proposed pasting votes to build many classifiers from small pieces or "bites" of data [4]. He proposed two strategies of pasting votes: Ivote and Rvote. Ivote sequentially generates datasets (and thus classifiers) by sampling, so each new train dataset has more instances that were more likely to be misclassified by the ensemble of classifiers already generated. In the Ivote approach, 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 [4], 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 dataset. Rvote requires the creation of many bags of a very small size (bites), and is a fast and simple approach. Breiman found that Rvote was not competitive in accuracy with Ivote or Adaboost. The detailed algorithms behind both the approaches are presented in Section 4 as part of DIvote and DRvote.

Pasting Ivotes entails multiple random disk accesses, which could swamp the CPU times. So Breiman proposed an alternate scheme: a sequential pass through the dataset. 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 dataset. However, the

sequential pass through the dataset approach led to a degradation in accuracy for a majority of the datasets used in the paper. Breiman also pointed out that this approach of sequentially reading instances from the disk will not work for highly skewed datasets. Thus, one important component of the power of pasting Ivotes is random sampling with replacement.

Breiman also noted that the approach of pasting Ivotes is scalable with memory. 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 dataset. The memory requirement for a dataset with 10^5 records and $J = 2$ will be close to a gigabyte; the only increase in the memory will be number of trees stored [4].

In our distributed approach pasting small votes, we divide a dataset 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.

4 Pasting DIvotes and DRvotes

The procedure for DIvote is as follows:

1. Divide the dataset into T disjoint subsets.
2. Assign each disjoint subset to a unique processor.
3. On each processor build the first small training set of size N ("bite") by sampling with replacement from its subset, and learn a classifier.
4. For the subsequent bites on each of the processors, an instance is drawn at random from the resident subset of data with all examples having equal probability of selection [4]. 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 selected for the subsequent bite. If not, then the instance is rejected with probability:

$$e(k) = p \times e(k-1) + (1-p) \times r(k)$$

$p = 0.75$. the same p value as used by Breiman,

$k =$ number of classifiers in the aggregate or ensemble so far, and

$r(k) =$ error rate of the kth aggregated classifiers on a T disjoint subset.

Repeat until N instances have been selected for the bite.

5. Learn the $(k+1)th$ classifier on the training set (bite) newly created by step 4.
6. Repeat steps 4 and 5, until the out-of-bag error estimate plateaus, or for a given number of iterations, to produce a desired number of classifiers.
7. 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 being 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 no intermediate steps of DIvote — steps 4 and 5 in the above algorithm — are required. However, DRvote does not provide the accuracies achieved by DIvote.

Pasting DIvotes or DRvotes has the advantage of not requiring any communication between the processors, unlike the distributed boosting approach by Lazarevic and Obradovic [13]. Thus, there is no time lost in communication among processors. DIvote can essentially build thousands of trees fast, as on each processor (hundreds of) trees are built independently. Furthermore, dividing the dataset into smaller disjoint subsets can also mitigate the need for gigabyte memory servers. Also, if the disjoint subset size is small compared to the main memory on a computer, the entire dataset can be loaded in the memory and randomly accessed; thus, excessive random disk accesses can be avoided. DIvote reduces the data set size on each processor, hence less examples must be tested by the aggregate classifiers during training, which also significantly reduces the computational time.

5 Experiments

We evaluated DIvote and DRvote by experiments on three small datasets, which were also used by Breiman [4], and one large dataset. We did a 10-fold cross-validation for the small datasets. Since our large dataset has a non-homologous test set, we did not run a 10-fold CV. For the small datasets we set N, the size of each bite, to be 800 examples, while for the large dataset we varied the bite size from 1/256th, 1/128th, and 1/64th of the entire dataset size. For the distributed runs, we divided the smaller datasets into 4 disjoint partitions, and the large dataset into 24 disjoint partitions. We also ran experiments with pasting Ivotes and Rvotes to get a benchmark of sequential performance.

5.1 Datasets

Our three small datasets are from the UCI repository [2]. The large dataset comes from the problem of predicting the secondary structure of proteins. Its training and testing sets ("test set one") were used in developing and validating, respectively, a neural network that won the CASP-3 secondary structure prediction contest [11]. The size of these datasets is summarized in Table 1.

5.2 Base Classifier and Computing Systems

We used the C4.5 release 8 decision tree software for our experiments. The sequential Rvote and Ivote experiments were run on a 1.4 GHz Pentium 4 linux workstation with 2 GB of memory, and an 8-processor Sun-Fire-880 with 32 GB of main memory. We ran DIvote and DRvote experiments on a 24-node Beowulf cluster. Each node on the cluster has a 900 MHz Athlon processor and 512MB of memory. The cluster is connected with 100Bt ethernet.

Table 1. Dataset Sizes, Number of Classes and attributes.

Dataset	Dataset size	Classes	Number of attributes
Satimage	6435	6	36
Pendigits	10992	10	16
Letter	20000	26	16
Jones	Training = 209,529; Testing = 17,731	3	315

5.3 Prediction Accuracy Comparison

To statistically validate the results, we performed a two-tailed paired t-test ($\alpha = 0.05$) on the 10-fold cross validation results of the small datasets. Table 2 shows the confusion matrix for statistical significance comparison. We observed identical statistical significance for our experiments with Satimage and Pendigits, so we have combined the two results. The results are reported at the end of 250 iterations. For instance, consider the first row in Table 2. It shows that DIvote is the same as Ivote, but significantly better than any other approach. The letter dataset differed from Satimage and Pendigits, as DRvote and Rvote performed significantly worse than C4.5. However, for letter also, DIvote and Ivote were comparable, and significantly better than Rvote, DRvote, and C4.5. Figure 1(a) shows the results on letter dataset.

Table 2. Significance Confusion Matrix for Satimage and Pendigits. Same = Not statistically different classification accuracy. Better = Statistically higher classification accuracy.

	DIvote	**Ivote**	**DRvote**	**Rvote**	**C4.5**
DIvote		Same	Better	Better	Better
Ivote	Same		Better	Better	Better
DRvote	Worse	Worse		Same	Better

For the large dataset, pasting DIvotes is actually more accurate than pasting Ivotes. In addition to pasting DIvotes and Ivotes accuracies, Figure 2(b) also shows that pasting DIvotes produces a more accurate ensemble of classifiers than the ensemble constructed of classifiers learned on the 24 disjoint subsets. It is also interesting to note that the average accuracy of a decision tree learned on bites of size 1/256th of the Jones dataset 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. Each of the individual DIvote classifiers optimizes the small decision space it is constructed on, so the end result is an ensemble of locally optimum classifiers. Thus, the combined hypothesis of this ensemble of hundreds or thousands of intelligently created classifiers achieves a high accuracy.

We conjecture that the bites (very small training sets) for DRvote are too similar to each other. Creating bites from the disjoint partitions, especially of small datasets, does not provide enough unique instances to each bite. Therefore, it is possible that even an aggregate of all bites is not a good representation of the training set, and the classifiers learned on these bites are not very diverse. For the letter dataset, DRvote and Rvote are significantly worse than C4.5. The letter dataset has 26 classes with 16 dimensions; each 800 sized bite will contain approximately 30 instances for each class. Thus, given the high dimensionality, the 30 examples may not mimic the real distribution of the dataset. To test this model, we created 100% bags on each disjoint partition [6] of the letter dataset, 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 are introducing more coverage, better individual classifiers, and diversity compared to the DRvote bites (800 instances). Since DIvote and Ivote sample heavily from misclassified instances, after each iteration or series of iterations they focus on different instances, thus creating more diverse classifiers with potentially more coverage.

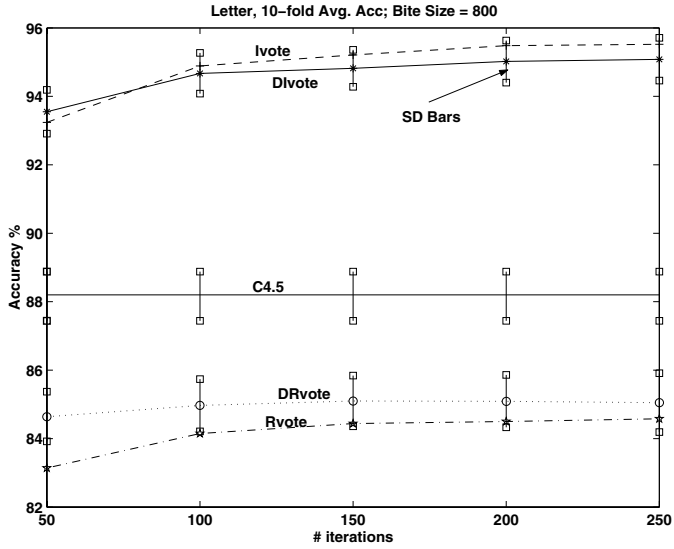

Fig. 1. Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for the Letter dataset

5.4 Timing

Table 3 show the timing (user and system time during training) ratios of DIvote to Ivote, and DRvote to Rvote on the Beowulf cluster. The experimental

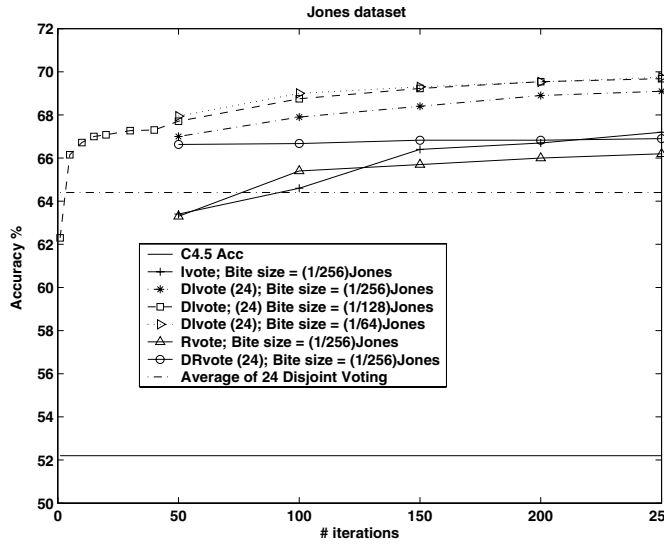

Fig. 2. Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for the Jones dataset

parameters were: number of iterations = 100; bite size $N = 800$ for the small datasets, and bite size $N = (1/256)$ * (Jones dataset size) for the Jones dataset. 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 datasets; $T = 24$ for the large dataset) 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 simultaneously.

One significant advantage of the proposed DIvote approach is that it requires much less time than Ivote. 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 dataset each disjoint partition has only 8730 instances as compared to 209,529 in the entire dataset). Also, a reduction in the training set size implies that the dataset can be more easily handled in main memory. The graphs show that as the dataset size increases, the ratio of DIvote time to Ivote time decreases, which suggests that the time taken for testing the aggregate classifiers, accumulating their votes, computing out-of-bag error, and intelligent sampling increases with dataset size. 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.

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. Nevertheless, DRvote builds T times as many Rvote classifiers in less time.

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

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

6 Conclusion

The overall conclusion of our work is that pasting DIvotes is a promising approach for very large datasets. Datasets 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. Our experiments show that DIvote is a fast, accurate, and scalable framework. We show that pasting DIvotes is very comparable in classification accuracy to Ivotes on the small datasets, and is the best for the large dataset. Our results support the theory that given an ensemble of diverse classifiers, an improvement in the accuracy can be observed. Pasting DIvotes is much faster than pasting Ivotes; pasting DIvotes on a processor takes less time compared to pasting Ivotes on a processor. Each processor works independently, without requiring communication at any stage of learning; the end result is an ensemble of thousands of DIvote classifiers. DIvote is able to build a representative model of the dataset through this ensemble of hundreds or thousands of 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. We wish to understand this more completely, however, and some of our future work is devoted to experiments designed to separate and illustrate these effects.

The DIvote framework is naturally applicable to the scenario in which datasets 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. A section of our future work is also devoted to experiments on the ASCI Red supercomputer [19].

Acknowledgments

This work was supported in part by the United States Department of Energy through the Sandia National Laboratories LDRD program and ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789, and the National Science Foundation under grant EIA-0130768.

References

1. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, Vol 36, 105 – 139. Kluwer (1999).
2. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, University of California, Irvine, Dept. of Information and Computer Sciences (1998).
3. Breiman, L. Bagging predictors. *Machine Learning*, Vol 24. Kluwer (1996) 123 – 140.
4. Breiman, L.: Pasting small votes for classification in large databases and on-line. *Machine Learning*, Vol 36. Kluwer (1999) 85–103.
5. Chan, P., Stolfo, S.: Towards parallel and distributed learning by meta-learning. *AAAI Workshop on Knowledge Discovery and Databases.* (1993) 227 – 240.
6. Chawla, N., Eschrich, S., Hall, L.O.; Creating ensembles of classifiers. *First IEEE International Conference on Data Mining.* (2000).
7. Chawla, N.V., Moore, T.E., Bowyer, K.W., Hall, L.O., Springer, C., Kegelmeyer, W.P.: Bagging is a small dataset phenomenon. *International Conference of Computer Vision and Pattern Recognition (CVPR).* (2000) 684 – 689.
8. Dietterich, T.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, Vol 40. Kluwer (2000) 139 – 158.
9. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning.* (1996).
10. Hall, L.O., Chawla, N.V., Bowyer, K.W., Kegelmeyer, W.P.: Learning rules from distributed data. *Workshop of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* (1999).
11. Jones, D.: Protein secondary structure prediction based on decision-specific scoring matrices. *Journal of Molecular Biology*, Vol 292. (1999) 195 – 202.
12. Latinne, P., Debeir, O., Decaestecker, C.: Different ways of weakening decision trees and their impact on classification accuracy of DT combination. *First International Workshop on Multiple Classifier Systems.* Lecture Notes in Computer Science, Vol 1857. Springer-Verlag, (2000) 200 – 210.
13. Lazarevic, A., Obradovic, Z.: The distributed boosting algorithm. *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* (2000).
14. Musick, R., Catlett, J., Russell, S.. Decision theoretic subsampling for induction on large databases. *Tenth International Conference on Machine Learning*, Amherst, MA. (1993) 212 – 219.
15. Provost, F.J., Hennessy D.N.: Scaling up: Distributed machine learning with cooperation. *Thirteenth National Conference on Artificial Intelligence.* (1996) 74 – 79.
16. Protein Data Bank. *http://www.rcsb.org/pdb/*
17. Provost, F., Jensen D., Oates, T.: Efficient progressive sampling. *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* (1999) 23 – 32.
18. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufman San Mateo, CA (1992).
19. Sandia National Labs.: ASCI RED, the world's first TeraFLOPS supercomputer. *http://www.sandia.gov/ASCI/Red.*
20. *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* http://www.acm.org/sigkdd/kdd2001/ (2001).