Distributed Learning on Very Large Data Sets

Lawrence O. Hall, Kevin W. Bowyer, W. Philip Kegelmeyer*, Thomas E. Moore, Jr. and Chi-ming Chao Department of Computer Science and Engineering, ENB 118 University of South Florida 4202 E. Fowler Ave. Tampa, Fl 33620 *Sandia National Laboratories P.O. Box 969, MS 9214 Livermore, CA, 94551-0969 {hall,kwb,tmoore4,cchao}@csee.usf.edu, wpk@ca.sandia.gov

ABSTRACT

One approach to learning from intractably large data sets is to utilize all the training data by learning models on tractably sized subsets of the data. The subsets of data may be disjoint or partially overlapping. The individual learned models may be combined into a single model or a voting approach may be used to combine the classifications of a set of models. An approach to learning models in parallel from arbitrarily large training data sets and combining them into a classifier is described. The training sets are disjoint in the work described here. A parallel implementation on the DOE's ASCI Red parallel supercomputer is described. Results with data sets small enough to be handled by a single processor show that data sets can be divided into a moderate number of distinct subsets without degrading classifier accuracy. Speedup results are shown for a parallel implementation on the ASCI Red with data sets too large to be handled on a single processor. Training sets of size 3 to 50 million examples are used to generate models on up to 64 processors to show how the learning process scales.

1. INTRODUCTION

There are now many areas in which very large data sets (millions and more training examples) may be acquired. Useful classification models may be derived from many of these labeled data sets. Areas in which a tremendous amount of data is available include visualization [1], phone fraud, long distance telephone calling patterns, credit card fraud [2], etc. The amount of data available for learning may be much greater than the amount of memory available on a single computer. There may be tens of millions of multidimensional training vectors, for instance.

A visualization example where very large numbers of la-

beled examples may be created is the 3D high-resolution physics data sets generated by simulation codes under the DOE's Accelerated Strategic Computing Initiative (ASCI) program [3, 4]. These simulations are replacing important physical experiments, and so must be conducted in the most exacting detail. But the very quality of that detail poses a conundrum, for if a scientist wants to visualize the data at the finest resolution available, in order to catch important and subtle details, they will not have time to look at all of it, and so may not find all of the areas of interest. And vice versa, of course; if they back off and work at a resolution that insures they will see the full extent of the data set, they will miss small anomalies [5].

Thus it is necessary to make visualization tools smarter, to let them begin to understand what in the data is of interest to the user, so that they can navigate, as well as illustrate, the data. One avenue is to invoke pattern recognition techniques, using a visualization session as a supervised learning session in which the user indicates by example, explicitly or implicitly, what is interesting. This will require being able to learn decision rules from very large data sets, which is challenging, as the data will not fit in memory and the amount of time needed to sift the data in a serial fashion will be exorbitant.

Given very large training data sets it has recently been suggested [6] that building individual classifiers on well chosen subsets and then applying voting to classify new examples will allow an accurate model to be built on a single computer. Others have suggested that in very large data sets many of the examples will be redundant [7, 8]. In the context of the above work, the use of subsets or perhaps a single well-chosen subset for training may be the right approach.

In this paper, we consider the case where either all the data (of very large size) is necessary to learn an accurate model, or the subset necessary to learn an accurate model is itself of very large size. In our approach, learning is done in parallel. Each processor learns a rule or decision tree model from its own unique set of examples. Example sets may be disjoint, the simplest case for partitioning the examples, or partially overlapping. Disjoint subsets are used in the experiments described in this paper. A parallel implementation on the Department of Energy's (DOE) ASCI Red parallel supercomputer is described. Experiments show what kind of speedup can be expected from learning models in parallel. Millions of training examples were used in experiments, indicating that this approach scales to very large data sets.

Experiments are performed on data sets from the UC Irvine repository [9] to look at the accuracy of a model learned on disjoint subsets of data versus a model learned on the complete data set. The experiments show that a model learned on disjoint subsets can achieve accuracy comparable to that of a model learned on the complete data set.

The paper proceeds as follows. Section 2 describes the learning algorithms utilized in the experiments reported later and provides a brief description of the ASCI Red supercomputer. Section 3 describes an approach to learning in parallel from very large data sets. Section 4 describes experiments with learning from disjoint partitions of training data sets. Section 5 describes parallel speedup and scale up experiments on the ASCI Red. Finally, Section 6 summarizes results presented in this paper.

2. LEARNING ALGORITHMS AND THE ASCI RED

In the work reported here, two different types of learning algorithms are used, RIPPER [10] and V5. RIPPER is a direct rule learner, which creates a set of rules directly from training data. In the process of building a rule, RIPPER breaks the training data set into a rule growing (learning) set and a rule pruning (prune) set. RIPPER rules have the assigned weight

$$P = \frac{p-n}{p+n},\tag{1}$$

where p is the number of positive examples in the prune set covered by the rule and n is the number of negative examples in the prune set covered by the rule.

V5 is a modified version of the decision tree learning algorithm embodied in C4.5 release 8 [11, 12]. Release 8 has significantly improved handling of continuous attributes, which is important for large-scale visualization data sets which will only have continuous attributes. V5 has been updated from C4.5 in two ways. It has been ported to run on the ASCI Red parallel supercomputer. It also allows for test results from a validation set to be stored as a weight at the leaves of the tree.

RIPPER and C4.5 provide roughly equivalent classification performance, with RIPPER a little better [10]. RIPPER is considered a fast rule learning algorithm and is used in all simulated parallel experiments reported here.

V5 is used in all experiments on the ASCI Red. Decision trees may be significantly faster in the classification of unseen examples when compared with rule sets. In learning from very large data sets, validation sets may be used to weight the decision that the tree makes. The validation sets themselves may be quite large and efficiency in classifying the examples in them may be important. Further, if one wishes to classify millions or billions of test examples using a vote of N classifiers, the faster classification of a decision tree may be important.

2.1 ASCI Red

The DOE's ASCI Red machine [3] has 4,640 compute nodes, each with 2 processors sharing 256MB of memory. The machine is capable of 3.15 Tera FLOPS. There are a total of 9,280 333 MHz Pentium III processors in a distributed memory architecture. The processors are connected in a mesh architecture and run a version of the UNIX operating system.

3. PARALLEL LEARNING ALGORITHM

The parallel learning system discussed here works as follows. Each of N processors is provided a training set. The training sets can be disjoint or they may overlap. A learning algorithm is applied by each processor to its training data set. The resultant model is saved for later use by any other processor. In this paper, the learning algorithm will either be a variation of that used by C4.5 (V5) or it will be the algorithm used by RIPPER.

To determine the class of an unseen example, the example is applied to each of the N models, which results in N classifications. The classifications may be weighted or unweighted. In the case of unweighted classifications, a majority vote is taken to determine the class of the tested example. In the case of weighted classifications, the sum of the weights is obtained for each class and the example is assigned to the class with the largest weight.

The advantage of our parallel learning algorithm is that it allows each processor to work independently. No processor depends on any other. Hence it is fully parallelizable, unlike an approach such as pasting bites [6] or boosting [13, 14], each of which requires that proceeding classifiers be built on training sets determined from already built classifiers.

It is possible that a combiner or arbiter [15, 16, 2] could be utilized with an appropriate training set to provide improved accuracy. The cost would be one more sequential step in the training process and a slightly more time-consuming testing procedure.

4. ACCURACY EXPERIMENTS ON DIS-JOINT TRAINING DATA

In the context of storage requirements, the most reasonable approach to using an extremely large size data set for parallel learning is to break it into disjoint subsets. Such an approach may potentially reduce the accuracy of the resulting classifier in comparison with one that could be built sequentially on the full training data set.

There are other ways in which the data could be given to each processor for learning. One possibility would be to give each processor a fixed size subset of the data that was randomly chosen with replacement. Such an approach would be in the spirit of bagging except that for very large training data sets each subset could be a very small percentage of the total data.

Fifteen data sets were chosen from the UC Irvine machine

learning repository [9] for use in the disjoint data set experiments. The data sets are shown in Table 1. Some of the larger data sets, which for the most part fit on a single processor, are included. Some relatively small data sets are also included to examine what happens in "worst-case" scenarios in which relatively small data sets are subdivided into disjoint subsets for training. We're interested in how much parallelism can be exploited before accuracy becomes lower than that obtained by training the same classifier on the full data set.

We used the relative accuracy shown below to compare our approach with RIPPER [17]. The relative accuracy is denoted by

$$RA = \frac{A_c - F_m}{A_r - F_m},\tag{2}$$

where for a given training set A_c is the accuracy of the classifier being compared to RIPPER, A_r is the RIPPER accuracy, and F_m is the fraction of the overall examples which are comprised of the majority class ($F_m \in [0, 1]$). The relative accuracy attempts to measure how much better or worse the classifier being compared to RIPPER was when compared with guessing that each example belongs to the majority class. RIPPER assigns a "predictive value" (P) to each learned rule. This value is used as a rule weight in the experiments reported here. In a comparison with several other approaches to voting with RIPPER, the above approach was found to provide the best performance [17].

The classification accuracy on a given data set for each classifier is the result of a 10-fold cross validation experiment. For the simulated parallel experiments, the 90% used as training data was split randomly across N processors with each processor receiving an equal size partition.

Most of the data sets could be split across four or more processors without a loss of classifier accuracy based on the result of a one tailed paired t-test (at the 0.05 level) using RA. Table 1 shows the number of processors that could be utilized on a given training data set before significant loss of accuracy occurs. In this case, "significant loss" is considered to be a relative accuracy < 0.98. In Table 1, it can be seen that the page-blocks data set¹ does not allow any partitioning and the reasonably large phoneme data set can only be partitioned into two subsets before the relative accuracy is less than RIPPER's. However, in general reasonably sized disjoint partitions do not result in a reduction of relative accuracy for the voting based classifier.

In Figure 1 we show a graph of the accuracy of our voting based classifier vs. RIPPER for the satellite image data set. For comparison purposes the default accuracy of guessing that each unseen example belongs to the majority class is shown on the graph. For this data set, up to 32 disjoint partitions can be used to produce a classifier of accuracy



Figure 1: Results from learning on disjoint partitions with the satellite image data set.



Figure 2: Results from learning on disjoint partitions with the spambase data set.

which is equivalent to that of RIPPER learning on the full training data set. In Figure 2 we show a graph comparing the spam email database performance of our parallelizable voting approach and RIPPER learning on the full training data set. In this case the performance from voting classifiers created on disjoint data sets becomes worse than RIPPER after four processors are utilized.

5. SPEEDUP AND SCALABILITY

The data sets used in our experiments on the ASCI Red were created by the synthetic data generator described in [18, 19]. Each example consists of 7 continuous attributes and may belong to one of two classes. The advantage of the synthetic data generator is that we can generate arbitrarily large training data sets. Results are reported on synthetic training data sets ranging in size from 25,000 examples to 6,400,000 examples. Unless otherwise stated, all results are an average over ten runs for each set of processors. We use an average because, as will be shown, I/O times can vary greatly depending on the load on the system. Since our jobs run in batch mode, it is impossible to predict the I/O load when the job is run.

The first experiment that we discuss is on a training set of size 800,000 examples. Figure 3 shows time required to run on the ASCI Red for from one to 32 processors. Experiments are run with powers of two number of processors (e.g. $2, 4, \ldots, 32, 64$). The I/O times (including the time required to write the resultant tree, the tree building time, and the total time for learning to be completed) are shown. Despite the fact that loading the data on the processors is a one-time cost, and that there is no subsequent interprocess communication, you can see that the I/O time steadily

¹The page-blocks data set could not be split into even two subsets before overall accuracy declined. This appears to be because there are 3 very small classes (28, 88 and 115) when compared with the majority class of 4913 examples. When splitting the very small classes, the examples in them may begin to appear to be noise when a classifier is built on a subset of examples with, for example, only 14 examples of one class and 2400 of another.

Name of	Description of	Number of		Class	Acceptable
data set		Examples	Classes	$\operatorname{distribution}$	partitions
cov-1	Randomized 5% of	29054	7	138, 475, 869,	4
(covtype)	forest cover			1026, 1788,	
	$\operatorname{ty}\mathbf{p}\mathbf{e}$			$10592,\ 14166$	
cov-2	Sequential 2nd 5%	29053	7	138, 475, 869,	8
(covtype)	of forest cover			1026, 1788	
	$\operatorname{ty}\operatorname{pe}$			$10592,\ 14165$	
DNA-	Exon/intron and	3186	3	765, 767	32
sequence	intron/exon sequence			1654	
Iris	Iris plants	150	3	$(50) \ge 3$	8
letter	Letter image	20000	26	$(734 813) \ge 26$	8
	recognition data				
$\operatorname{mushroom}$	Edible or	8124	2	3916, 4208	8
	$\operatorname{poisonous}$				
optidigits	Optical	5620	10	$(554\ 572) \ge 10$	16
	recognition of				
	handwritten digits				
page-	Blocks of page layout	5473	5	28, 88, 115,	1
blocks	of a document			$329, \ 4913$	
pendigits	Pen-based handwritten	10992	10	$(1042 \ 1056) \ge 10$	8
	digits				
$_{\rm phoneme}$	Real-time French and	5404	2	1586, 3818	2
	Spanish recognition				
Pima	Pima-Indian	768	2	$268,\ 500,$	4
	${ m diabetes}$				
satimage	Satellite image	4435	7	$415, \ 470, \ 479,$	32
				$961,\ 1038,\ 1072$	
segment	Image segmentation	2310	7	$(330) \ge 7$	4
$\operatorname{shuttle}$	Shuttle data set	43500	7	6, 11, 37, 132,	32
				$2458,\ 6748,$	
				34108	
spambase	Spam e-mail	4601	2	1813, 2788	4

Table 1: Summary of data sets used in RIPPER experiments.



Figure 3: Experimental results learning from 800,000 generated examples using 2, 4, 8, 16, 32 and 64 ASCI Red processors.

increases and in fact becomes the dominant time. The tree time is the time required for the slowest processor to finish building and pruning its tree on its training data set. The overall time decreases to a minimum at 16 processors, which results in a speedup of 14.5 times over sequential learning on one ASCI Red processor.

In Figure 4 timing results are shown for 1.6 million examples with learning done on from 2 to 64 processors. In this case,



Figure 4: Experimental results from learning from 1,600,000 generated examples on a varying number of ASCI Red processors (2, 4, 6, 8, 16, 32, 64).

learning could **not** be successfully done on 1 processor, due to the size of the training set. If we project the time required on one processor to be about 1.9 times that on two (and thrashing would likely make it even more), the maximum speedup is approximately 14 times when using 16 processors.

The overall I/O load can be highly variable depending upon what other users are doing with the ASCI Red at any given



Figure 5: Experimental results from learning with 25,000 examples on each of 64 processors. Each plotted point is 1 of 10 trials.

time. An example of the variation in I/O times for 10 experiments with 25,000 examples on each of 64 processors is shown in Figure 5. For seven of the ten trials the I/O time is approximately 100 seconds. For two trials it is approximately 400 seconds and for one trial it is over 1000 seconds. In this example the average time required to complete learning is significantly higher than the median time required. On a dedicated machine, the lower I/O times would be obtained.



Figure 6: Experimental results from learning on 50,000 generated examples per processor on a varying number of ASCI Red processors. On 64 processors there are 3.2 million examples used for learning.

The next three experiments are designed to show the scalability of our parallel learning approach. In Figure 6, timings for 50,000 training examples per processor for 1 to 64 processors (at powers of two) are shown. In Figure 7, we show timings for 100,000 training examples per processor. At 64 processors there were a total of 6,400,000 training examples. In Figure 8, timings for 800,000 examples per processor; at 64 processors there were a total of 51,200,000 training examples. The points on this last graph are average times over only 4 trials. The total time was affected mostly by I/O which increases in an approximately linear way as the number of processors is increased on the ASCI Red.

6. SUMMARY

The work reported here shows that a model may be learned in parallel from disjoint subsets of extremely large data sets, with resulting accuracy essentially equivalent to that of a model learned sequentially on the complete data set. By using overlapping subsets (in a manner similar to bagging), it may be possible to get improved accuracy when learning on extreme data sets. Such extreme data sets might be



Figure 7: Experimental results from learning on 100,000 generated examples per processor on a varying number of ASCI Red processors from 1 to 64. There are 6.4 million training examples with 64 processors.



Figure 8: Experimental results from learning on 800,000 generated examples per processor on a varying number of ASCI Red processors from 1 to 64. There are 51,200,000 training examples with 64 processors.

generated from large-scale visualization problems, where it is important to learn the important regions for future visual data analysis. The disjoint subsets studied here are simple to produce, but potentially make learning more difficult.

It is not clear what the exact limit is on the number of parallel processors which can be utilized on an individual data set before classification accuracy suffers. It is possible that the clever use of overlapping subsets of examples will allow for more parallelism without a reduction in classification accuracy.

Our parallel implementation on the ASCI Red appears to allow the train sets to be as large as the number of available processors multiplied by the number of examples that fit in each processor's memory without causing that processor to thrash. Since there is *no* interprocessor communication once the data is loaded on the processor, this same type of learning implementation could be implemented on a set of distributed workstations utilizing MPI calls [20], for example. Our results suggest that the classification performance of a classifier learned in parallel on the minimum number processors necessary to fit all the examples into the main memories of the processors will be equivalent to what would be obtained if training was to be done on a single processor.

Acknowledgments

This research was partially supported 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. We also thank Cathy Smith who has participated in defining discussions of this work and Nitesh Chawla for help with code and concepts.

7. REFERENCES

- A. Nakano, R. Kalia, and P. Vashishta, "Scalable molecular-dynamics, visualization, and data-management algorithms for materials simulations," *Computing in Science and Engineering*, vol. 1, no. 5, pp. 39–47, 1999.
- [2] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection," in *Proc. KDD-98*, 1998.
- [3] Sandia National Labs, http://www.sandia.gov/ASCI/Red/UserGuide.htm, ASCI Red Users Manual, 1997.
- [4] P. Heermann, "First-generation ASCI production visualization environment," *IEEE Computer Graphics* and Applications, pp. 66-71, Sept/Oct 1999.
- [5] M. Marefat, A. Varecka, and J. Yost, "An intelligent visualization agent for simulation-based decision support," *IEEE Computational Science and Engineering*, vol. 4, no. 3, pp. 77–82, 1997.
- [6] L. Breiman, "Pasting bites together for prediction in large data sets," *Machine Learning*, vol. 36, no. 1,2, pp. 85-103, 1999.
- [7] F. Provost, D. Jensen, and T. Oates, "Efficient progressive sampling," in *Proceedings of the Fifth* ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (S. Chaudhuri and D. Madigan, eds.), pp. 23-32, 1999.
- [8] J. Catlett, "Megainduction: A test flight," in Proceedings of the Eight International Workshop on Machine Learning, 1991.
- [9] C. Merz and P. Murphy, UCI Repository of Machine Learning Databases. Univ. of CA., Dept. of CIS, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html.
- [10] W. Cohen, "Fast effective rule induction," in Proceedings of the 12th Conference on Machine Learning, 1995.
- [11] J. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann, 1992. San Mateo, CA.
- [12] J. Quinlan, "Improved use of continuous attributes in C4.5," Journal of Artificial Intelligence Research, vol. 4, pp. 77-90, 1996.
- [13] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1,2, 1999.

- [14] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Machine Learning: Proceedings of the Thirteenth National Conference*, pp. 148-156, 1996.
- [15] P. Chan and S. Stolfo, "Sharing learned models among remote database partitions by local meta-learning," in Proceedings Second International Conference on Knowledge Discovery and Data Mining, pp. 2-7, 1996.
- [16] P. Chan and S. Stolfo, "On the accuracy of meta-learning for scalable data mining," *Journal of Intelligent Information Systems*, vol. 8, pp. 5–28, 1997.
- [17] C. Chao, "Distributed data mining," Master's thesis, University of South Florida, Tampa, Fl., 2000.
- [18] M. V. Joshi, G. Karypis, and V. Kumar, "Scalparc: A new scalable and efficient parallel classification algorithm for mining large datasets," in *Proceedings of the International Parallel Processing Symposium*, pp. 573-579, 1998.
- [19] M. V. Joshi, G. Karypis, and V. Kumar, "Scalparc: A new scalable and efficient parallel classification algorithm for mining large datasets," tech. rep., University of Minnesota, 1998.
- [20] M. Lauria and A. Chien, "MPI-FM: High performance MPI on workstation clusters," *Journal of Parallel and Distributed Computing*, vol. 40, pp. 4–18, 10 Jan. 1997.