Parallelizing CART Using a Workstation Network

Phil Spector Leo Breiman*

Department of Statistics
University of California, Berkeley

January, 1995

Abstract

The CART (Classification and Regression Trees) program, developed by Breiman, *et. al.*[1985], has been used in a wide variety of disciplines to solve classification problems where the available variables do not meet the usual assumptions for classical analyses. To do this, the procedure considers, for each continuous variable, a cut-off value which is most effective in discriminating between groups in question, and for each discrete variable, a partitioning of the discrete values to achieve the same goal. The procedure continues recursively by applying the same criteria to the subgroups of the data which were formed by earlier splits. Since, at each level of the data splitting, a potentially large number of variables must be considered, and since at a given point in the process the partitioning ability of each variable is measured independently of other variables, the technique lends itself very naturally to parallel processing, with each processor evaluating the effectiveness of some subset of the variables, and a central process coordinating the information.

This paper investigates a scheme for parallelizing the CART algorithm, using the commercially available program C-Linda (Johnson[1988] and Wolfe[1992]), on a network of 20 Sun Microsystems

*Partially supported by NIH Grant PR7536

SparcStations. While some reductions in execution time were obtained, they were considerably smaller than expected, and increasing the number of machines used sometimes increased execution time instead of decreasing it.

1 Introduction

Some years ago, one of the authors (Phil Spector) wrote software that distributed large Monte Carlo simulations out over our departmental workstation network (currently 40 Sun Microsystems SPARCstations), gathered the results and aggregated them as desired. Since that time, our network has functioned at supercomputer speed on Monte Carlo problems. These problems have a structure that made them easily disassembled into pieces needing almost no intercommunication between workstations. The results led us to hope that problems that could be cut into pieces not requiring much intercommunication could also be handled effectively on a workstation network. To explore this, we decided to try to parallelize a version of CART over the network.

CART(see Breiman *et. al.*[1985]) is a program that constructs a binary prediction tree for regression and classification. Since it searches through many candidate splits at each node, tree construction is fairly compute intensive compared, say, to linear regression. Also, because CART cuts the data into disjoint pieces and then operates separately on each piece, we believed that intercommunication requirements were not extensive. In short, CART seemed a good test bed to see how effectively a workstation system could be employed as a parallel computer on statistical procedures.

The steps in this experiment were the following. First, a special version of CART was written designed to minimize network communications requirements. Second, the program was distributed over the network using the C-Linda parallelizing language (See Johnson[1988] and Wolfe[1992]). Third, we experimented with the sizes of the problems fed into the parallel CART program and with the number of workstations used to get results on running times.

Our conclusions are disappointing. Network communication speed and capacity form a narrow bottleneck as compared to CPU speed. Using multiple workstations decreased running times only for large problems, and even for these, any decrease using 4 or 5 machines was

as large or larger as when 20 machines were used.

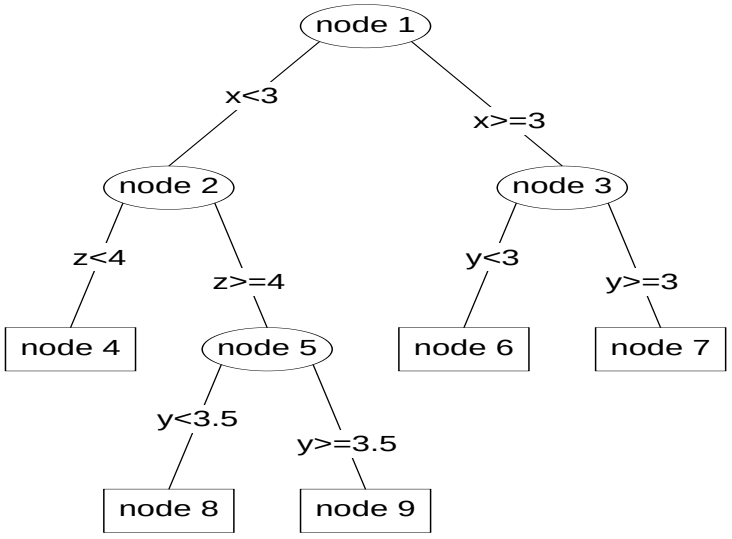
Our outline is this: in the next section (2) we give a brief description of the basic CART concepts. Section 3 describes the distribution strategy in the CART program designed for this project. Section 4 describes the C-Linda implementation, and Section 5 presents the results of our experiments. Concluding remarks are given in Section 6.

2 The CART Program

The basic strategy of the CART algorithm for classification provides a recursive partitioning of a multivariate sample in the following way. At each stage of the partitioning procedure, each variable is studied for its potential in classifying the observations in the sample into their actual groups. This is done by examining how effectively each split of the variable divides the data with respect to the observation's group membership. In this context, a split is a binary division of the observations. For continuous predictor variables, each value found in the data set is considered; observations less than or equal to the value are placed into one group (known as the left node) and observations greater than the value are placed in the right node. For categorical variables, the splits are defined as all possible binary divisions of the unique values of the categorical variable. To assess the quality of each split, a measure of node impurity is employed. A variety of such measures are available; in this study, the measure used was the Gini measure. To determine the way the observations will be divided at each stage of the recursive procedure, the most effective split for each variable is compared in terms of its ability to reduce the impurity of the resulting nodes which such a split would produce. The value of the split for that variable which produces the optimal reduction in impurity is then used to divide the node which is being currently considered. The next step in the recursive procedure is to apply the same technique to both the left and right nodes created in the previous step. Finally, after a prespecified number of nodes is formed, pruning techniques are used to eliminate some of the later splits which were not sufficiently useful in reducing node impurity. (In this study, the final pruning was done by a single machine; only the actual splitting of the data was distributed to machines on the network.) The final

result of the analysis is usually displayed in the form of a classification tree, as shown in Figure 1.

Figure 1: Display of a CART analysis



Note that once the data have been split in two, then the tree construction on each piece of the data is carried out independently of the work on the other piece. Also, the search for the best split on a specified variable at a node is carried out independently of the values of the other variables.

3 Distribution Strategy

At first look, an obvious strategy for distribution suggests itself: use one CPU to get the first split. Then farm out the data in the left and right nodes to different workstations. When each of them is split, farm out the pieces to four workstations, and so on. This strategy was considered and rejected. Its advantage is that once the initial data transfer is done, little intercommunication is necessary. The negatives are that if the initial data set is very large, it may not fit into the memory of a single workstation and paging is slow. The initial split, with a single CPU doing all the work, may be slow. The transfer from disk of large amounts of data as the construction filters down may also slow the construction. Instead, a strategy was adopted which does a more balance allocation of computing and memory. Each workstation is assigned all data in a certain number of variables. For instance, in a classification problem with 60 variables, if 20 workstations were used, each is assigned 3 variables.

If the problem is classification with 100,000 cases, then each workstation would be loaded with the 100,000 values of a case number, classification of that case, and the values of 3 variables in that case. This requires only a modest amount of memory. The computation would proceed as follows: each workstation would know which values of its variables were in a given node. At a signal from the mother machine, it would search through all of its variables to find which one gave the best split, at what location and what the value of the Gini criterion was at the best split. These three numbers would be communicated from the daughter machines to the mother machine. The mother machine would find the best split among all those proffered. Then all machines are notified of which cases went right and left and rearrange the values of their variables. More specifically, when an optimal split is identified for a particular variable, all the other variables from observations in the affected node must be reordered to correspond to this optimal split. In this way, the values for observations in each node are contiguous in the memory of the computer.

This distribution also has drawbacks. One is that the next stage of the recursive partitioning can not take place until each of the machines which is evaluating the splits has completed its task. Thus, if one machine which is involved in the computations is markedly slower than the others, then the faster machines will have to wait for the slower

one to complete its computations before proceeding to the next stage of the procedure. The other, and more serious, is the increased need for communication among the different machines once the optimum split is found. Since the data being analyzed must be reorganized into left and right nodes at each stage of the procedure, the machine which has found the optimum split must broadcast information regarding the ordering of the variable in question to all the other machines, so that they may update the node membership before advancing to the next stage of the procedure.

4 Implementation Using C-Linda

Simulations were carried out using the C-Linda system[1]. In this system, communication between machines on the network is carried out by placing objects in a conceptual tuple-space. An object in tuple space consists of one or more elements which may be character or numerical in value. Any machine participating in the parallel execution of the program can either place objects in tuple space, or, by specifying a partial tuple, extract objects which match certain criteria. Thus, the first step in the parallel execution of CART consists of a central computer (hereafter referred to as the mother) determining how many machines are available, and how many variables in the data set will be assigned to each other machine (hereafter referred to as the children). Next, the appropriate data is placed in tuple space with an identifier unique to the machine which will be responsible for the data, and each child machine extracts its data from tuple space. Note that this allows analysis of extremely large data sets, since no one computer needs to have access to all the data at any time; the mother computer can simply read the data from disk in small pieces, and then distribute it around the network through the tuple space. At each stage of the iterative process, the children report back to the mother by placing a tuple with their unique identifier and the maximum reduction in node impurity which was achieved for any split within the variables assigned to that child. Then, the mother machine determines the variable whose optimum split was the overall most effective at reducing node impurity. Note that this requires responses from each of the children before the recursive splitting of the nodes can continue.

[1]Linda is a trademark of Scientific Computing Associates, New Haven.

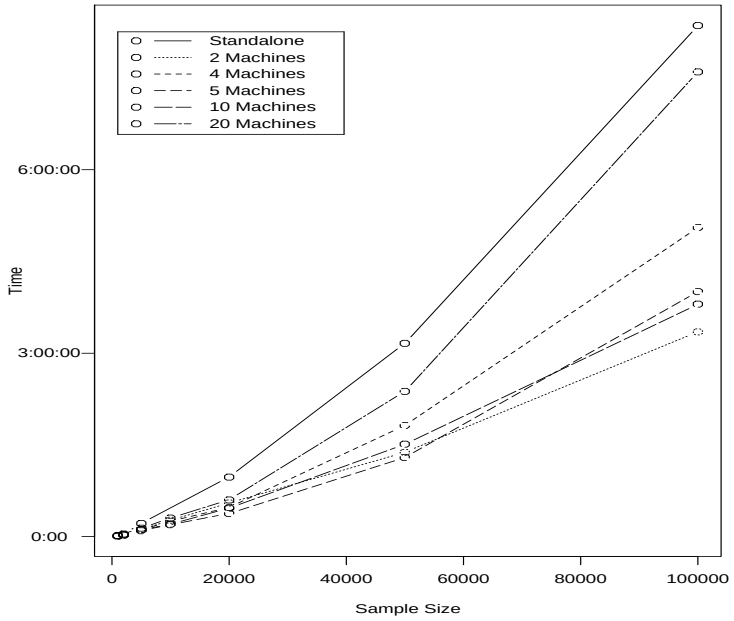
When all the children have reported, the mother machine informs the child responsible for the variable which resulted in the globally optimal split that it should transmit its reordering information to all the other children. Once each of the other children extracts the reordering information from tuple space and reorders the observations for its variables, the recursive process continues. Finally, after a prespecified number of nodes is formed, the mother machine prunes the tree using the techniques described in Breiman, *et. al.*(1985). The decision rule thus obtained can then be displayed in a diagram similar to Figure 1, or new test cases could be assigned to one of the groups based on the decision rule.

5 Simulation Results

A simulation study was designed to investigate the effects of the numbers of observations and variables, and the number of machines used, on the execution time of the CART algorithm. Twenty workstations, mostly SUN Sparc-1+s, connected by a 10 Mbit ethernet network were employed in the study. Sample sizes (number of observations) of 1000, 2000, 5000, 10000, 20000, 50000 and 100000 were used, each with either 20 or 40 variables. Every third variable was categorical, with between 8 and 12 categories, and the dependent variable had three levels. For all combinations of observations and variables, the algorithm was executed on a single workstation. In the 20 variable case, the algorithm was executed in parallel across the network using 2, 4, 5, 10 and 20 machines, corresponding to 10, 5, 4, 2 and 1 variables per machine, respectively. In the 40 variable case, 4, 5, 8, 10 and 20 machines were used, corresponding to 10, 8, 5, 4 and 2 variables per machine. The execution times, measured in minutes of actual clock time, are displayed in Figures 2 and 3.

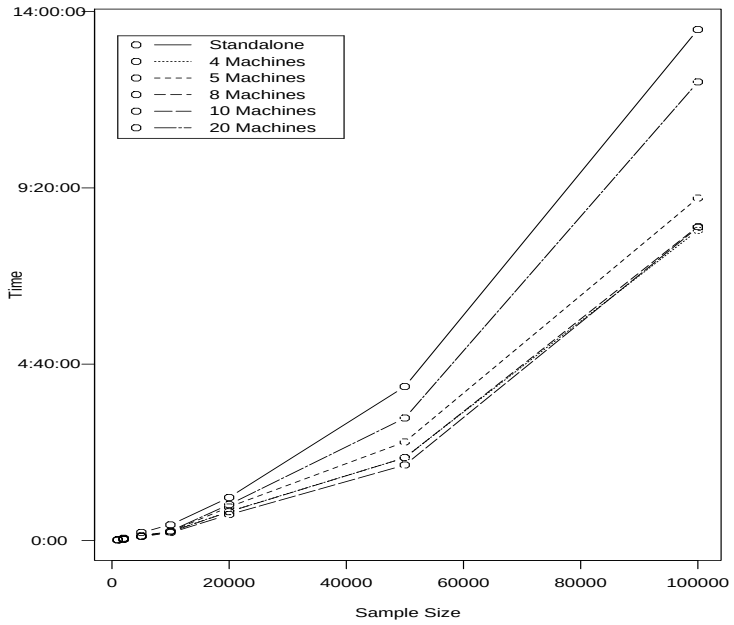
Examination of these graphs shows that, while there is a reduction in execution time of the programs when multiple machines are used, the reduction is by no means linear with the number of machines used. For example, in the case of 20 variables and 10000 observations, the execution time for a single machine was 1784 seconds. By using five machines in parallel on the network, this time was reduced to 702 seconds, roughly 40 percent of the single machine time. This is half as fast as the optimal reduction which one might expect from using

Figure 2: Results of Simulation for 20 variables



five machines. In addition, increasing the number of machines above five was not effective in reducing execution time. Using 20 machines, the execution time was 1090 seconds, or a little over 60 percent of the single machine time, and far slower than the theoretical maximum. For the CART algorithm described above this is primarily due to the increased computational burden of distributing the data among the machines and the communication among the machines necessary to progress through the recursive splitting. In addition, since all the machines must finish their computations at each stage before the next recursive partitioning can begin, problems of synchronicity among machines increase as more machines are used. This is especially critical if there are differential loads on the machines used for parallelization due to other user's activities on those machines. A similar pattern exists for the 40 variable case, with 10 machines resulting in the shortest execution time, approximately 50 percent of the time required for a single machine. Thus, more machines do not necessarily reduce the execution time, and even the most effective configuration of machines

Figure 3: Results of Simulation for 40 variables



does not achieve a reduction in time close to the theoretically maximum reduction.

6 Conclusions

For the problem at hand, parallelization clearly is not a panacea, and large reductions in time will not be achieved through parallelization across the network. However, reductions in time on the order of 40 to 50 percent are not unusual. This becomes especially important for the case of very large data sets, where no single machine on the network would be able to accommodate all of the data. In such a case, the parallelized technique could still be feasible, since each machine only needs to access a fraction of the data. Of course, communication delays would be increased as the size of the data set increases, but for very large data sets, this technique might provide the only practical alternative.

A basic problem of parallelization is that techniques which may be computationally efficient on a single processor may not necessarily lead to similar efficiencies when a program is parallelized across a network. In the current study, since each variable is evaluated independently of the others at each stage of the recursion, the parallelization scheme seemed very natural.

But, the price which is paid is that an array of ordering information must be broadcast to all the machines in the network at each stage of the partitioning process, so that the data residing in the memory of each machine can be appropriately reordered. While this strategy is effective when data is stored on a single machine, it is an open question as to what other techniques might be more efficient for keeping track of node membership when the data is distributed among many machines on a network. It may be necessary to rethink the algorithm from first principles in order to arrive at an effective solution.

Many of the difficulties encountered in the present study might be alleviated if a shift was made from parallelization across a network to parallelization through a single shared memory computer with multiple processors, or if the speed of the network interconnecting the machines used was increased. These issues and challenges need to be addressed more thoroughly as parallelization of statistical algorithms becomes more widespread.

7 References

1 Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1985). *Classification and Regression Trees* , Wadsworth/BrooksCole, Monterey, CA.

2 Johnson, R.C. (1988) Linda as a memory model turns sequential to parallel,*Electronic Engineering Times*, **493**, 60.

3 Wolfe, A. (1992) Parallel tools kick applications into high gear, *Electronic Engineering Times*, **686**, 30.