From: ISMB-94 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Distributed Machine Learning: Scaling up with Coarse-grained Parallelism

Foster John Provost
Daniel N. Hennessy
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260
{foster | hennessy}@cs.pitt.edu

Abstract

Machine learning methods are becoming accepted as additions to the biologist's data-analysis tool kit. However, scaling these techniques up to large data sets, such as those in biological and medical domains, is problematic in terms of both the required computational search effort and required memory (and the detrimental effects of excessive swapping). Our approach to tackling the problem of scaling up to large datasets is to take advantage of the ubiquitous workstation networks that are generally available in scientific and engineering environments. This paper introduces the notion of the *invariant-partitioning property*--that for certain evaluation criteria it is possible to partition a data set across multiple processors such that any rule that is satisfactory over the entire data set will also be satisfactory on at least one subset. In addition, by taking advantage of cooperation through interprocess communication, it is possible to build distributed learning algorithms such that only rules that are satisfactory over the entire data set will be learned. We describe a distributed learning system, CorPRL, that takes advantage of the invariant-partitioning property to learn from very large data sets, and present results demonstrating CorPRL's effectiveness in analyzing data from two databases.

Introduction

Machine learning (ML) methods are becoming accepted as additions to the biologist's data-analysis tool kit. In the domains of biology, medicine, high energy physics, chemistry, and others, ML techniques have helped to discover new rules and correlations [Provost, Buchanan, Clearwater, Lee, & Leng, 1993]. However, scaling these techniques up to large data sets is problematic in terms of both the required computational search effort and required memory (and the detrimental effects of excessive swapping). Catlett estimates that on current hardware ID3 [Quinlan, 1986] would take several months to learn from a million records in the flight data set from NASA [Catlett, 1991a]. Furthermore, this is a problem which will continue to get worse, especially for biological domains. The Human-Genome Project is only one example of huge data sets that are either currently or will soon be available for study. Other problems exist that necessitate the use of large data sets [Catlett, 1991a; Danyluk & Provost, 1993; Provost & Aronis, 1994; Rendell & Seshu, 1991].

Our approach to tackling the problem of scaling up machine learning for large datasets is to take advantage of the ubiquitous workstation networks that are generally available in scientific and engineering environments. We are exploring the coarse-grained parallelization of ML techniques across a heterogeneous network of workstations (including IBM-PC's). The current focus is the partitioning of the set of examples among the workstations and the communication of results between them. The machine learning algorithm is only slightly modified to allow a wrapper for communication around it.

This paper introduces the notion of the invariant-partitioning property--that for certain evaluation criteria it is possible to partition a data set across multiple processors such that any rule that is satisfactory over the entire data set will also be satisfactory on at least one subset. In addition, by taking advantage of cooperation through interprocess communication, it is possible to build distributed learning algorithms such that only rules that are satisfactory over the entire data set will be learned.

In the next section we first present an example of a simple, intuitive evaluation criterion that exhibits the invariant-partitioning property. We then present a more useful rule evaluation criterion that also exhibits the invariant-partitioning property. Next we present a distributed learning system, CorPRL, that takes advantage of the invariant-partitioning property to learn from very large data sets, and present results demonstrating CorPRL's effectiveness in analyzing protein crystallography data and infant mortality data. Finally we discuss other approaches to scaling up learning to very large data sets, and conclude by discussing future directions.

The Invariant-Partitioning Property

In order to discuss the invariant-partitioning property, it is useful to define some terms. A rule, r, is a class description that will either cover or not cover each example in a data set, E. Thus, coverage statistics can be determined for r and E. Let P be the number of positive examples in E. Let N be the number of negative examples in E. The number of true positives, TP, is the number of positive examples in E covered by r. The number of false positives, FP, is the number of negative examples in E covered by r. For a subset, E_i, of E, T_i, N_i, TP_i, and FP_i are defined analogously.

Now, let us define a rule evaluation criterion to be the combination of a rule evaluation function, $f(r,E)$, which takes a rule and an example set and produces a scalar evaluation score, and a threshold, c. With respect to the rule evaluation criterion, a rule, r, is *satisfactory* over an example set, E, if $f(r,E)>=c$.

Given a set of examples, E, and a partition of E into N disjoint subsets, E_i, $i=1..N$, the invariant-partitioning property is the phenomenon that for some rule evaluation criteria the following holds: if a rule r is satisfactory over E, then there exists an i such that r is satisfactory over E_i.

The implication of the invariant partitioning property is that distributed learning algorithms can be designed such that each processor has only a subset of the entire set of examples, but every rule that would appear satisfactory to a monolithic processor (with the entire set) will appear satisfactory to at least one distributed processor. In addition, by taking advantage of communication between processors, it is possible to ensure that for the distributed algorithm as a whole, only rules that would appear satisfactory to a monolithic processor will appear satisfactory to the set of distributed processors.

Positive threshold criterion

Let us now consider a simple, intuitive example of the invariant partitioning property in action. Consider a particular rule evaluation criterion: $f(r,E)=TP/P$, $c=T$ (a constant). This *positive threshold* criterion states that a rule is satisfactory if the fraction of the positive examples that it covers exceeds a threshold.

Intuitively the invariant partitioning property can be seen to hold by the following argument. Assume for the moment that we partition E such that for a given rule, r, $f(r,E_i)=f(r,E)$ for all E. Then clearly if r is satisfactory over E, then r is satisfactory for at least one E_i (in fact, for all E_i).

Now assume that we remove enough positive examples covered by r from subset E_x and place them in other subsets, such that $f(r,E_x)<T$. There must be another subset, E_y, for which the number of positive examples has increased, and thus $f(r,E_y)>T$. In other words, whenever the positive coverage statistic is lowered in one subset, it is necessarily increased in some other.

A simple proof that r will be satisfactory over at least one subset follows:

Assume that for a rule, r:
$TP/P > T$

But, given a partition of n subsets of E:
$\forall i,\ TPi/Pi < T$

(i.e., r is not satisfactory over any E_i)

Then:

1) $\forall i\,\{TP_i < T * P_i\}$
2) $\Sigma(TP_i) < \Sigma(T * P_i)$
3) $\Sigma(TP_i) < T * \Sigma(P_i)$
4) $TP < T * P$
5) $TP/P < T \implies$ Contradiction

Limitations of the property

Unfortunately, the invariant-partitioning property does not hold for every possible evaluation criterion. For instance, it holds for the positive coverage criterion, and in an analogous fashion for a negative coverage criterion. However, a useful strategy is to combine the two by requiring that a rule meet both criteria [Clearwater & Provost, 1990]. This conjunction of independent statistics will not produce the invariant-partitioning property. The problem stems from the requirement that both criteria be satisfied in the same partition. Although it is the case that for a rule, r, if both conditions are met in the overall example set there will necessarily be a non-empty set of partitions for which the positive-coverage criteria is met and there will necessarily be a non-empty set of partitions for which the negative-coverage criteria will be met; there is no guarantee that these sets will intersect. The two independent criteria do not exhibit the "averaging" across subsets necessary for the invariant-partitioning property.

There are several ways that this problem of multiple criteria could be addressed for a distributed rule-learning system. One way is to relax the negative threshold based on the number of partitions. However, even with an very small negative threshold, the adjusted negative thresholds become virtually useless even after partitioning the data set into only a few subsets. Another way to address this problem is to allow the criteria to function disjunctively during rule generation and check for the conjunction during a distributed testing stage. However, this will generate an unnecessarily large number of rules to be tested, a computationally expensive process. Alternatively, rules which meet one of the criteria by any of the partitions could be placed on a blackboard, and a rule would not be promoted to the testing phase until the other criterion is met by at least one of the subsets (not necessarily the one that originally posted the rule).

Certainty factor criterion

However, a criterion based on multiple statistics can exhibit the invariant-partitioning property, if the statistics do not act independently when evaluating the criterion. A useful method of evaluating rules based on both the positive and negative coverages is to use a certainty factor. Define the *certainty factor criterion* as: $f(r,E)=TP/(TP+FP)$, $c=CF$. That is, the certainty factor for a rule is the estimated probability that the rule is correct when applied. We have been using a rule-learning program to assist public health scientists in the analysis of infant mortality data. The certainty factor criterion corresponds directly to infant survival rate, when rules are learned to predict live births (and corresponds directly to infant mortality rate when rules are learned to predict infant death). The scientists are interested in finding demographic groups with unusually low/high infant mortality rate--and a rule learner based on the certainty factor criterion can search for just those groups.

The certainty factor criterion also exhibits the invariant-partitioning property. Similar to the proof for the positive threshold criterion, consider the following:

Assume that for a rule, r:
$TP/(TP+FP) > CF$

But, given a partition of n subsets of E:
$\forall i,\ TP_i/(TP_i + FP_i) < CF$

(i.e., r is not satisfactory over any E_i)

Then:

1) $\forall i \{TP_i < CF * (TP_i + FP_i\}$
2) $\Sigma(TP_i) < \Sigma(CF * (TP_i+FP_i))$
3) $\Sigma(TP_i) < CF * \Sigma(TP_i+FP_i)$
4) $TP< CF * (\Sigma(TP_i)+\Sigma(FP_i))$
5) $TP < CF * (TP+FP)$
6) $TP/(TP+FP) < CF$ ==> Contradiction

A Distributed Learning System

We have designed and implemented CorPRL (Coarse-grained Parallel Rule Learner) based on the invariant-partitioning property. The key to CorPRL is the partitioning and distribution of the examples across a network of conventional workstations each running an instance of a machine learning program that is able to communicate partial results to the other programs.

CorPRL

CorPRL is based upon the Rule Learner (RL) program [Clearwater & Provost, 1990; Provost, et al., 1993], a descendant of the Meta-DENDRAL system [Buchanan & Mitchell, 1978]. RL performs a general-to-specific heuristic search of a syntactically defined space of rules directed by a partial domain model. Given a learning problem, *i.e.*, the names of one or more target classes, a set of their examples, and a partial domain model (PDM), RL searches for rules by examining a large but limited number of combinations of features. The plausibility of a rule is determined by its performance (how accurately it classifies examples) and its concordance with assumptions, constraints, and domain knowledge. The results is a disjunction of **IF**-*condition*-**THEN**-*class* rules, where a condition is a conjunction ("AND") of features with a statistically computed likelihood of accuracy of the prediction.

In CorPRL, as a local copy of RL discovers a plausible *candidate rule* (one that is locally satisfactory), it broadcasts the rule to the other machines to gather the statistics globally. If the rule meets the evaluation criterion globally, it is posted as a satisfactory rule.

The first step is to partition the data into N subsets (where N is the number of machines being used). Each of the subsets is then assigned to a machine. The next step is to provide the infrastructure for communication when individual processes detect a plausible rule. When a rule meets the evaluation criterion for a subset of the data, it becomes a candidate for meeting the evaluation criterion globally; the invariant-partitioning property guarantees that each rule that is satisfactory over the entire data set will be satisfactory over at least one subset. It is important to note that the invariant-partitioning property does not guarantee that every candidate rule will be satisfactory. It is possible, due to the distribution of the data during partitioning, that the statistics for the rule are artificially high (*e.g.*, one subset happens to get many of the true positive examples for a rule and none of the false positive examples). However, processes can cooperate to ensure that the system as a whole produces only globally satisfactory rules. When a local RL run discovers a candidate rule, the rule's statistics are reviewed against the data in the other sets of the partition.

To facilitate this review, processes are created that are dedicated to receiving messages about candidate rules, computing the statistics for the rules, and communicating the results back to the creator of the rule. Initially, this rule checker was implemented as a single, central process that examined the entire data set. We are currently in the process of distributing this task as co-processes to the RL runs. Under this scheme, a candidate rule would be broadcast to the other RL processes, which would each update the statistics for the rule based on their local data set, and return their results.

The RL processes are started on the individual machines. As each produces a candidate rule, it sends a message to the checkers which return the global statistics. If the global statistics satisfy the evaluation criteria, the rule is reported as a "good" rule. If it is not satisfactory, its local statistics are replaced with the global statistics and the rule is made available to be further specialized.

The communication between the RL processes is setup via TCP/IP socket connections between the processes. A central communication server is responsible for the distribution of messages. As a process is started, it logs itself with the communication server. As connections become necessary to transfer messages, the server establishes the link between the specific processes. This provides a flexible communication network that we have currently tested on a variety of unix workstations as well as IBM-style PC's.

Experimental Results

The high-level goal of our work is study the use of machine learning techniques as aids to scientists for the exploratory analysis of data. We have applied CorPRL to two collaborative projects with scientists from the University of Pittsburgh. One domain is protein crystallography. We have applied CorPRL to data on crystal growing experiments for X-ray crystallography of proteins and other macromolecules. The point is to find heuristics (rules) that associate procedures (such as rapid cooling), additives, or other factors, with the growth of crystals of sufficient size and regularity that X-ray data may be obtained from them. Gilliland [Gilliland, 1987] has constructed a registry, called the Biological Macromolecule Crystallization Database (BMCD), which consists of 1025 successful crystallization conditions over 820 macromolecules (with over 100 features).

The second problem we are considering is the analysis of public health data on infant mortality, based on database of 3.5 million infant births and deaths from the U.S. Department of Health. The infant mortality researchers are interested in finding demographic subgroups with unusually high or low infant mortality rates, with the ultimate goal of discovering knowledge relevant to public policy.

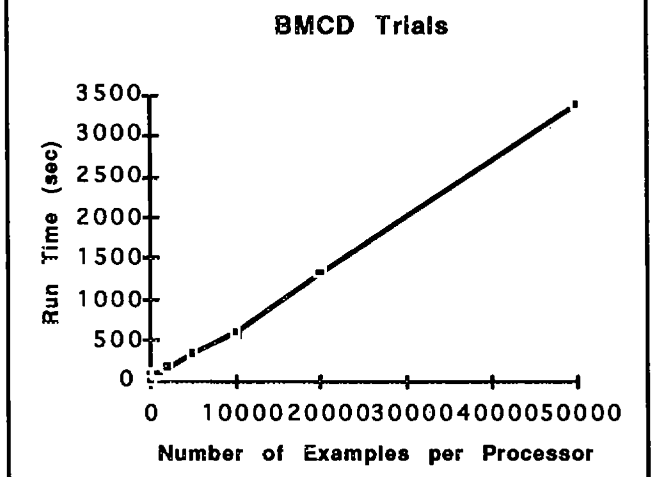

Figure 1: Run Time vs Number of Examples for the BMCD Data

High efficiency

Our experiments to date have concentrated on studying the degree of scaling obtainable with simple distributed learning techniques. Analytic results predict a linear slow down in the run time of RL with increasingly large data sets. Let T_p be the run time of the parallel learner and T_s be the corresponding run time of a serial version of the system, run on the same problem (with a monolithic example set). The *speedup*, S, of the parallel system over the serial version is defined as T_s/T_p. For N processors, the efficiency of the parallel system is S/N. An optimistic target for a distributed learning system would be an efficiency of 1. That is, by dividing the problem into N subproblems a speedup of N can be achieved. This would mean that the problem is perfectly decomposable, and the property of any overheads of running the distributed system is trivial.

For the BMCD data we simulated a large problem by taking 500 examples as a base example set, and creating larger and larger data sets by duplicating the examples (*i.e.*, 2000 examples would be 4 copies of the original 500). This ensures that the same space is searched for all experiments, factoring out the property of different degrees of pruning, *etc.* We then ran the basic RL system on the series of data sets on state-of-the-art workstations and recorded the run times. As Figure 1 shows, the increase in run time is linear in the size of the data set.

Consider the numbers for a data set of 20,000 records. A single, monolithic run of RL would take approximately 20 minutes--annoyingly slow for a researcher trying to answer an interesting question. By taking advantage of 10 workstations in a laboratory network, a run of CorPRL takes approximately 3 minutes. This yields an efficiency of about .82, indicating that there is a small penalty for dividing up the data and running subprocesses.

A major possible source of distributed overhead is the cost of communication and the global rule reviewing. However, since the reviewing of individual rules is an off-line problem, and will be performed by a separate process running concurrently (in a Unix environment) with the rule learning, communication synchronization does not slow down the main learning process. For the current version of the system, a process on a separate workstation does the rule review, which would decrease the efficiency numbers by adding a processor to the denominator. For example, in the above problem the efficiency would have to be calculated using N=11, yielding an efficiency of .75 (which is still very efficient).

We are currently implementing a decentralized rule review. Our preliminary results indicate that the rule review cost is negligible as compared to the cost of searching the rule space. For example, when a single process is run on 2000 examples (as above), and a separate process is used to review the rules it generates, no appreciable slow down is noticed. (In practice, the rule review process would also be reviewing the candidate rules found by other processes; however, all preliminary results indicate that the slowdown is negligible.)

The BMCD experiments show the effect of partitioning the data across multiple processors, with essentially the same search performed on the subprocesses as was performed on the monolithic processor. We have also run experiments with the infant mortality data to observe the effect of partitioning the data set in a more realistic setting. The infant mortality data set comprises over 3 million records. We separated out the approximately 50,000 records corresponding to births in the state of Pennsylvania, and ran RL on increasingly large subsets of this set. Note that for this data set, very large numbers of data points are *necessary* for learning, because there are very small demographic groups that are of interest to the public health researchers. In order to have statistical confidence in a rule learned about a very small group, a large number of records is needed.

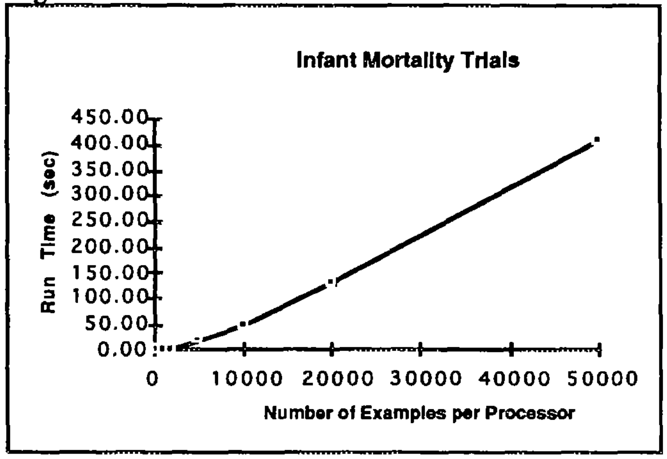

Figure 2: Run Time vs Number of Examples for the Infant Mortality Data

As Figure 2 shows, a similar relationship is observed on the infant mortality data as is observed on the BMCD data. The data points in Figure 2 represent averages over 3 trials with randomly selected subsets of the 50,000 records (the final point is from a single run). Considering a data set of size 20,000 the run time of RL is approximately 2.2 minutes (for these experiments, only a subset of the features was selected). Ignoring the cost of overhead, a run of CorPRL would take approximately 5.7 seconds. Interestingly, even when an extra processor is used for the rule review, the efficiency of such a partitioning is approximately 2.1 (indicating what is sometimes known as a superlinear speedup).

The invariant-partitioning property is observed

In order to examine whether for these runs the invariant partitioning property is indeed observed (as the above theory predicts), we examined the rules learned by the multiple processors in runs of CorPRL using multiple processes on multiple workstations and compared them to the rules learned by a monolithic RL using the union of the CorPRL processors' data sets.

On the BMCD data, we took the set of 500 examples and split it into 5 sets of 100 examples each. CorPRL was run using 6 workstations (one running the rule review process). As expected, the individual CorPRL processes learned different rule sets: some did not find all the rules found by the monolithic RL; some produced spurious rules that were not validated by the global review. However, as predicted by the invariant-partitioning property, the final rule set produced by CorPRL was essentially the same as the final rule set produced by the monolithic RL. The only difference in the rule sets was that CorPRL found some extra rules not found by RL. This is due to the fact that RL conducts a heuristic (beam) search. Because of the distribution of examples across the subsets of the partition, some processors found rules that had fallen off the beam in the monolithic search. Thus, the distributed version actually learned *better* than the monolithic version in addition to learning substantially faster (albeit, there was a computational cost in terms of efficiency). An interesting followup experiment would be to increase the beam width of the monolithic RL so that the same number of processor-seconds were spent and compare the results of the learning.

On the infant mortality data, we ran CorPRL on a data set of 3000 examples divided over 3 machines (plus a fourth for rule review). The invariant-partitioning property was also observed for these runs when compared with a run of RL on a set of the same 3000 examples.

Related Work

Little work has been done to scale up machine-learning techniques so that they are effective for the analysis of large problems. Machine learners typically search a space of generalizations of the data, which can be time consuming. Even "fast" learners slow down linearly in the number of examples used for training. However, databases are getting ever larger and scientists are becoming more and more interested in machine-learning techniques. These two factors make the scaling up of machine-learning techniques essential.

Catlett's work [Catlett, 1991a] addresses the problem that in building decision trees, most of the time is spent sorting the values of numeric attributes. He shows that in some cases, continuous attributes can be discretized without a significant loss in accuracy of the resultant concept description [Catlett, 1991b]. Windowing [Quinlan, 1986], the process of selecting (non-randomly) subsets of the data from which to learn, has been shown to be effective in reducing the run time in noise-free domains, but it can increase run time when noise is present [Wirth & Catlett, 1988]. Catlett also studied peepholing [Catlett, 1992], the process of using subsets of the data for selecting attributes at each choice point in a decision tree. Peepholing can reduce the run time, because the number of values that must be sorted at each decision point is reduced. The run time appears (empirically) to be reduced from a quadratic function of the number of examples to a near-linear function. He also shows that peepholing can reduce run times by an order of magnitude even with small numbers of examples. He predicts two orders of magnitude with 100,000 examples [Catlett, 1992].

Even with these approximations, the run time of the learners is still linear in the number of examples, so learning with very large data sets can still be prohibitively expensive. Catlett had projected that learning with a million examples (on NASA data) would require several months of CPU time [Catlett, 1991a]. Even with a speedup of two orders of magnitude, the learning time would still be counted in days. This prediction is consistent with other results on learning in a purely symbolic domain [Provost & Aronis, 1994]. In such a domain, the problem of sorting numeric values is avoided, and the run time is, theoretically, a linear function of the number of examples. Unfortunately, when run on a single workstation, swapping becomes a major issue. Parallelism can be used to help scale up to large problems either in combination with the above approaches, or by itself. Note that the above techniques introduce further approximations into the learning process. Musik et al. [Musik, Catlett, & Russell, 1993] introduce a technique for making sampling decisions in a principled way, based on the expected error and the computational cost of making it.

Very little work addresses using parallelism to scale up learning to large problems. Similar to the present work, Chan and Stolfo [Chan & Stolfo, 1993a; Chan & Stolfo, 1993b] study scaling up using a coarse-grained parallelization across the data set. They divide the data among several processors, each of which learns a concept description from its subset. The learned concept descriptions are then coalesced to form a composite concept description. The method of composition is to form an "arbiter tree," which arbitrates among the predictions of the several concept descriptions to determine a final prediction. Not unexpectedly, degraded performance (as compared to learning with the entire data set) can result from such techniques. Unlike our approach, these methods do not allow for cooperation between processors as learning progresses. The main benefit of this independence is that different learning algorithms can be incorporated easily, since only the predictions of the learned concept descriptions are shared.

There has also been little work on using massively parallel architectures to take advantage of the inherent parallelism in machine learning algorithms for scaling up to large problems. Cook and Holder [Cook & Holder, 1990] used the CM-2 Connection Machine to parallelize a perceptron algorithm [Rumelhart, Hinton, & Williams, 1986] and AQ [Michalski, Mozetic, Hong, & Lavrac, 1986]. Zhang [Zhang, Mckenna, Mesirox, & Waltz, 1989] also utilized the massive parallelism of the CM-2 in a parallelization of a backpropagation neural network. None of these approaches directly addresses the problem of massive data sets. Provost and Aronis [Provost & Aronis, 1994] have been very successful in scaling up learning techniques to large datasets using massive parallelism. We offer complementary techniques to investigators for whom distributed networks of workstations are more readily available than massively parallel supercomputers.

Other work in Artificial Intelligence has addressed the parallelization of heuristic search, for tasks other than learning. Several researchers have implemented heuristic search routines (IDA*) on massively parallel architectures with impressive results [Cook & Lyons, ; Mahanti & Daniels, 1993; Powley, Ferguson, & Korf, 1993]. In this work, the search problem is parallelized with respect to portions of the search tree; subtrees are given to the processors, each of which performs a heuristic search. For a particularly hard problem (12.6 billion nodes), Powley, et al., estimate the run times on state-of-the-art workstations to be between one-half and 2 days, as compared to 25 minutes on the CM-2 Connection Machine. This yields a real-time speedup of between 30 and 100.

Previous work has also dealt with search on MIMD (multiple instruction multiple data) machines; for example, Rao and Kumar discuss parallel depth-first search [Kumar & Rao, 1987; Rao & Kumar, 1987]. As in the work above, a key to effective parallel search on MIMD machines is load balancing--the dynamic distribution of work when processors become idle. In the present work, we try to provide the distributed processors with roughly equal learning tasks at the outset. Future work will consider more elaborate load balancing schemes.

Discussion

We demonstrate a powerful yet practical approach to the use of parallel processing for addressing the computational complexity problems of machine learning on very large data sets. CorPRL does not require the use of expensive, highly specialized, massively parallel hardware. Rather, it takes advantage of more readily available, conventional hardware making it more broadly applicable to the scientific and engineering communities.

Furthermore, CorPRL does not suffer from the loss of accuracy that approximation and data set sampling techniques do. In fact, when the subprocesses are given the same inductive bias as a monolithic learner, CorPRL not only learns substantially faster but also learns better. Although the invariant-partitioning property may not hold for every possible evaluation criterion, it has been both theoretically and experimentally demonstrated to work for real-world scientific data analysis problems. Even for those evaluation functions where the invariant-partitioning property does not hold, simple modifications may allow the system to address these problems successfully.

To date, we have only explored the distribution of a homogeneous set of ML programs (all identical copies of RL) using the same inductive bias over a fairly homogeneous set of machines (UNIX workstations and a few IBM-PC workstations). This work indicates that the most basic implementation of a distributed learning system can be very effective in scaling up to very large data sets. Future directions include the extension of the system to subprocesses that utilize multiple learning biases, subprocesses that run on machines with widely different processing capabilities, and the load balancing issues associated with heterogeneous processing. In addition, the cooperation between processors in the present work was limited to the rule review. Preliminary results indicate that partial results can be used to scale down the amount of search necessary by each subprocess, based on the amount of learning left (by the system as a whole) [Provost, 1992]. Other forms of cooperation can be envisaged; work on cooperating distributed processing has shown great benefits of the cooperation of heterogeneous agents [Clearwater, Huberman, & Hogg, 1991].

Acknowledgements

We would like to thank John Aronis for useful discussions, Drs. Sharma and Mattison from the University of Pittsburgh School of Public Health, Dr. Rosenberg from University of Pittsburgh Department of Biology, and the W.M. Keck Foundation.

References

Buchanan, B., & Mitchell, T. (1978). Model-directed Learning of Production Rules. In D.A.Waterman & F. Hayes-Roth (ed.), *Pattern Directed Inference Systems*, p. 297-312. New York: Academic Press.

Catlett, J. (1991a). Megainduction: a Test Flight. In *Proceedings of the Eighth International Workshop on Machine Learning*, p. 596-599. Morgan Kaufmann.

Catlett, J. (1991b). On changing continuous attributes into ordered discrete attributes. In Y. Kodratoff (ed.), *Proceedings of the European Working Session on Learning*. Springer-Verlag.

Catlett, J. (1992). Peepholing: choosing attributes efficiently for megainduction. In D. Sleeman & P. Edwards (ed.), *Proceedings of the Ninth International Conference on Machine Learning*, p. 49-54. Morgan Kaufmann.

Chan, P., & Stolfo, S. (1993a). Meta-Learning for Multistrategy and Parallel Learning. In *Proceedings of the Second International Workshop on Multistrategy Learning*.

Chan, P., & Stolfo, S. (1993b). Toward Parallel and Distributed Learning by Meta-Learning. In *Proceedings of the AAAI-93 Workshop on Knowledge Discovery in Databases*.

Clearwater, S., Huberman, B., & Hogg, T. (1991). Cooperative Solution of Constraint Satisfaction Problems. *Science*, **254**(1991), p. 1181-1183.

Clearwater, S., & Provost, F. (1990). RL4: A Tool for Knowledge-Based Induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, p. 24-30. IEEE C.S. Press.

Cook, D., & Holder, L. (1990). Accelerated Learning on the Connection Machine. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, p. 448-454.

Cook, D., & Lyons, G. (to appear). Massively Parallel IDA* Search. *Artificial Intelligence Tools*.

Danyluk, A. P., & Provost, F. J. (1993). Small Disjuncts in Action: Learning to Diagnose Errors in the Telephone Network Local Loop. In *Proceedings of the Tenth International Conference on Machine Learning (ML-93)*. Morgan Kaufmann.

Gilliland, G. C. (1987). A biological macromolecule crystallization database: a basis for a crystallization strategy. In R. Geige, A. Ducruix, J. C. Fontecilla-Camps, S. Feigelson, R. Kern, & A. McPherson (ed.), *Crystal Growth of Biological Macromolecules*, North Holland.

Kumar, V., & Rao, V. (1987). Parallel depth-first search, Part II: analysis. *International Journal of Parallel Programming*, **16**(6), p. 501-519.

Mahanti, A., & Daniels, C. (1993). A SIMD approach to parallel heuristic search. *Artificial Intelligence*, **60**, p. 243-282.

Michalski, R., Mozetic, I., Hong, J., & Lavrac, N. (1986). The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, p. 1041-1045. AAAI-Press.

Musik, R., Catlett, J., & Russell, S. (1993). Decision Theoretic Subsampling for Induction on Large Databases. In *Proceedings of the Tenth International Conference on Machine Learning*, p. 212-219. Morgan Kaufmann.

Powley, C., Ferguson, C., & Korf, R. (1993). Depth-first heuristic search on a SIMD machine. *Artificial Intelligence*, **60**, p. 199-242.

Provost, F. J. (1992). *Policies for the Selection of Bias in Inductive Machine Learning*. Ph.D. Thesis, University of Pittsburgh.

Provost, F. J., & Aronis, J. (1994). *Scaling up inductive learning with massive parallelism*. (Technical Report ISL-94-3). Intelligent Systems Laboratory, University of Pittsburgh.

Provost, F. J., Buchanan, B. G., Clearwater, S. H., Lee, Y., & Leng, B. (1993). *Machine Learning in the Service of Exploratory Science and Engineering: A Case Study of the RL Induction Program* (Technical Report ISL-93-6). Intelligent Systems Laboratory, University of Pittsburgh.

Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, 1, p. 81-106.

Rao, V., & Kumar, V. (1987). Parallel depth-first search, Part I: Implementation. *International Journal of Parallel Programming*, 16(6), p. 479-499.

Rendell, L., & Seshu, R. (1991). Learning Hard Concepts through Constructive Induction: Framework and Rationale. *Computational Intelligence*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (ed.), *Parallel Distributed Processing*, Cambridge, MA: MIT Press.

Wirth, J., & Catlett, J. (1988). Costs and Benefits of Windowing in ID3. In J. Laird (ed.), *Proceedings of the Fifth International Conference on Machine Learning.*. Morgan Kaufmamn.

Zhang, X., Mckenna, M., Mesirox, J., & Waltz, D. (1989). *An efficient implementation of the backpropagation algorithm on the connection machine CM-2* (Technical Report RL89-1). Thinking Machines Corp.