Distributed Data Mining Workshop at International Conference of Knowledge Discovery and Data Mining , 1998

Combining Decision Trees Learned in Parallel

Lawrence O. Hall, Nitesh Chawla and Kevin W. Bowyer
Department of Computer Science and Engineering, ENB 118
University of South Florida
4202 E. Fowler Ave.
Tampa, Fl 33620
hall@csee.usf.edu

ABSTRACT

Very large data sets may be utilized for visualization. To focus attention on the salient regions of a data set being visualized, it is useful to have information on the interesting regions of data. It is possible to learn the salience of regions of data but very slow, if possible, to do so serially on currently available terabyte plus datasets. This paper describes an approach in which decision trees can be learned in parallel from disjoint subsets of a complete data set. The learned decision trees are converted to rules and the rules are combined into a single rule set. The combination process is based on an approach, suggested in Williams 1990 dissertation, in which rules that match one or more examples but assign them to different classes are resolved. Similar rules are also combined into more general rules. An alternate approach to combining the rule sets based on work of Provost and Hennessy 1996 is also discussed. Results on two small data sets indicate the decision tree to rules with rule conflict resolution approach has promise.

Introduction

Electronic databases are growing quite large. Applying data mining to a very large set of examples from a database is potentially quite time consuming. The number of data records may overwhelm a computer system's memory making the process of learning very slow. Datasets used for visualization may be very large. Users attempting to determine salient or interesting aspects of a data set to be visualized may only want to visit salient subsets. The concept of salient may be learned from examples, but the example sets are likely to be very large. For some visualization tasks up to a terabyte of examples may be collected (Kegelmeyer 1998). An approach to speeding up the learning when the training data set is very large is to parallelize the machine learning approach so that data and calculation are distributed over many processors and memories. This paper examines an approach to learning concepts utilizing parallel processing.

Different representations of concepts may be learned from a set of labeled data such as, neural networks, rules, and decision trees (Mitchell 1997). Decision tree learning (Quinlan 1992; Breiman *et al.* 1984) is reasonably fast and accurate. Our approach to learning on large data sets is to parallelize the process of learning by utilizing decision trees. It is straightforward to reduce a decision tree to rules and the final representation used in this research consists of a rule base created from decision trees.

The strategy pursued here is to break a large data set into n disjoint partitions, then learn a decision tree on each of the n partitions in parallel. A decision tree will be grown on each of n processors independently. After growing the n decision trees, they must be combined in some way. In work by Chan and Stolfo (Chan & Stolfo 1996; 1997) the decision trees are combined using meta-learning. The decision trees remain individual trees and new examples are run through all or a subset of the trees with a classification decision made based on some meta-rules for combining the outputs of individual tree classifiers. Domingos (Domingos 1997) builds n individual trees on overlapping subsets of the original data set. These trees are used to classify some generated examples which are added to the original training set and an individual tree is grown on the augmented training set. This approach produces accurate, stable trees but makes the training set larger.

Provost and Hennessy (Provost & Hennessy 1996) introduce an approach to learning and combining rules on disjoint subsets of a full training data that is quite effective. A rule based learning algorithm is used to generate rules on each subset of the training data. As a rule is generated, if it is "satisfactory" it is passed on for evaluation on the other data sets. All rules that are "satisfactory" on the full data set are retained and theorems show that these rules will be a superset of the rules generated when learning is done on the full training set.

Our goal is to have a single decision system after learning is done independently on n disjoint subsets of data. The independent learners can be viewed as agents learning a little about a domain with the knowledge of each agent to be combined into one knowledge base. Towards this end the independent decision trees might be combined into a single decision tree. However, there are significant complexities in attempting

such an approach. In our approach, decision trees at each of n nodes will be converted to rules and the rules will then be combined into a single rule set, as first described by Williams (Williams 1990). This single rule set will then be used to classify unseen examples. At the present time we focus on classification domains in which all attributes are continuous. The work is directly extendible for domains with mixed nominal and continuous attribute types in any combination.

The rest of this paper consists of four sections. Section 2 is a discussion of building the decision trees and converting a tree to a set of rules. Section 3 discusses how to combine rule sets. Section 4 contains experimental results on two small data sets. Finally, Section 5 is a summary of the current work and future directions.

Decision trees to rules

At each node in a decision tree an attribute must be chosen to split the node's examples into subsets. In this paper, we only consider the case of continuous attributes. There are different measures (Breiman *et al.* 1984; Mingers 1989b; Quinlan 1992) which can be applied to determine how good a particular split is for an attribute. Continuous attribute splits are typically of the form $Attribute_1 \leq X$ or $Attribute_1 > X$. We have used C4.5 (Quinlan 1992) release 8 (Quinlan 1996) in building decision trees.

Consider a continuous attribute A which takes on N distinct values (e.g. for A=3, A=5, A=7, N=3). If the attribute values are sorted then there are N-1 possible split thresholds at $t = (v_i + v_{i+1})/2$, where v_i is a value of attribute A and $v_i < v_j | i < j$ so the values are in sorted order. If one allows only binary splits then every threshold provides unique subsets K_1 and K_2 of the examples at node K. The ability to choose the threshold t to maximize the splitting criterion favors continuous attributes with many distinct values (Quinlan 1996).

The choice of a particular threshold for splitting is found as follows (Quinlan 1996). Let C denote the number of classes and p(K,j) the proportion of cases at node K which belong to the jth class. The information at node K is

$$Info(K) = -\sum_{j=1}^{C} p(K,j) \times log_2(p(K,j)). \quad (1)$$

The information gained by a test T with L outcomes (L=2 for binary splits of continuous attributes) is

$$Gain(K,T) = Info(K) - \sum_{i=1}^{L} \frac{|K_i|}{|K|} \times info(K_i). \quad (2)$$

The information gained by a test is strongly affected by the number of outcomes (i.e. is biased towards cases with many outcomes, becoming maximal when there is just 1 case in each subset K_i). Hence, Quinlan uses the gain ratio criterion (Quinlan 1992; 1996) to select among attributes. However, for only continuous attributes with binary splits the information gain suffices. The bias towards continuous attributes with many distinct values is overcome by adding a penalty term to the Gain which is the ratio of the number of distinct values at node K to the number of examples at K. The threshold ranking value (TRV) at node K is then

$$TRV = GAIN(K,T) - log_2(N-2)/|K|. \quad (3)$$

The TRV is used to choose the splitting threshold for a continuous attribute A. The attribute with the highest TRV value and its associated split will be used in the decision tree.

Quinlan has shown that selecting continuous splits in this way produces compact and accurate trees (Quinlan 1996) when compared with the gain ratio criterion.

The second aspect of creating a final decision tree is pruning the tree to remove nodes that do not add accuracy and thereby reduce tree size. Pruning is likely to be very important for large training set which will produce large trees. There are a number of methods to prune a decision tree (Mingers 1989a; Oates & Jensen 1997). In C4.5 an approach called pessimistic pruning (Quinlan 1992) is implemented. This approach to pruning is very useful for small data sets as it does not require a separate test set for the pruning process. Pessimistic pruning is quite fast and has been shown to provide trees that perform adequately (Mingers 1989a; Quinlan 1992). However, it is forced to use an estimate of error at any node in a decision tree which is not clearly sound.

It has been shown that error complexity or cost complexity pruning of decision trees yields small and accurate trees (Mingers 1989a; Oates & Jensen 1997). This approach requires a separate pruning test set which should be easily available in the case of large datasets of labeled examples. The error complexity approach involves creating and evaluating all possible pruned subtrees from the initial decision tree which may prove quite costly on large decision trees.

A less time consuming method which appears to result in accurate trees of reasonable size (Mingers 1989a) is reduced error pruning (Quinlan 1987b). This approach also requires a separate test set. It is less time consuming than error complexity pruning since it considers only reductions of the tree which reduce error on the pruning test set. However, reduced error pruning results in larger trees than error complexity pruning, which can be an issue for large datasets.

Recently, Oates and Jensen (Oates & Jensen 1997; 1998) have shown that for large data sets it can be the case that tree size will increase with the number of training examples while the accuracy of the tree is not affected by adding training examples. They used C4.5 release 5 (which does not use a penalty term for continuous attribute splits) and tested several pruning algorithms. They found that only error complexity pruning was (in some cases) able to keep tree size in check when there was no increase in accuracy with additional training examples. We found that the trees were much smaller using C4.5 rel. 8 and that for the Australian data set (Oates & Jensen 1998;

Merz & Murphy) using pessimistic pruning accuracy was still slightly growing as tree size grew. However, the trend of larger trees with more training examples and no increase in accuracy pointed out in their papers is of concern.

Figure 1 shows a decision tree turned into a set of rules by simply following paths to leaves with simplifications of removing subsumed conditions. The rules can be created from pruned or unpruned trees. Rules can be pruned separately from trees. An approach included with C4.5 (Quinlan 1992) to pruning rules is so time intensive (Kufrin 1997; Oates & Jensen 1998) that it may also require parallelization for large training set sizes. Rule pruning does not necessarily fix the problem of larger training sets giving no increase in accuracy over smaller training sets but larger rule sets (Oates & Jensen 1998).

We are experimenting with the generation of rules from pruned trees. The simple experiments reported here discuss results from pruned and unpruned decision trees.

Creating a merged rule set

A decision tree will be learned from each of n disjoint subsets of a complete set training data. Each of these n trees may be learned in parallel and rules may then be generated from them. These rules will be combined into one rule set. In the proceeding, we assume that two rulesets at a time are combined. To combine n rulesets, approximately $log_2(n)$ combinations will be necessary with rule sets that have been conflict resolved into 1 set being further resolved until all rule sets have been combined.

Rules can be combined by simply taking the merge of the n rule sets into a new rule set. However, there may be rules that conflict. That is, two rules may match a specific training example, but put the example into different classes (Williams 1990). These conflicting rules must be resolved. There may also be rules which have the same number of conditions and put examples in the same class, but have different values for the conditional tests. These rules can be merged into one rule.

Our approach to rule conflict resolution, partially described in (Hall, Chawla, & Bowyer 1998) begins with Williams' basic approach (Williams 1990), where multiple decision trees, each with a different bias (e.g. choose a nominal attribute over a continuous attribute for node splitting in the case of a tie in utility), were generated from the same training data set. Rules were generated from the different trees and then combined into a single rule set.

Two rule sets will be combined on a processor, call it A, on which one of the two rule sets was created. Hence, some of the examples used in creating the two rule sets are locally available. Processor A will be passed the rules to be merged, but not the data the rules were created with. Every created rule has a index into a list of the examples that it covers. In the case of two rules having overlapping antecedent conditions and different right hand sides (classes), processor A will request the relevant training examples from the remote processor on which the conflicting rule was created. Processor A must pass the rule identifier so that the proper examples may be indexed and returned. The returned training examples together with the local training examples covered by the rule created on processor A make up a conflict set of examples used in step 2 in the proceeding.

The first step in conflict resolution is to "scope" continuous attributes by finding all rule pairs which

- have the same number of antecedent conditions,
- have one or more attributes that are the same but the continuous value chosen for the test is different (e.g. length ≤ 5 and length ≤ 5.7),
- the continuous values differ by no more than 60% of the value of the lower (this is user settable and in place in case of large gaps in the data), and
- classify examples into the same class.

If the attribute test is $>$ then the smaller of the two rule values is used (e.g. length > 5 and length > 8 results in length > 5 as the condition of the modified rules). If the attribute is $\leq$ then the larger of the two values is used in the modified rules.

The second step is to find the conflicting rules. First, identify all pairs of rules that have all but one condition the same and have different classes on the right hand side. These rules are considered to be in conflict. These conflicts are resolved as described in (Hall, Chawla, & Bowyer 1998; Williams 1990). As the training sets used on each processor are disjoint, unlike (Williams 1990) there are other types of rule conflicts that may occur. These conflicts occur in rules where the number of conditions may be unequal and not all conditions may match. For example, two rules could have no conditions in common and put examples in different classes. We do not consider this case here as we believe it is unlikely to occur if the training sets contain similar distributions of examples from a coherent larger training set.

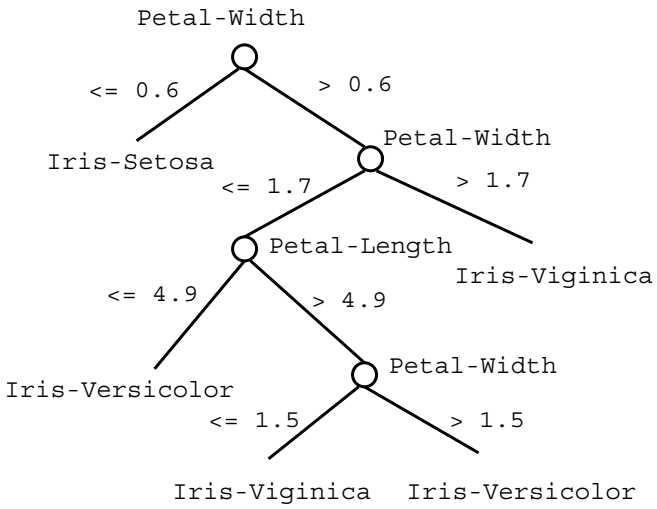
An example that has been observed in our experiments involves two rules in which conditions partially overlap. For example consider the rules R1, R2 and R3

```
R1: If petalwidthincm <= 0.6
    then Iris-setosa

R2: If petalwidthincm > 0.5
       petalwidthincm <= 1.7
    then Iris-versicolor

R3: If petalwithincm > 1.5
    then Iris-viginica.
```

Rules R1 and R2 conflict as do the rule pair R2 and R3. Here, 1 condition overlaps in both sets of conflicting rules. In general, we have $m > 0$ overlapping conditions. The set of examples covered by the "conflict rules" can be reduced to a set of examples in conflict (that match both rules). Then a condition (or more than one) may be adjusted such that the resultant modified rules make the minimum number of misclassifications on the conflict set of examples. Currently, we




```
R1: If Petal-Width <= 0.6 --> Iris-Setosa
R2: If 0.6 < Petal-Width <= 1.7 and Petal-Length <= 4.9 --> Iris-Versicolor
R3: If Petal-Width > 1.7 --> Iris-Viginica
R4: If 0.6 < Petal-Width <= 1.5 and Petal-Length > 4.9 --> Iris-Viginica
R5: If 1.5 < Petal-Width <= 1.7 and Petal-Length > 4.9 --> Iris-Versicolor
```

Figure 1: The C4.5 tree produced on the full Iris dataset and the corresponding rules.

adjust just one condition. For example, R1 no longer conflicts its test is adjusted to be **petalwidthcm** $\leq .5$.

A more complex problem is a condition in one rule overlaps with an entire interval from 2 conditions in another rule, as shown in R4 and R6 below. Now, we will strengthen R4 but this will rule out some examples (for instance add the condition petalwidthincm $\leq$ 1.500). Now a new rule is needed to cover lost examples, **unless** another rule for the class Iris-viginica covers the lost examples. Here we have R5 which does cover them. It is in conflict with R6 though and this will be resolved as above.

```
R4: If petallengthincm > 4.900
       petalwidthincm > 0.400
    then class Iris-viginica

R5: If petalwidthincm > 1.500
    then  class Iris-viginica

R6: If petallengthincm > 4.900
       petalwidthincm > 1.500
       petalwidthincm <= 1.700
    then class Iris-versicolor
```

If R5 did not exist the new rule:

```
nr: If petallengthincm > 4.900
       petalwidthincm > 1.700
    then class Iris-viginica
```

must be created to join the strengthened R4 call it R4s.

```
R4s: If petallengthincm > 4.900
       petalwidthincm > 0.400
       petalwidthincm <= 1.5
    then class Iris-viginica
```

To resolve the above conflicts rules from different classes must be checked for overlapping conditions and no conditions which are mutually exclusive. The conflicts are then resolved as discussed.

When Step 2 finds no new conflicts, go back and repeat Step 1. Then merge the two rule sets together and eliminate any redundant rules that have been created by the process of removing conflicts.

Experimental results

Simple initial experiments to test the feasibility of this approach were done on two data sets. The Iris data (Fisher 1936; Merz & Murphy) which has 4 continuous valued attributes and classifies 150 examples as one of 3 classes of Iris plant. The second is the Pima Indians Diabetes data set (Merz & Murphy) which has 8 numeric attributes and classifies 768 examples into one of 2 classes. We have done an experiment simulating a parallel 2-processor implementation for both data sets and a 3-processor implementation for the Iris data. Our results are an average of a 10-fold cross-validation. The 10-fold cross validation was done by breaking the data into 10 train/test sets. For the Iris data and the 2-processor experiment the breakdown is 135 train/15 test examples in each fold, so that the test sets were mutually exclusive. Then the training data was split in the middle into 2 subsets of 67 and 68 examples. For each fold 2 decision trees were generated one on each subset, rules were generated, the conflicts among rules were resolved and the rules were merged into one set. Finally, the resultant rule set was used to classify the

Table 1: Results on the Iris data set using 10-fold cross-validation for a 2 processor partition. SD - standard deviation.

C4.5 % Correct ± sd	Unpruned % Correct ± sd	Pruned % Correct ± sd
95.3 ± 6.01	96 ± 5.33	96 ± 5.33

Table 2: Results on the Iris data set using 10-fold cross-validation for a 3 processor partition. SD - standard deviation.

C4.5 % Correct ± sd	Unpruned % Correct ± sd	Pruned % Correct ± sd
95.3 ± 6.01	96.67 ± 5.37	96.67 ± 5.37

15 test examples for each fold. The diabetes data set is handled in a similar manner.

The 3 processor experiment with the Iris data set meant that the 135 training examples of each fold were broken into 3 training sets of size 45 each. Three trees were built with rules generated and then 2 rule sets were combined into one. The combined rule set was then combined with the remaining unmodified rule set to provide the final set of rules for testing.

The classification accuracy when generating rules from the unpruned and pruned trees for the 2 processor simulation with the Iris data is shown on the first row of results in Table 1 and compared with the accuracy when one decision tree is generated from each fold. The accuracy is slightly better than that of the C4.5 decision trees for both the pruned an unpruned trees. On this data set the pruned and unpruned rules are the same. The default C4.5 parameters were used with one exception. Since no pruning was done with the default parameters, the certainty factor was changed from 25 to 1. With the lowered certainty factor pruning is done on only 4 of the decision trees generated and in every case on a maximum of 1 of the 2 decision trees generated from the original 135 example training set. However, after merging the generated rules the final rule sets are the same as when rules are created from the unpruned tree. On average there was 1 conflicting pair of rules resolved per fold. The average number of rules was 7 which is more than c4.5 which results in an average of 4.9 rules.

The results from the 3 processor simulation for the Iris data are shown in Table 2. In this case the rules make 1 less error than in the 2 processor experiment and 2 less errors than C4.5. The average number of rules is 9.

The results from 10-fold cross-validation on the Diabetes data set for a 2 processor implementation are shown in Table 3. The average number of rules obtained from C4.5 are 28.7 from the unpruned tree compared to 40.8 via our approach and 23.8 for the pruned trees compared to 34.1 for our approach. In this small example, the accuracy is comparable with a cost of larger rule sets on average. On average 2 or 3 conflicting rules were resolved in each fold.

Table 3: Results on the Pima Indian Diabetes data set using 10-fold cross-validation. sd - standard deviation.

C4.5 % Correct ± sd	Unpruned % Correct ± sd	Pruned % Correct ± sd
73.33 ± 4.66	74.94 ± 2.93	73.64 ± 4.12

Summary and discussion

In the approach to learning from large training sets discussed here, a data set is broken into n disjoint subsets. A decision tree is generated on each of the n subsets and rules are generated from the decision tree. The rule sets will then be combined into a single rule set with conflicts among rules resolved. This approach might also be used by agents which learn rules from examples and then want to share knowledge. Initial tests on the Iris and Diabetes data sets are promising. The cross-validated results are the same as or better than those obtained using C4.5.

We intend to investigate an alternate way of combining rule sets which would use the approach of Provost and Hennessy (Provost & Hennessy 1996). Each rule created from a decision tree may be evaluated by the certainty factor suggested by Quinlan (Quinlan 1987a) normalized for skewed distributions:

$$f(r, E) = \frac{(TP - 0.5)}{TP + \rho FP)}, \tag{4}$$

where r is the rule being evaluated, E is a training data set, TP is the number of true positives, FP is the number of false positives, and ρ is the ratio of positive examples to negative examples in the training set. A rule is considered "satisfactory" if $f(r, E) \geq c$ for some threshold c. Any satisfactory rule created from an individual decision will be further evaluated on the data used to create all n-1 other decision trees (for n separate decision trees). If it remains satisfactory, it will be retained. For conflicting rules, at least one of them will certainly not be found satisfactory.

Currently, we are testing on several larger datasets using more partitions of the data. We also plan to conduct experiments on the DOE's "ASCI Red" parallel computing system (San 1997).

Acknowledgements: This research was partially supported by the United States Department of Energy through the Sandia National Laboratories LDRD program, contract number DE-AC04-76DO00789.

References

Breiman, L.; Friedman, J.; Olshen, R.; and Stone, P. 1984. *Classification and Regression Trees.* Belmont, CA.: Wadsworth International Group.

Chan, P., and Stolfo, S. 1996. Sharing learned models among remote database partitions by local meta-learning. In *Proceedings Second International Conference on Knowledge Discovery and Data Mining*, 2–7.

Chan, P., and Stolfo, S. 1997. On the accuracy of meta-learning for scalable data mining. *Journal of Intelligent Information Systems* 8:5–28.

Domingos, P. 1997. Knowledge acquisition from examples via multiple models. In *International Confernce on Machine Learning.*

Fisher, R. 1936. The use of multiple measurements in taxonomic problems. *Ann. Eugenics* 7.

Hall, L.; Chawla, N.; and Bowyer, K. 1998. Decision tree learning on very large data sets. In *International Conference on Systems, Man and Cybernetics.*

Kegelmeyer, W. 1998. Avatar. Technical report, Sandia National Labs, http://www.ca.sandia.gov/avatar.

Kufrin, R. 1997. Generating c4.5 production rules in parallel. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 565–570.

Merz, C., and Murphy, P. *UCI Repository of Machine Learning*
Databases. Univ. of CA., Dept. of CIS, Irvine, CA. http://www.ics.uci.edu/˜ mlearn/MLRepository.html.

Mingers, J. 1989a. An empirical comparison of pruning methods for decision tree induction. *Machine Learning* 4(2):227–243.

Mingers, J. 1989b. An empirical comparison of selection methods for decision tree induction. *Machine Learning* 3(4):565–570.

Mitchell, T. 1997. *Machine Learning.* N.Y.: McGraw-Hill.

Oates, T., and Jensen, D. 1997. The effects of training set size on decision tree complexity. In *Proceedings of the 14th International Conference on Machine Learning*, 254–262.

Oates, T., and Jensen, D. 1998. Large datasets lead to overly complex models: an explanation and a solution. In *KDD'98.* Preprint Univ. Mass. Amherst, Paper to appear.

Provost, F., and Hennessy, D. 1996. Scaling up: Distributed machine learning with cooperation. In *Proceedings of AAAI'96*, 74–79.

Quinlan, J. 1987a. Generating production rules from decision trees. In *Proceedings of IJCAI-87*, 304–307.

Quinlan, J. 1987b. Simplifying decision trees. *International Journal of Man-Machine Studies* 27:227–248.

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

Quinlan, J. 1996. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research* 4:77–90.

Sandia National Labs, http://www.sandia.gov/ASCI/Red/UserGuide.htm. 1997. *ASCI Red Users Manual.*

Williams, G. 1990. *Inducing and Combining Multiple Decision Trees.* Ph.D. Dissertation, Australian National University, Canberra, Australia.