Decision Tree Learning on Very Large Data Sets

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

Consider a labeled data set of 1 terabyte in size. A salient subset might depend upon the users interests. Clearly, browsing such a large data set to find interesting areas would be very time consuming. An intelligent agent which, for a given class of user, could provide hints on areas of the data that might interest the user would be very useful. Given large data sets having categories of salience for different user classes attached to the data in them, these labeled sets of data can be used to train a decision tree to label unseen data examples with a category of salience. The training set will be much larger than usual. This paper describes an approach to generating the rules for an agent from a large training set. A set of decision trees are built in parallel on tractable size training data sets which are a subset of the original data. Each learned decision tree will be reduced to a set of rules, conflicting rules resolved and the resultant rules merged into one set. Results from cross validation experiments on a data set suggest this approach may be effectively applied to large sets of data.

1. 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 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 [6]. 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 [10]. Decision tree learning [15, 2] 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 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 [3, 4] 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.

Our goal is to have a single decision system after learning is done independently on the n 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 [17] into a single rule set. 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 a small data set. Finally, Section 5 is a summary of the current work and future directions.

2. 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 [2, 8, 15] 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 [15] release 8 [16] 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 [16].

The choice of a particular threshold for splitting is found as follows [16]. 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 [15, 16] 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 [16] 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 [9, 12]. In C4.5 an approach called pessimistic pruning [15] 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 [9, 15]. 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 [9, 12]. 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 sub-trees 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 [9] is reduced error pruning [14]. 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 [12, 13] 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 [13, 11] using pessimistic pruning accuracy was still slightly growing as tree size grew. However, the trend of larger trees with more train 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 [15] to pruning rules is so time intensive [7, 13] that it may also require parallelization for large train set sizes. Rule pruning does not necessarily fix the problem of larger train sets giving no increase in accuracy over smaller training sets but larger rule sets [13].

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

3. Combining rule sets

Under the paradigm explored in this paper, a single training data set will be broken into n subsets. A decision tree will be learned from each of these n subsets in parallel. Rules will then be generated from the decision trees. These rules will be combined into one rule set. In the proceeding, we assume that two rulesets at a time are combined.

Rules can be combined by simply taking the merge of the n rules sets into a new rule set. However, there may be rules that conflict. That is, two rules may match a specific train example, but put the example into different classes [17]. 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 is based on Williams work [17], 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 data set. Rules were generated from the different trees and then combined into a single rule set.

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

- have the same number of antecedent conditions and
- 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).

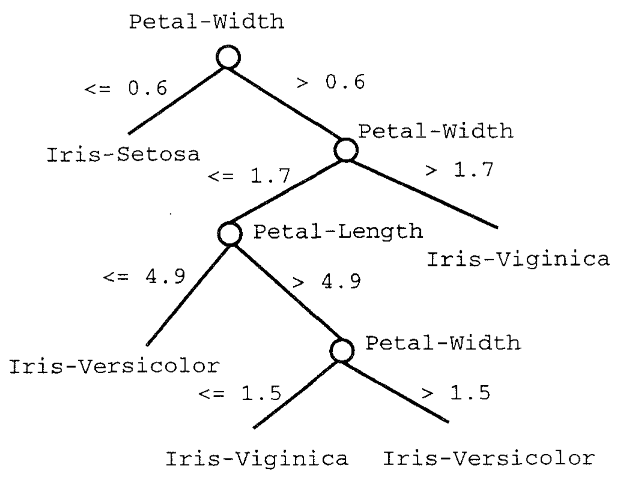
If the attribute test is $>$ then the smaller of the two rule values is used (e.g. length > 5 and length > 8 results in length > 5 as the condition of the combined rule). If the attribute is $\leq$ then the larger of the two values is used in the combined rule.

The second step is to 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. Assume the following two conflicting rules
r1: p1 > 0.6 & p4 $> 4.9 \rightarrow$ class1 and
r2: p1 > 0.6 & p2 $\leq 1 \rightarrow$ class2.
Each rule will be strengthened by adding the negation of the conflicting condition to the rule. The negation of the condition, p2 ≤ 1, is p2 > 1, making the new rule 1,
nr1: p1 > 0.6 & p4 > 4.9 & p2 $> 1 \rightarrow$ class1.
Similarly the new rule 2 is
nr2:p1 > 0.6 & p2 ≤ 1 & p4 $\leq 4.9 \rightarrow$ class2.

Now examples that have attribute values p1 > 0.6 & p4 > 4.9 & p2 ≤ 1 triggered one (or both) of the two original rules (r1 and r2) before they were strengthened, but no longer do so. Hence, the next step is to restore the lost example coverage. If all the "conflict" examples that match the conditions above are of one class (e.g. class1), then introduce the rule: nrc1: p1 > 0.6 & p4 > 4.9 & p2 $\leq 1 \rightarrow$ class1. If the examples that match these conditions are of mixed classes find a new condition (for continuous attributes an attribute and split point) that partitions the examples into pure subsets. If one cannot be found, take the best test and add it to the rules. There will now be two new rules for these conflicting examples:
nrc1: p1 > 0.6 & p4 > 4.9 & condn $\rightarrow$ class1,
nrc2: p1 > 0.6 & p4 > 4.9 & $\neg$ condn $\rightarrow$ class2.

Next eliminate any redundant conditions in individual rules. For example in rule rr: p4 > 0.6 & p3 > 5 & p4 $> 1.4 \rightarrow$ class1, the condition p4 > 0.6 is redundant since it is subsumed by the last condition. Now repeat Step 2 to find if new conflicts have been introduced.

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.



```
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.

The final rules will be ordered by their accuracy taken from the original tree in all cases except for conflict resolution rules for which the accuracy is calculated on the conflict set. In the case of ties in accuracy, the most specific rules will be put first. The rule firing system fires the first rule it encounters in lexical order. A default rule of the dominant class is used.

There is one exception to the above ordering and it is with the rule created from the first split in a decision tree when that split leads to a leaf. That very general rule will be placed first, because more specific rules from another rule set could misclassify some of the examples classified by the rule.

4. Experimental results

Simple initial experiments to test the feasibility of this approach were done on the Iris data [5, 11] which has four continuous valued attributes and classifies examples as one of three classes of Iris plant. There are 150 examples in the Iris data. We have done an experiment simulating a 2-processor parallel implementation. 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 of 135 train/15 test examples, so that the test sets were mutually exclusive. Then the train 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 15 examples for each fold.

The classification accuracy when generating rules from the unpruned and pruned trees 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 matches 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 train set. However, after merging the generated rules the final rule sets are the same as when rules are created from the unpruned tree.

A variation of the experiment was done in which the rules were ordered by specificity (i.e. those with the most

Table 1. Results on the Iris data set using 10-fold cross-validation. With rule ordering by accuracy and by specificity.

Rule Order	C4.5 % Correct	Unpruned % Correct	Pruned % Correct
Accuracy	95.3%	95.3%	95.3%
Specificity		96%	96%

conditions in the antecedent were put first) with the rule accuracy used to order rules of the same specificity. This was done because rule accuracy is not recalculated after conflict resolution and is only an approximation of the accuracy on the train set. As shown in the second row of Table 1 one less error was made with the specificity ordering approach.

5. Summary

In the approach to learning from large training sets discussed here, a data set is broken into n 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 data set are promising. The cross-validated results are the same as or better than those obtained using C4.5. Currently, we are testing on larger datasets using more partitions of the data. We also plan to conduct experiments on the DOE's "ASCI Red" parallel computing system [1].

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

[1] ASCI Red Users Manual, http://www.sandia.gov/ASCI/Red/UserGuide.htm, 1997.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, P.J. Stone, Classification and Regression Trees, Wadsworth International Group, Belmont, CA. 1984.

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

[4] P. Chan and S. Stolfo, "On the accuracy of Meta-Learning for Scalable Data Mining", *Journal of Intelligent Information Systems*, V. 8, pp. 5-28, 1997.

[5] R.A. Fisher, "The use of multiple measurements in taxonomic problems", *Ann. Eugenics*, V. 7, pp. 179-188, 1936.

[6] W.P. Kegelmeyer, AVATAR, http://www.ca.sandia.gov/avatar/, 1998.

[7] R. Kufrin, "Generating C4.5 Production Rules In Parallel", *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, Providence, RI, pp. 565-570, July 1997.

[8] J. Mingers, "An Empirical Comparison of selection methods for decision tree induction", *Machine Learning*, 3 (4), pp. 319-342, 1989.

[9] J. Mingers, "An Empirical Comparison of pruning methods for decision tree induction", *Machine Learning*, 4 (2), pp. 227-243, 1989.

[10] T.M. Mitchell, Machine Learning, McGraw-Hill, N.Y., 1997.

[11] C.J. Merz and P.M. Murphy, "UCI Repository of Machine Learning Databases", Univ. of CA., Dept. of CIS, Irvine, CA., Machine readable data repository, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[12] T. Oates and D. Jensen, "The Effects of Training Set Size on Decision Tree Complexity", *Proceedings of the 14th International Conference on Machine Learning*, pp. 254-262, 1997.

[13] T. Oates and D. Jensen, "Large Datasets Lead to overly Complex Models: an Explanation and a Solution", Preprint Univ. Mass. Amherst, submitted to Knowledge Discovery and Data Mining, 1998.

[14] J.R. Quinlan, "Simplifying Decision Trees", *International Journal of Man-Machine Studies*, V. 27, pp. 227-248. 1987.

[15] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA.

[16] J.R. Quinlan, "Improved Use of Continuous Attributes in C4.5", Journal of Artificial Intelligence Research, V. 4, pp. 77-90, 1996.

[17] G.J. Williams, "Inducing and Combining Multiple Decision Trees", PhD Thesis, Australian National University, Canberra, Australia, 1990.