Parallel Learning Using Decision Trees: A Novel Approach

Sattar Hashemi[1], Mohammad. R. Kangavari[1]

[1] Department of Computer Engineering,
Iran University of Science and Technology
Tehran , Iran

Abstract. Decision trees are one of the most effective and widely used induction methods that have received a great deal of attention over the past twenty years. When decision tree induction algorithms used with uncertain rather than deterministic data, the result is complete tree, which can classify most of the unseen samples correctly. This tree would be pruned in order to reduce its classification error and over-fitting.

Recently, parallel decision tree researches concentrated on dealing with large databases in reasonable amount of time. In this paper we present new parallel learning methods that are able to induce a decision tree from some overlapping partitioned training set. Our methods are based on combination of multiple induction methods; each one is running on different processor. These methods have been developed based on Kramer and fuzzy mode to control and combine the result of learning methods in order to generate the final tree. Experimental results show that if the attributes and classes in training set have uniform distribution and the size of training set are not too small, these methods result statistically lower error rate in comparison to existing methods.

1. Introduction

Decision tree learning program have received a great deal of attention over the past twenty years in the field of machine learning and KDD (knowledge discovery in databases) [11]. Several factors contribute to their popularity: Decision tree learning programs are fast and effective [2], They work remarkably well with no tweaking of parameters, which has facilitated their wide use in the comparison of different algorithms. Decision trees also work comparatively well with very large datasets [13], with large number of variables, and with mixed type data (continuous, nominal, Boolean, etc.). These qualities result in part from the simple yet powerful divide-and-conquer algorithm underlying decision tree learners, and in part from the high quality software packages that have been available for learning decision trees (most notably, CART [1] and C4.5 [3]).

However most parallel learning research has concentrated on dealing with large datasets in reasonable a mount of time [7],[10], multiple models research involves learning a set of classifiers, an ensemble, and then combining their classification or evidence in order to make more accurate classifications [4],[8],[12]. The research typically compares the error rate of the ensemble on test examples to the error rate of single model learned on the same data.

Our goal is to have a single decision tree (named PDT) after learning is done independently on three overlap subset of data. The independent learners can be viewed as agents and the knowledge of each agent is combined into one knowledge base. Towards this end the independent decision tree might share information in each node to make single decision tree. However, there are significant complexities in attempting such an approach. In PDT, three processor at each node share information about the selected attribute in that node, then the most informative one is chosen for next development of tree using two novel methods, named Kramer and Fuzzy, Then tree is pruned using EBP [5], and final pruned tree will be used to classify unseen examples.

The rest of this paper consists of six sections. Section 2 is a discussion of building the decision trees, section 3 analyzes the tree pruning methods, section 4 discusses how to combine decision trees, and section 5 contains experimental result on some known datasets. Finally, section 6 is a conclusion of current work and future directions.

2. Decision Tree Growing

There are a number of alternative suggestions for measure to be used in selecting attributes for growing a tree. Three of them that are used in our model as follow:

Gain ratio measure (GR) [9]. The default splitting criterion used by C4.5 is gain ratio, an information based measure that takes into account different numbers (and different probabilities) of test outcome. Let that C denote the number of classes and p(D, j) the proportion of cases in D that belong to the J_{th} class. The residual uncertainty about the class to which a case in D belongs can be expressed as

$$Info(D) = -\sum_{j=1}^{C} p(D, j) * Log_2(P(D, j)) \quad (1)$$

And the corresponding information gained by a test T with k outcomes as

$$Gain(D,T) = Info(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} * Info(D_i) \quad (2)$$

The information gained by a test is strongly affected by the number of outcome and is maximal when there is one case is each subset D_i. On the other hand, the potential information obtained by partitioning a set of cases is based on knowing the subset D_i into which a case falls; this split information

$$Split(D,T) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} * Log_2\left(\frac{|D_i|}{|D|}\right) \quad (3)$$

Tend to increase with the number of test outcomes. The gain ratio criterion assesses the desirability of a test as the ratio of its information gain to its split information. The gain ratio of every possible test is determined and, among those with at least average gain, the split with maximum gain ratio is selected.

The Chi Square statistic (c^2) [9]. Hart and Mingers have employed another measure to select among attributes the Chi Square statistic. This is the traditional statistic for measuring the association between two variables in a contingency table. It compares the observed frequencies with frequencies that one would expect if there were no association between the variables. The resulting statistic is distributed approximately as the chi-square distribution, with larger values indicating greater

association. In these experiments Yates' correction is used for 2*2 tables. The basic equation for this function is:

$$c^2 = \sum\sum \frac{(x_{ij} - E_{ij})^2}{E_{ij}} \quad (4)$$

where $E_{ij} = x_i x_j / N$, i.e., the expected value for each cell in contingency table.

One level Look Ahead (Look) [6]. It searches top-down to find more significant attribute than the root attribute, and then pull more significant attribute up to the root to obtain a tree with fewer number of nodes. Quinlan's method finds the most significant attribute only, but this method takes not only the attribute that Quinlan's method finds but also those attributes that are found in next level into consideration. It compares those attributes to find the most significant attribute and place the newly found most significant attribute at that node. For sake of simplicity we assume that each attribute has two possible values. The algorithm can be generalized with more than two possible values in each attribute. We also assume that each node in the decision tree is labeled by the attribute p_i, which is used to partition the dataset at that node.

(1) Build a decision tree by Quinlan's method, name the tree T_1, and count the number of descendants. Name the root of T_1 as p_1, the left child of p_1 as p_2 and the right child of p_1 as p_3.
(2) Build a new decision tree T_2 by using the attribute p_2 as the initial attribute at the root node of T_2 to partition the dataset and also count the number of descendants.
(3) Repeat step (2) for p_3 and name it T_3.
(4) Compare the numbers of descendants of T_1, T_2, and T_3, and take the tree T* which has the least number of nodes among those three trees.
(5) Apply this algorithm to immediate children of the root of T*. (Then the algorithm will be applied throughout the tree recursively.)

3. Decision Tree Pruning

Decision trees that are grown using one of the methods discussed in last section can become cumbersomely large for several reason. One possible cause is noise, when examples have a large amount of feature noise (i.e. erroneous feature values) or class noise (i.e. mislabeled class value), the induction algorithm may expand the tree too far based on irrelevant distinctions between examples. Noise can cause some irrelevant features to be included among the selected tests. This leads to trees that overfit the training example and that do poorly in front of unseen samples.

The goal of pruning a tree is to eliminate tests that were dictated by noisy data. Two categories of pruning methods are usually distinguished: include pre-pruning and post-pruning techniques. Pre-pruning techniques decide eliminate tree nodes and branches during the initial construction of the tree. Post-pruning is the more popular simplification method. The input to a post-pruning (or simply pruning) algorithm is an unpruned tree T, and its output is a pruned tree T', formed by removing one or more subtree from T. Post-pruning technique do not usually search all possible T'; instead , they rely on heuristics to guide their search.

Some experimental result was shown that there is no one pruning method that did best for all the dataset, but in relative terms, the "error based pruning" used in the

standard C4.5 algorithm produced consistently good results. For such results, this approach is used for PDT pruning.

Error Base Pruning (EBP). This is the pruning method implemented in C4.5 [3], [5], one of the learning systems that we employed in our experiments for building the trees. This method use information in the training set for building and simplifying trees.

EBP visits the nodes of T_{max} (fully expanded tree) according to a button-up post-order traversal strategy instead of a top-down strategy. The true novelty is that EBP simplifies a decision tree T by grafting a branch T_t (subtree of node t) onto the place of the parent of t itself, in addition to pruning nodes (see fig. 1).

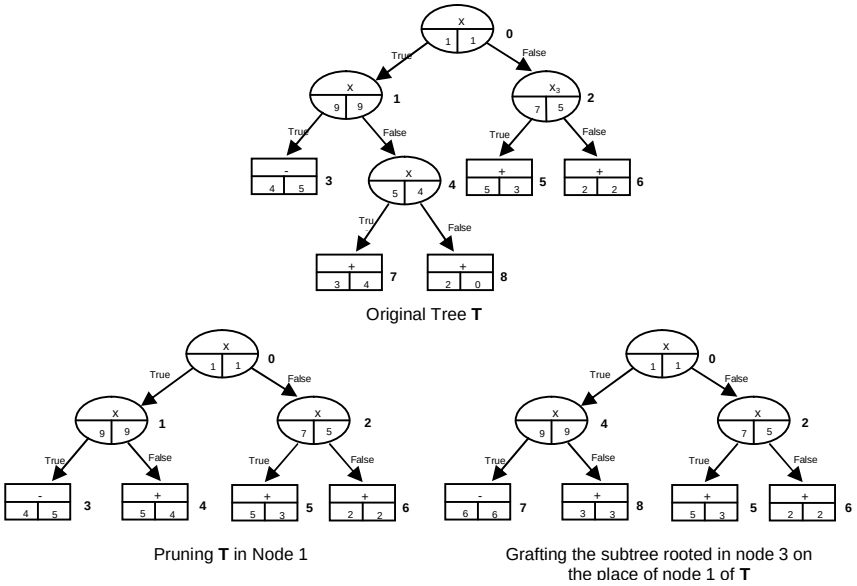

Fig. 1. Decision tree pruning using EBP

The sum of the predicted error rates of all the leaves in a branch T_t is considered to be an estimate of the error rate of the branch itself. Thus, by comparing the predicted error rate for t with that of the branch T_t and of the largest sub branch T_t, rooted in a child t' of t, we can decide whether it is convenient to prune T_t, to graft $T_{t'}$ onto the place of t or to keep T_t.

4. Parallel Decision Tree

In this section, we develop parallel formulations (PDT) for the classification decision tree construction and hybrid schemes that select good features of datasets. In our parallel formulation, Original dataset has been randomly split into two subsets: training set (90 percent) and test set (10 percent), and the training set has been divided into three overlap subsets (with 10 percent overlap). Each of these subsets is assigned to one processor. An overview of PDT is shown in fig. 2.

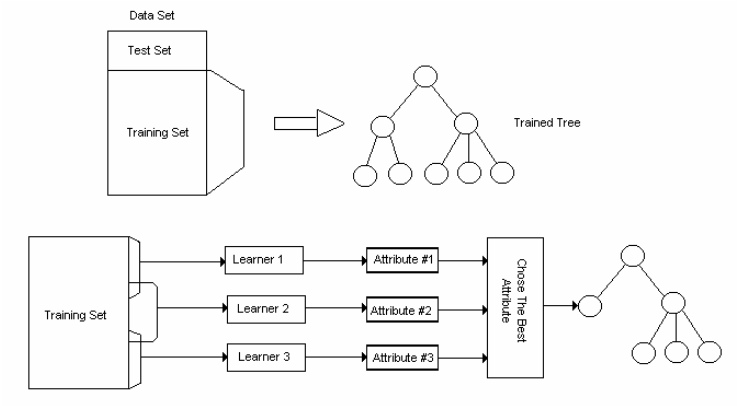

Fig. 2. The overviews of PDT approach.

Major steps for PDT construction are given as below:

1) Select a node to expand according to a decision tree expansion strategy (e.g. Depth-first or Breadth-first), and call that node as the current node. At the beginning, root node is selected as the current node.
2) Assign the subsets of the local data at the current node to each of three processors.
3) Simultaneously, the GR, c^2, and the Look measures of each attribute at each processor are computed and best three attributes are selected.
4) Among three selected attributes, the most informative one is chosen by our approaches Kramer and Fuzzy as follow.
5) Depending on the branching factor of the tree desired, create child nodes for the same number of partitions of attribute values, and split training cases accordingly.
6) Repeat above steps (1-5) until no more nodes are available for the expansion.
7) Prune the final tree using EBP method.

Now we explored two alternatives for combining classifications or evidence of each individual processor in order to make the final PDT tree.

Kramer Method. The attribute proposed by each processor is noted and the most frequent one is used for the ensemble expansion. For this operation at least two proposed attributes must be the same. If different attributes are suggested by each processor, we have defined a relation function that estimated the similarity of each pair of attributes, such as A and B. Let N be the total number of sample in current node and p, r the row and column of contingency table respectively, the Kramer relation of these two attribute is computed as follow

$$R(A,B) = \sqrt{\frac{c^2}{N * Min(r-1, d-1)}} \tag{5}$$

This relation is computed for all three attributes, and the attribute with highest Kramer relation with two other attributes is selected to develop the tree. Note that relation between two attribute such as A and B $R(A,B)$ is a positive number

below one that explores the dependency of these two nominal attribute and is transposition i.e. $R(A,B) = R(B,A)$.

Fuzzy Method. Another method for combination of these attribute is Fuzzy method. This method define $R(A,B)$ as a matrix which rows equal to distinct values of attribute A and columns are equal to number of distinct values of B, and define as

$$R(A,B) = R(A,C) * R(C,B) \quad (6)$$

Where C is total classes $\{C_1, C_2, ..., C_k\}$ and * denote matrix product. Therefore this relation is not transposition i.e. $R(A,B) \neq R(B,A)$ because the product of two matrixes is not transpositive. Suppose that A have i distinct values, then $R(A,C)$ is a $i \times k$ matrix that $R(i,k)$ element represent the frequency of samples in current node with V_i value that belong to class C_i. These elements are divided by number of total samples to be normalized. Therefore all elements of this matrix are real numbers between 0 and 1.

Now we present a method for computing the cardinal of a matrix such as $|R(A,B)|$, and among these attribute the attribute with highest cardinality is chosen to expand the tree. For this purpose, first a vertical projection of $R(A,B)$ is computed and define a one dimensional matrix $p[1....i]$, then a cut point that empirically achieved $1/k^2$ (k is number of classes of all samples) is applied to map p matrix to a binary matrix. Elements which their values are higher than a map to one and the others are map to zero. The sum of theses elements represents the cardinality of p matrix. Despite the Kramer method, this method can chose an attribute between two same attributes that are suggested by two processors and don't need to vote among them.

5. Experimental Results

Simple initial experiments to test the feasibility of PDT were done on datasets that are taken form UCI repository [14]. These datasets contain some natural and artificial domains. The experiments have been done with parallel 3-processor simulation that each of these processors runs one heuristic over the corresponding subset. The results are average of 10-fold cross-validations. The 10-fold cross-validation was done by breaking the data into 10 train/test sets, so that the test sets were mutually exclusive. For each train fold three classification trees were generated, one on each of three overlapped subsets, and among selected attributes in each subset, the best one was chosen by our proposed heuristic to grow the PDT. Finally, the PDT was pruned and used to classify the unseen example of each test fold.

Table 1. Error Rate of different methods

	C4.5	*Kramer Method*	*Fuzzy Method*
Iris	5.3	5.3	5.2
Voting	6.3	5.7	4.8
Hypo	0.54	0.49	0.415
Led-1000	27	26.1	26.5
Led-200	32.7	34.2	38
Golf	28.2	37	35
Glass	35	38	37.5

The classification error of final pruned PDT over datasets are shown in table 1 and compared with the error rate of C4.5. The default C4.5 parameters were used. Our experimental results showed that the proposed attributes in the first rounds of learning process unusually are the same, therefore in the first steps tree growing was fast because of low computational cost, but in middle toward bottom of tree these attributes were different, and tree growing in this manner became slow. Most of variant that are produce by PDT in bottom of tree is pruned by pruning method. In some noisy datasets fuzzy method have better performance because if two processor select same noisy attribute, Kramer method select this attribute by voting, but it is not the case for Fuzzy method, and this method have higher exploration in dataset.

Size of dataset is another important factor in our approach, as seen in the table, although in Hypo and Led_1000, PDT emerged better performance than C4.5, in Led-200 and Golf its error rate increased due to partitioning of dataset and loosing comprehensibility. The other reason for good performance of PDT in artificial domain is capability of Look Measure in artificial domains. Voting dataset is similar to Led dataset but base error in this dataset is low (most of attributes that are suggested in each node are similar) thus PDT that used Fuzzy method achieved better performance because of its exploration power. Although Iris dataset is not so huge but PDT represent acceptable accuracy because samples and classes in this dataset are distributed uniformly, however none uniform distribution in Glass dataset results unbalanced trees and reduction in accuracy. Another important feature of PDT in comparison to other parallel learner that produce many trees, is capability to reform to decision rule to be better understandable and used in expert system.

6. Conclusion

Parallel approach in learning from the examples that discussed in this paper, broke training set into three overlap subsets, each of these subsets evaluated by an individual heuristic (processor), and then the best attribute is selected for expanding the tree. A tree that constructed base on PDT, gives better performance as compared to using the single (C4.5) growing methods when data set is not too small and classes and attributes in dataset are distributed uniformly.

These results provide empirical validation (for trees) of the widely held belief that the multiple models approach is able to do better than the single model approaches when the learned models make uncorrelated errors.

References

1. Breiman, L., Friedman, J. H., Olsen, R.A., & Stone, C.J. Classification and Regression Trees, Wadsworth international group, 1984.
2. *Mingers*, J. An empirical comparison of selection measures for decision-tree induction. Machine learning 3: 319-342, 1989.
3. Quinlan, J.R. Constructing Decision Tree in C4.5: Programs for Machine Learning, pp.17-26, Morgan Kaufman Publishers, 1993.
4. Kamal Ali, On explaining degree of error reduction due to combining multiple decision trees. IBM Almaden research center, CA, 95120, 1996
5. Floriana Esposito, Donato Malebra, and Giovanni Seminar, A comparative analysis of methods for pruning decision trees, IEEE transaction on pattern analysis and machine intelligence, vol. 19, no. 5, 1997.
6. *K.V.* Sreerama, On growing better decision trees from data, PhD thesis, Johns Hopkins University, 1997.
7. Anurag Srivastava , Eui-Hong Han, and Vipin Kumar, Parallel formulation of decision tree classification algorithms, Information technology lab, Hitachi America, 1998.
8. Provost, F., & Kolluri, A survey of methods for scaling up inductive algorithms, Data mining and knowledge discovery, 3(2), 131-169, 1999.
9. Lim,T.-J.,Loh,W.Y.,& Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine learning, 40(3), 203-228, 2000.
10. S. Orlando, P.Palmerini, R. Perego, F.Silverstri, Scheduling High Performance Data Mining Task on a Data Grid Environments, proceedings of Euro-par 2002
11. Gerald Benoit, DATA MINING, Annual review of information science and technology, 2002, B.Cronin, ed., in press.
12. C.mastroianni , D.Tailia, P.Trunfio, Managing heterogeneous resource in data mining application on grid using XML-Based metadata, proceeding of IPDPS 2003, IEEE computer society press, April 2003. A
13. Shu-Tzu Tasi, chao-Tung Yang, Decision tree construction for data mining on grid computing, IEEE conference on e-technology, e-commerce and e-service. 2004.
14. Merz, C., and Murphy, P. UCI Repository of Machine Learning Databases, University of California, Dept. of CIS, Irvine, CA. http://www.ics.uci.edu/~mlearn/MLRepository.html. 2004.