Enhanced C4.5 Algorithm for Decision Support System

Amrita R. Agrawal (Kedia)
amritaramkishan@gmail.com

International conference on COMPUTING, BUSINESS APPLICATIONS and LEGAL ISSUES
IMT Gaziabad, UP, India. 3rd and 4th March 2011

Abstract: Every organization collects a large amount of data regarding customer profiles, business transactions, market interests and other valuable information. Data mining searches large stores of data for patterns and delivers results that can be utilized either in an automated decision support system or assessed by human analysts. Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object to determine appropriate classification according to decision tree rules.

C4.5 is among the most popular algorithm in decision tree construction. Here, we are suggesting steps to develop a strong decision tree construction algorithm based on C4.5.

Key Words: Decision tree, Decision support System, C4.5, Data mining

I. INTRODUCTION

A. Decision support systems (DSS): DSS are a computer-based information system that supports business or organizational decision-making activities. DSSs serve the management, operations, and planning levels of an organization and help to make decisions, which may be rapidly changing and not easily specified in advance.

B. A decision tree (DT): DT is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Often times these techniques were originally developed for statisticians to automate the process of determining which fields in their database were actually useful or correlated with the particular problem that they were trying to understand. Decision trees are commonly used for gaining information for the purpose of decision making. ‘Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Decision tree learning is one of the most widely used and practical methods for inductive inference’.

C. C4.5 algorithm: C4.5 is an improvement of IDE3 algorithm, developed by Quinlan Ross (1993). It is based on Hunt’s algorithm and also like IDE3, it is serially implemented. Pruning takes place in C4.5 by replacing the internal node with a leaf node thereby reducing the error rate. Like IDE3 the data is sorted at every node of the tree in order to determine the best splitting attribute. It uses gain ratio impurity method to evaluate the splitting attribute (Quinlan, 1993).

II. LITERATURE SURVEY

A. DT Algorithm Issues

1. Choosing splitting attributes: Which attributes to use for splitting attributes impacts the performance applying the built DT Ordering of splitting attributes: The order in which the attributes are chosen is important.

2. According to the ordering of attributes, requiring unnecessary comparisons.
3. **Splits**: The number of splits is obviously based on the domain. If the domain is continuous or has a large number of values, the number of splits to use is not easily determined.

4. **Tree Structure**: To improve the performance of applying the tree, an appropriate tree structure should be decided upon.

5. **Stopping Criteria**: The creation of the tree definitely stops when the training data are perfectly classified, but there is a trade-off between accuracy of classification and performance.

6. **Training Data**: The DT structure depends on the training data:
   a. If training data set is too small, there will not be enough to work with.
   b. If training data set is too large, it may lead to overfitting.

7. **Pruning**: Remove redundant comparisons or remove subtrees to achieve better performance.

### B. Decision Tree Construction Algorithms

Various DT algorithms are available. Most popular of them are as follows:

1. IDE3
2. SPRINT
3. CART
4. SLIQ
5. C4.5

#### C. Why C4.5? Studies show that [14]

- **SPRINT** classifier has the highest classification accuracy among all the classifiers, this is followed by C4.5.
- The class size, attribute number and record number do not affect the classification accuracy of SPRINT and C4.5 compared to other classifiers. The classification accuracy of the IDE3 and CART classifiers depends to a large extent the class size, attribute number and record number of the data sets.
- The classification accuracy of IDE3 is better than that of CART as ID3 has a high accuracy for large data that have been pre-processed (noise and outliers removed) and loaded into the memory at the same time. But that are not too large (small-medium data sets), the classification accuracy of CART is more than that of IDE3.
- SPRINT and C4.5 algorithms have a good classification accuracy compared to other algorithms used in the study.

Therefore, we can say that, irrespective of the class size, number of attributes and records of the data set volume, categorical and continuous attributes, C4.5 always behave very close to the best Decision tree Construction strategy.

### III. PROPOSED METHODOLOGY

#### A. Apply Basic C4.5 Algorithm:

Let the training data is a set \( S = s_1, s_2, \ldots \) of already classified samples. Each sample \( s_i = x_1, x_2, \ldots \) is a vector where \( x_1, x_2, \ldots \) represent attributes or features of the sample. The training data is augmented with a vector \( C = c_1, c_2, \ldots \) where \( c_1, c_2, \ldots \) represent the class to which each sample belongs.

In pseudo code the algorithm is:

1. check for base cases
2. for each attribute \( a \)
3. find the normalized information gain from splitting on \( a \)
4. let \( a_{\text{best}} \) be the attribute with the highest normalized information gain
5. create a decision node that splits on \( a_{\text{best}} \)
6. recurse on the sublists obtained by splitting on \( a_{\text{best}} \), and add those nodes as children of node
Gain ratio:
\[
\text{Gain Ratio} (D, t) = \frac{\text{Gain} (D, t)}{\text{SplitInfo} (D, t)}
\]

B. Handling Missing Values: The results of the simulation study with Twala[15] show that the proportion of missing data, the missing data mechanism, the pattern of missing values, and the design of database characteristics (especially the type of attributes) all have effects on the performance of any MDT. The effects of missing data have been found to adversely affect DT learning and classification performance, and this effect is positively correlated with the increasing fractions of missing data. From the eight current techniques investigated that Expectation Maximization with Multiple imputation (EMMI) is the overall best method for handling both incomplete training and test data. The EM algorithm [Dempster et al., 1977] is used to approximate a probability function (p.f. or p.d.f.). EM is typically used to compute maximum likelihood estimates given incomplete samples. Using this algorithm, find out the missing value of the attribute.

C. Handling Continuous valued attributes: The process of making ranges for continuous valued attributes can be summarized as below:
1. Sort the attribute data based on attribute value
2. Consider the largest value as initial threshold and note down its associated result.
3. For each decreasing attribute value; keep the same threshold as long as the Result is of same type
4. If the Result differs; reset the threshold & repeat

D. Pruning: Lior Rokach, Oded Z. Maimon[20] says that, several studies compare the performance of different pruning techniques (Quinlan, Mingers and Esposito). The results indicate that some methods (such as Cost complexity Pruning, Reduced error Pruning) tends to over-pruning, i.e. creating smaller but less accurate decision trees. Other methods (like Error Based Pruning, Pessimistic Error Pruning and Minimum Error Pruning) bias towards under-pruning. Compared to these approaches, algorithm for optimal pruning of decision trees proposed by Hussein Almuallim [21] gives better result.

Algorithm: The reduction in error which results from expanding a leaf is viewed as “profit”, while the associated increment in the size of the tree is viewed as “cost”. Now, given a decision tree DT that we would like to prune, consider generating another tree, T, by doing the following:
- Add a new node, to, as the parent of ti (the root of DT). This new node, thus, becomes the root of T.
- For each node ti in T, 1 ≤ i ≤ n, let \( \text{cost}(t_i) = d(t_i) - I \) and let \( \text{COST}(t_0) = 1 \)
- For each node ti in T, 1 ≤ i ≤ n, let \( \text{profit}(t_i) = \text{error}(t_i) - \sum \text{error}(a) \) and let \( \text{profit}(t_0) = 0 \).
  - a: child of I
- Delete all the leaves of DT.
We will call the tree T constructed as above the cost-profit (CP) tree of DT.

For a predecessor-closed subtree T’, let us denote by \( \text{cost}(T’) \), the total cost, and by \( \text{profit}(T’) \), the total profit of the nodes of T’. If \( DT’ \) is the pruned decision tree which corresponds to T’, then
- \( \text{cost}(T’) = s(DT’), \)
- \( \text{profit}(T’) = \text{base-error} - \text{error}(DT’). \)

The time and space complexities of OPT-2 are both O(nC), where n is the number of test nodes in the initial decision tree, and C is the number of leaves in the target (pruned) decision tree.

IV. CONCLUSION
In this research, we have proposed to improve a rule-base classification algorithm C4.5. The main objective of this research is to boost up the classification accuracy and simultaneously roll back timing to build a classification model. We aim to continue this research by analyzing the data files. We will find out why it is performing better in proposed method.

V. REFERENCES
[2] J. Han, M. Kamber “Data Mining” Morgan Kaufmann Publication


[10] Lecture notes in Data Mining By Michael W. Berry, Murray Browne


[20] Lior Rokach, Oded Z. Maimon, Data mining with decision trees: theory and applications