ML: A Multiple Learner Approach for Data Mining with Diverse Decision Tree Inducers

Dipak V. Patil¹ and R. S. Bichkar

¹Asst. Prof., Department of Computer Engineering Sandip Institute of Technology and Research Centre Nashik, M.S., India
E-mail: dipakvpatil@hotmail.com

²Professor, Department of E &TCG.H. Raisoni College of Engineering & Management, Pune, M.S., India
E-mail: bichkar@yahoo.com

ABSTRACT: The decision tree learning has received considerable attention in areas like data mining, pattern recognition, and machine learning. Distinct researchers have developed many distinct inducers. They have proposed solutions to different problems like feature selection, splitting criteria, pruning, handling noise etc. The performance measure for decision tree is the competitive classification accuracy. A Multiple learner approach utilizes potential advantages of available decision tree learner. We have experimented on several decision tree inducers that include the standard C4.5, evolutionary decision trees and the oblique decision tree. The proposed algorithm selects the most accurate classifier amongst all that offers the highest classification performance of these decision tree inducers on particular data set. The decision trees induced using various techniques and following Multiple Learner (ML) algorithm were observed to have higher competitive performance for classification accuracy on test data.

Keywords: Decision tree inducer, Multiple Learner, classification accuracy, and optimization

1. INTRODUCTION

A decision tree is a predictive model that is used in data mining and machine learning. A decision tree maps observations about an item to conclusions about its target value. Depending on the nature of output there are two types of decision trees, classification tree having discrete output values and regression tree having continuous output. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications.

A number of reasons of appreciation of decision tree based classification are available in the literature [1]. Decision tree models have wider applicability. A decision tree classification is performed by a series of simple, uncomplicated tests whose semantics are by instinct clear to domain experts. The decision tree does not need backtracking and it takes polynomial time with respect to the number of attributes and inputs. The pre-classified examples are used to acquire the knowledge and alleviate the bottleneck problem of developing expert systems. The decomposition of attributes according information gain, implements better use of available attributes and also implies computational efficiency in classification. Moreover decision trees can handle the erroneous or incomplete data set.

Decision tree is a classifier in the form of tree data structure that contains decision nodes and leaves. A decision node specifies a test to be carried on single attributes value whereas a leaf specifies a classification. A solution is present for each probable outcome of the test in the form of a child node. A performance measure of a decision tree over a set of cases is called classification accuracy. It is defined as the percentage of correctly classified instances over test data.

The different decision tree algorithms address different problems and the different decision tree algorithms have different strengths and those need to be utilized.

The proposed algorithm selects the classifier with highest classification accuracy amongst the three algorithms namely C4.5 algorithm, Evolutionary Decision Tree and Oblique Decision Tree and thus taps efficiencies with different learning algorithms. In next section 2.0 we provide working of different decision tree learning algorithms. The section 3.0 explains the proposed ML algorithm and the last section provides conclusions.
2. DECISION TREE CONSTRUCTION ALGORITHM

In this section we provide working of three different decision tree learning algorithms in details.

2.1. Decision Tree Construction Algorithm (C4.5)

The C4.5 tree construction algorithm is one of the standard algorithms for induction of decision tree. The algorithm evolves a decision tree beginning from training set T, which is a set of instances. We may call the instances as tuple in database terminology. Let the classes be denoted by the set \( \{ C_1, C_2, \ldots, C_n \} \). Each tuple specifies values for set of attributes and a class. Each attribute may have continuous or discrete values. The missing values are also allowed.

A divide and conquer strategy is used in the C4.5 algorithm [2] without permitting backtracking to construct a decision tree. At each decision node the data set T is partitioned into k subsets denoted as \( T_1, T_2, \ldots, T_k \). The algorithm is applied to obtain locally best option. The algorithm proceeds as follows.

Initially the class frequency is computed for instances in set T whose class is \( C_j \). The node is a leaf, with associated class \( C_j \), the most recurrent class if T contains one or more or all instances belong to a same class \( C_j \). The weighted sum of the instances in set T whose class is \( C_j \) is defined as the classification accuracy of the leaf. The information gain of all attributes is calculated. If set T contains instances that are having two or more classifications the set T is partitioned into subsets, specifically, instances with a feature value not greater than and the instances with a feature value greater than a certain threshold value called local threshold value. For the test at the node, the feature with highest information gain is chosen. We have to apply the divide and conquer based approach of sub tree building algorithm recursively to each subset of training instances. Calculate Classification accuracy if accuracy is less than the overall majority class prune the subtree. The default pruning method for C4.5 algorithm is error based pruning.

The process of classification of data is to maximize the information gain. The maximum information leads to increase in classification accuracy [3]. The next discussion will provide basics of information gain and gain ratio as explained in [2].

2.1.1. Gain and Information Gain

Quinlan used information gain and gain ratio in decision tree algorithm. Quinlan defined that the information obtained from a message depends on its probability and is calculated in bits as \(- \log_2(P)\) bits.

Let \( T \) be any set of cases, total number of cases in T are denoted by |\( T \)| and the number of cases in S that belong to \( C_i \) are denoted by freq (\( C_i, T \)). Consider one case picked from T. The Probability \( P \) as per above definition is given by

\[
P = \frac{\text{freq} (C_i, T)}{|T|}
\]

Information Obtained from probability \( P \) is given by \(- \log_2 (P)\) bits. The summation of classes in proportion to their frequencies is performed to calculate expected information from set of cases in \( S \) pertaining to class association, is given by

\[
\text{info}(T) = \sum_{i=1}^{k} P \times \log_2(P) \text{bits}.
\]

The equation when applied to training set \( T \), \( \text{info}(T) \) provides the average information required to calculate the class of a case in \( T \). As the test \( X \) on the preferred attributes \( T_1, T_2, \ldots, T_k \) are the subsets of the partitioning produced on set \( T \). The expected information is sum over these subsets.

\[
\text{info}_i(T) = \sum_{i=1}^{k} \frac{|T_i|}{|T|}\times\text{info}(T_i)
\]

The information gained with test \( X \) by partitioning \( T \) is given by

\[
\text{Gain}(X) = \text{Info}(T) - \text{Info}(T)\]

The attribute on which test maximizes information gain is selected.

2.1.2. Gain Ratio

The gain criterion is biased towards test with many outcomes. The preconceived notion inbuilt in the Gain criterion can be rectified with a type of normalization with adjustment.

Similar to the definition of \( \text{info}(T) \), we have

\[
\text{splitinfo}(X) = -\sum_{i=1}^{k} \frac{|T_i|}{|T|} \times \log_2 \left( \frac{|T_i|}{|T|} \right)
\]
Split info($X$) calculates information generated by dividing training set $T$ into $k$ subsets on test $X$. The information gain calculates the information generated by dividing $T$.

\[
\text{gain ratio (X)} = \frac{\text{gain(X)}}{\text{split info(X)}}
\] (6)

The attribute on which test performs maximum gain ratio is selected.

2.2. Evolutionary Decision Trees

Construction of decision tree is identified as NP-complete problem that leads us to use genetic algorithms in decision tree construction. While dealing with larger, potentially huge search space genetic algorithms [4] provide global search through space in many directions simultaneously, thereby improving the probability of finding the global optimum to obtain optimal combinations of things and solutions.

Genetic algorithms [5] Goldberg 1999) combine survival of the fittest among string structures with a structured yet randomized information exchange. To improve performance, genetic algorithm efficiently uses the past information with randomized search on new search points. The offspring are evolved using the crossover and mutation technique. The chromosomes are then evaluated for a certain fitness values and the best solution is accepted while the remaining solutions are discarded. This process continues until final chromosome with best fitness value and thus is taken as the best solution of the problem.

Papagelis A. and Kalles D. [4] proposed GATree, a genetically evolved decision tree in which the genetic operator directly operate on decision tree and not on a string representations used in conventional genetic algorithms. GATree uses a flexible, comprehensive, global metric of tree quality that tries to optimize accuracy and size. To build a population of binary decision trees GAlib’s [6] tree representation has been used in GATree. Initially, a random attribute is selected from dataset and if it is nominal one of its possible values is randomly selected. In case of continuous attribute an integer value is randomly selected in permissible range. Thus, the size of the search space is reduced. An arbitrary node of a desired tree is selected and it is substituted by that node’s test-value with a new arbitrary chosen value to perform mutation at the same time when the arbitrary node is a leaf, it substitutes the installed class with a new arbitrary chosen class. Two arbitrary nodes are selected and nodes of those sub-trees are swapped to perform crossover operation. Since a Predicted value depends only on leaves, the crossover operator does not affect the decision trees consistency. The fitness function is percentage of correctly classified instances on the test data set by the decision tree.

<table>
<thead>
<tr>
<th>Training Data</th>
<th>GATree Accuracy</th>
<th>C4.5 Accuracy</th>
<th>OC1 Accuracy</th>
<th>ML Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian</td>
<td>91.30</td>
<td>90.72</td>
<td>83.80</td>
<td>91.30</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>96.42</td>
<td>96.28</td>
<td>96.72</td>
<td>96.72</td>
</tr>
<tr>
<td>Heart</td>
<td>89.60</td>
<td>87.77</td>
<td>82.67</td>
<td>89.60</td>
</tr>
<tr>
<td>Iris</td>
<td>95.33</td>
<td>97.33</td>
<td>83.34</td>
<td>97.33</td>
</tr>
<tr>
<td>Diabetes</td>
<td>76.29</td>
<td>79.34</td>
<td>75.5</td>
<td>79.34</td>
</tr>
<tr>
<td>Average</td>
<td>89.79</td>
<td>90.28</td>
<td>84.40</td>
<td>90.85</td>
</tr>
</tbody>
</table>

2.3. The Oblique Decision Tree

The Oblique decision tree is a decision tree induction system designed for applications where the instances have numeric feature values. The oblique classifier builds decision trees that contain linear combinations of one or more attributes at each internal node. The oblique decision tree partitions the space of examples with both oblique and axis-parallel hyperplanes. Decision trees can use splits that have more than one attribute at each internal node and thus it is more generally known as multivariate decision trees. The oblique decision trees conduct multivariate tests that may result in much smaller and accurate trees in some domains. The oblique hyperplanes are more complicated than finding axis-parallel partitions, demanding greater computational effort and thus are not as popular as univariate decision trees. Murthy et al. [7] extended the work of Breiman et al. [8] and proposed a system for induction of oblique decision trees in that they have developed a randomized algorithm for inducing oblique decision trees from examples. Randomization helps in many learning concepts. The algorithm is fully implemented as an
oblique decision tree induction system. The randomized hill-climbing algorithm used in OC1 is more efficient than other existing randomized oblique decision tree methods.

The OC1 uses goodness measure impurity that should be minimum. The following impurity measures can be used as per the need of the application: information gain, Gini index, twoing rule, max minority, sum minority and sum of variations. Towing rule first proposed by Breiman et al. is the default impurity measure. Murthy et al. [9] explained twoing rule as follows. The set of instances $T$ at the node about to be split contains $(n > 0)$ instances that be a member of one of $k$ categories. In the beginning this set is the complete training set. $T$ is divided into two non-overlapping subsets, $T_L$ and $T_R$, by hyperplane $H$. The impurity measure at the start checks if $T_L$ and $T_R$ are homogeneous and belongs to the same category and in that case return minimum zero impurity. The value to be computed is defined as

$$\text{Towing value} = \frac{|T_L| \cdot |T_R|}{n^2} \left( \sum_{i=1}^{k} \left( \frac{L_i}{|T_L|} - \frac{R_i}{|T_R|} \right)^2 \right)$$  

(7)

Where $|T_L|$ and $|T_R|$ represents the number of instances on the left and right of a split at node $T$, and the number of instances at node $T$ are represented by $n$. The number of instances in category $i$ on the left and right of the split are represented by $L_i$ and $R_i$ respectively. The towing value is a goodness measure rather than an impurity measure. OC1 uses the reciprocal of this value.

The randomized hill-climbing algorithm used in OC1 is more efficient than other existing randomized methods. The ability to generate oblique trees often produces very small trees compared to axis parallel methods. The oblique trees are also more accurate than axis parallel trees when the dilemma requires an oblique split. OC1 uses cost complexity pruning [10] as default pruning method. The OC1 software can be used to create both standard, axis-parallel decision trees and oblique (multivariate) trees.

3. MULTIPLE LEARNER (ML) ALGORITHMS
The proposed algorithm works as follows.

1. Construct decision trees on same data set with three algorithms namely C4.5 algorithm, Evolutionary Decision Tree and Oblique Decision Tree.
2. Compare classification accuracy of all the algorithms.
3. Select classifier with highest classification accuracy.
4. Use that classifier as hypothesis for classifying unseen examples on that data set.

4. RESULTS
We have used data sets [11] from University of California Irvine repository. Experiments were performed to explore classification accuracy on decision tree using various standard decision tree inducers namely C4.5 (WEKA)[12], GATree [4] and OC1 [7]. The k-fold cross validation method was used to obtain the results.

During experiments, the default parameter sets were used for C4.5 and OC1 algorithms. In case of GATree, the parameters were set as follows: crossover rate = 0.99, mutation rate = 0.01, stopping criterion = 100 generations, size factor $X = 1000$ (to generate smaller trees).

The algorithm selects the classifier with highest classification accuracy and it is used as final classifier a final hypothesis for classifying unseen examples. The results obtained are summarized in Table 1, which shows the accuracy on test data of the induced tree for three classifiers and Multiple Learner (ML) algorithm. The average values of accuracy and tree size are plotted in Fig.1 for quick comparison.

5. CONCLUSION
It can be observed that the average classification accuracy of Multiple Learner (ML) algorithm is higher than the average classification accuracy of single classification algorithm. Thus the results obtained by Multiple Learner (ML) algorithm are superior to those obtained by single algorithms on GATree, C4.5 and OC1 algorithms.

In this proposed algorithm we integrate potential strengths of various learning algorithms. In GATree, the genetic algorithms pursue potentially huge search space providing global search through space in many directions simultaneously, thereby improving the probability of finding the global optimum to obtain optimal combinations of things and solutions.

The Oblique decision tree induction system is designed for applications where the instances have numeric feature values. The randomized hill-climbing algorithm used in OC1 makes it efficient. A divide and
conquer strategy is used in the C4.5 algorithm without permitting backtracking obtains locally best option to construct a decision tree.

A Multiple learner approach utilizes potential advantages of available decision tree learner and thus the multiple learner algorithm obtains better-quality results.

REFERENCES


