Abstract

Decision Trees are considered to be one of the most popular approaches for representing classifiers. Researchers from various disciplines such as statistics, machine learning, pattern recognition, and Data Mining have dealt with the issue of growing a decision tree from available data. This tutorial presents updated survey of current methods for constructing decision tree classifiers in a top-down manner. This tutorial suggests a unified algorithmic framework for presenting these algorithms and describes various splitting criteria and running methodologies.

Keywords: Decision tree, Information Gain, Gini Index, Gain Ratio, Pruning, Minimum Description Length, C4.5, CART, Oblivious Decision Trees

1. DECISION TREES

A decision tree is a classifier expressed as a recursive partition of the instancespace. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called “root” that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes). In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values. In the simplest and most frequent case, each test considers a single attribute, such that the instance space is partitioned according to the attribute’s value. In the case of numeric attributes, the condition refers to a range.

![Decision Tree Presenting Response to Direct Mailing](image)

Fig 1 Decision Tree Presenting Response to Direct Mailing
Each leaf is assigned to one class representing the most appropriate target value. Alternatively, the leaf may hold a probability vector indicating the probability of the target attribute having a certain value. Instances are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path. Figure 1 describes a decision tree that reasons whether or not a potential customer will respond to a direct mailing. Internal nodes are represented as circles, whereas leaves are denoted as triangles. Note that this decision tree incorporates both nominal and numeric attributes. Given this classifier, the analyst can predict the response of a potential customer (by sorting it down the tree), and understand the behavioral characteristics of the entire potential customers population regarding direct mailing. Each node is labeled with the attribute it tests, and its branches are labeled with its corresponding values. Geometrically interpreted as a collection of hyper planes, each orthogonal to one of the axes. Naturally, decision-makers prefer less complex decision trees, since they may be considered more comprehensible. Furthermore, according to Breiman et al. (1984) the tree complexity has a crucial effect on its accuracy. The tree complexity is explicitly controlled by the stopping criteria used and the pruning method employed. Usually the tree complexity is measured by one of the following metrics: the total number of nodes, total number of leaves, tree depth and number of attributes used. Decision tree induction is closely related to rule induction. Each path from the root of a decision tree to one of its leaves can be transformed into a rule simply by conjoining the tests along the path to form the antecedent part, and taking the leaf’s class prediction as the class value. The resulting rule set can then be simplified to improve its comprehensibility to a human user, and possibly its accuracy (Quinlan, 1987).

2. Algorithmic Framework for Decision Trees

Decision tree inducers are algorithms that automatically construct a decision tree from a given dataset. Typically the goal is to find the optimal decision tree by minimizing the generalization error. However, other target functions can be also defined, for instance, minimizing the number of nodes or minimizing the average depth. Induction of an optimal decision tree from a given data is considered to be a hard task. It has been shown that finding a minimal decision tree consistent with the training set is NP-hard (Hancock et al., 1996). Moreover, it has been shown that constructing a minimal binary tree with respect to the expected number of tests required for classifying an unseen instance is NP-complete (Hyafil and Rivest, 1976). Even finding the minimal equivalent decision tree for a given decision tree (Zantema and Bodlaender, 2000) or building the optimal decision tree from decision tables is known to be NP-hard (Naumov, 1991).

The above results indicate that using optimal decision tree algorithms is feasible only in small problems. Consequently, heuristics methods are required for solving the problem. Roughly speaking, these methods can be divided into two groups: top-down and bottom-up with clear preference in the literature to the first group.
Figure 2 presents a typical algorithmic framework for top–down inducing of a decision tree each node further subdivides the training set into smaller subsets, until no split gains sufficient splitting measure or a stopping criteria is satisfied.

3. **Univariate Splitting Criteria**

3.1 **Overview**

In most of the cases, the discrete splitting functions are Univariate. Univariate means that an internal node is split according to the value of a single attribute. Consequently, the inducer searches for the best attribute upon which to split. There are various Univariate criteria. These criteria can be characterized in different ways, such as:

i. According to the origin of the measure: information theory, dependence and distance.

ii. According to the measure structure: impurity based criteria, normalize impurity based criteria and Binary criteria. The following section describes the most common criteria in the literature.

3.2 **Impurity-based Criteria**

Given a random variable $x$ with $k$ discrete values, distributed according to $P = (p_1, p_2, \ldots, p_k)$, an impurity measure is a function $\phi : [0, 1]^k \to \mathbb{R}$ that satisfies the following conditions:

- $\phi(P) \geq 0$
- $\phi(P)$ is minimum if $\exists i$ such that component $p_i = 1$.
- $\phi(P)$ is maximum if $\forall i, 1 \leq i \leq k, p_i = 1/k$.
- $\phi(P)$ is symmetric with respect to components of $P$.
- $\phi(P)$ is smooth (differentiable everywhere) in its range.

---

**Figure 2** Top-Down Algorithmic Frameworks for Decision Trees Induction

```plaintext
TreeGrowing(S, A, y)
Where:
S - Training Set  A - Input Feature Set  y - Target Feature
Create a new tree $T$ with a single root node
IF One of the Stopping Criteria is fulfilled THEN
    Mark the root node in $T$ as a leaf with the most common value of $y$ in $S$ as a label
ELSE
    Find a discrete function $f(A)$ of the input attributes values such that splitting
    $S$ according to $f(A)$'s outcomes $(v_1, \ldots, v_n)$ gains the best splitting metric.
    IF best splitting metric > threshold THEN
        Label $t$ with $f(A)$
        FOR each outcome $v_i$ of $f(A)$:
            Set $Subtree_i = TreeGrowing(\sigma f(A) = v_i, S, A, y)$.
            Connect the root node of $tT$ to subtree $i$ with an edge that is labeled as $v_i$
        END FOR
    ELSE
        Mark the root node in $T$ as a leaf with the most common value of $y$ in $S$ a label.
    END IF
END IF
RETURN $T$

TreePruning(S, T, y)
Where:
S - Training Set
Y - Target Feature
T - The tree to be pruned
DO
    Select a node $t$ in $T$ such that pruning it maximally improve some evaluation criteria
    IF $t \neq \phi$ then $T = pruned(T, t)$
    Until $t = \phi$
RETURN $T$
```
Note that if the probability vector has a component of 1 (the variable x getsonly one value), then the variable is defined as pure. On the other hand, if all components are equal, the level of impurity reaches maximum. Given a training set S, the probability vector of the target attribute y is defined as:

\[
P_y(S) = \left( \frac{|\sigma_y = c_1S|}{|S|}, \ldots, \frac{|\sigma_y = c_{|\text{dom}(y)|}S|}{|S|} \right)
\]  

(1)

The goodness-of-split due to discrete attribute a is defined as reduction in impurity of the target attribute after partitioning S according to the values

\[
\Delta \Phi(a, S) = \phi \left( P_y(S) \right) - \sum_{j=1}^{\text{dom}(a)} \frac{|\sigma_{a_j} = v_{a_j}S|}{|S|} \cdot \phi \left( P_y \left( \sigma_{a_j} = v_{a_j}S \right) \right)
\]

(2)

3.3 INFORMATION GAIN

Information gain is an impurity-based criterion that uses the entropy measure (origin from information theory) as the impurity measure (Quinlan, 1987).

\[
\text{Information Gain}(a, S) = \text{Entropy}(y, S) - \sum_{v_{a_j} \in \text{dom}(a)} \frac{|\sigma_{a_j} = v_{a_j}S|}{|S|} \cdot \text{Entropy}(y, \sigma_{a_j} = v_{a_j}S)
\]

(3)

Where:

\[
\text{Entropy}(y, S) = - \sum_{v_{a_j} \in \text{dom}(y)} \frac{|\sigma_{a_j} = c_jS|}{|S|} \cdot \log_2 \left( \frac{|\sigma_{a_j} = c_jS|}{|S|} \right)
\]

(4)

3.4 GINI INDEX

GiniIndex is an impurity-based criterion that measures the divergences between the probability distributions of the target attribute’s values.

\[
\text{Gini}(y, S) = 1 - \sum_{c_j \in \text{dom}(y)} \left( \frac{|\sigma_{a_j} = c_jS|}{|S|} \right)^2
\]

(5)

3.5 LIKELIHOOD-RATIO CHI-SQUARED STATISTICS

The likelihood–ratio is defined as

\[
\text{Gini Gain}(a, S) = \text{Gini}(y, S) - \sum_{v_{a_j} \in \text{dom}(a)} \frac{|\sigma_{a_j} = v_{a_j}S|}{|S|} \cdot \text{Gini}(y, \sigma_{a_j} = v_{a_j}S)
\]

(6)

This ratio is useful for measuring the statistical significance of the information gain criterion. The zero hypothesis (H0) is that the input attribute and the target attribute are conditionally independent. If H0 holds, the test statistic is distributed as A2 with degrees of freedom equal to: (dom(ai); 1) ϕ(dom(y); 1).
3.6 **DKM Criterion**

The DKM criterion is an impurity-based splitting criterion designed for binary class attributes. The impurity-based function is defined as:

$$DKM(y, S) = 2 \cdot \left( \frac{|\sigma_y = y |}{|S|} \cdot \frac{|\sigma_y = \bar{y} |}{|S|} \right)$$  \hspace{1cm} (7)

It has been theoretically proved that this criterion requires smaller trees for obtaining a certain error than other impurity-based criteria (information gain and Gini index).

3.7 **Normalized Impurity Based Criteria**

The impurity-based criterion described above is biased towards attributes with larger domain values. Namely, it prefers input attributes with many values over attributes with less values (Quinlan, 1986). For instance, an input attribute that represents the national security number will probably get the highest information gain. However, adding this attribute to a decision tree will result in a poor generalized accuracy. For that reason, it is useful to “normalize” the impurity-based measures.

3.8 **Gain Ratio**

The gain ratio “normalizes” the information gain as follows (Quinlan, 1993):

$$GainRatio(a_i, S) = \frac{Information\ Gain(a_i, S)}{Entropy(a_i, S)}$$  \hspace{1cm} (8)

3.9 **Distance Measure**

The distance measure, like the gain ratio, normalizes the impurity measure. However, it suggests normalizing it in a different way:

$$
\frac{\Delta \Phi(a_i, S)}{\sum_{k,j \in dom(a_j)} \sum_{y \in dom(y)} \frac{|\sigma_{a_k=1 \land y=k} |}{|S|} \cdot \log_2 \frac{|\sigma_{a_k=1 \land y=k} |}{|S|} - \frac{|\sigma_{a_k=0 \land y=k} |}{|S|} \cdot \log_2 \frac{|\sigma_{a_k=0 \land y=k} |}{|S|}}$$  \hspace{1cm} (9)

3.10 **Binary Criteria**

The binary criteria are used for creating binary decision trees. These measures are based on division of the input attribute domain into two sub-domains.

3.11 **Towing Criterion**

The gini index may encounter problems when the domain of the target attribute is relatively wide. In this case it is possible to employ binary criterion called towing criterion. This criterion is defined as:

$$twotng(a_i, dom_1(a_i), dom_2(a_i), S) = 0.25 \cdot \frac{|\sigma_{a_k=dom_1(a_i)} |}{|S|} \cdot \frac{|\sigma_{a_k=dom_2(a_i)} |}{|S|} \cdot \left( \sum_{y \in dom(y)} \left( \frac{|\sigma_{a_k=dom_1(a_i) \land y=y} |}{|\sigma_{a_k=dom_1(a_i)} |} \cdot \frac{|\sigma_{a_k=dom_2(a_i) \land y=y} |}{|\sigma_{a_k=dom_2(a_i)} |} \right)^2 \right)^{\frac{1}{2}}$$  \hspace{1cm} (11)
When the target attribute is binary, the gini and towing criteria are equivalent. For multi-class problems, the towing criteria prefer attributes with evenly divided splits.

3.12 ORTHOGONAL (ORT) CRITERION
The ORT criterion was presented by Fayyad and Irani (1992). This binary criterion is defined as:

\[ ORT(a_i, dom_1(a_i), dom_2(a_i), S) = 1 - \cos \theta (P_{i1}, P_{i2}) \]  

(12)

3.13 KOLMOGOROV–SMIRNOV CRITERION
A binary criterion that uses Kolmogorov–Smirnov distance has been proposed in Friedman (1977) and Rounds (1980). Assuming a binary target attribute, namely dom(y) = {c1; c2}, the criterion is defined as:

\[ KS(a_i, dom_1(a_i), dom_2(a_i), S) = \left| \frac{\sigma_{a_i \in dom_1(a_i), \ AND \ y = c_1 S}}{\sigma_{y = c_1 S}} - \frac{\sigma_{a_i \in dom_2(a_i), \ AND \ y = c_2 S}}{\sigma_{y = c_2 S}} \right| \]  

(13)

This measure was extended in (Utgoff and Clouse, 1996) to handle target attributes with multiple classes and missing data values. Their results indicate that the suggested method outperforms the gain ratio criteria.

3.14 AUC–SPLITTING CRITERIA
The idea of using the AUC metric as a splitting criterion was recently proposed for obtaining the maximal area under the convex hull of the ROC curve. It is important to note that unlike impurity criteria, this criterion does not perform a comparison between the impurities of the parent node with the weighted impurity of the children after splitting.

4. MULTIVARIATE SPLITTING CRITERIA
In multivariate splitting criteria, several attributes may participate in a single node split test. Obviously, finding the best multivariate criteria is more complicated than finding the best Univariate split. Furthermore, this type of criteria may dramatically improve the tree’s performance, these criteria are much less popular than the Univariate criteria.

5. STOPPING CRITERIA
The growing phase continues until a stopping criterion is triggered. The following conditions are common stopping rules:

1. All instances in the training set belong to a single value of y.
2. The maximum tree depth has been reached.
3. The number of cases in the terminal node is less than the minimum number of cases for parent nodes.
4. If the node were split, the number of cases in one or more child nodes would be less than the minimum number of cases for child nodes.
5. The best splitting criteria is not greater than a certain threshold.

6. PRUNING METHODS
6.1 OVERVIEW
Pruning methods originally suggested in were developed for solving this dilemma. According to this methodology, a loosely stopping criterion is used, letting the decision tree to over fit the training set. Then the over-fitted tree is cut back into a smaller tree by removing sub-branches that are not contributing to the generalization accuracy. It has been shown in various studies that employing pruning methods can improve the generalization performance of a decision tree, especially in noisy domains.
When the goal is to produce a sufficiently accurate compact concept description, pruning is highly useful. Within this process, the initial decision tree is seen as a completely accurate one. Thus, the accuracy of a pruned decision tree indicates how close it is to the initial tree. There are various techniques for pruning decision trees. Most of them perform top-down or bottom-up traversal of the nodes. A node is pruned if this operation improves a certain criteria.

6.2 Cost–Complexity Pruning

Cost-complexity pruning proceeds in two stages. In the first stage, a sequence of trees \( T_0, T_1, \ldots, T_k \) is built on the training data where \( T_0 \) is the original tree before pruning and \( T_k \) is the root tree. In the second stage, one of these trees is chosen as the pruned tree, based on its generalization error estimation. The tree \( T_{i+1} \) is obtained by replacing one or more of the sub-trees in the predecessor tree \( T_i \) with suitable leaves. The sub-trees that are pruned are those that obtain the lowest increase in apparent error rate per pruned leaf:

\[
\alpha = \frac{s(\text{pruned}(T, t), S) - s(T, S)}{|\text{leaves}(T)| - |\text{leaves}(\text{pruned}(T, t))|}
\]  

(14)

The best pruned tree is then selected. If the given dataset is large enough, the authors suggest breaking it into a training set and a pruning set. The trees are constructed using the training set and evaluated on the pruning set. On the other hand, if the given dataset is not large enough, they propose to use cross-validation methodology, despite the computational complexity implications.

6.3 Reduced Error Pruning

A simple procedure for pruning decision trees, known as reduced error pruning, the procedure checks for each internal node, whether replacing it with the most frequent class does not reduce the tree’s accuracy. In this case, the node is pruned. The procedure continues until any further pruning would decrease the accuracy.

6.4 Minimum Error Pruning (MEP)

The minimum error pruning performs bottom-up traversal of the internal nodes. In each node it compares the l-probability error rate estimation with and without pruning. The l-probability error rate estimation is a correction to the simple probability estimation using frequencies. If \( S_t \) denotes the instances that have reached a leaf \( t \), then the expected error rate in this leaf is:

\[
\varepsilon'(T, t) = 1 - \max_{c_j \in \text{classes}(y)} \frac{|c_j = c_t| S_t + l \cdot P_{\text{approx}}(y = c_t)}{|S_t| + 1}
\]  

(15)

Probability of \( y \) getting the value \( c_i \) and \( d_t \) denotes the weight given to the \( a \)-priori probability. The error rate of an internal node is the weighted average of the error rate of its branches. The weight is determined according to the proportion of instances along each branch.

6.5 Pessimistic Pruning

Pessimistic pruning avoids the need of pruning set or cross validation and uses the pessimistic statistical correlation test. The basic idea is that the error ratio estimated using the training set is not reliable enough. Instead, a more realistic measure, known as the continuity correction for binomial distribution, should be used:

\[
\varepsilon'(T, S) = \varepsilon(T, S) + \frac{|\text{leaves}(T)|}{2 \cdot |S|}
\]  

(16)

The last condition is based on statistical confidence interval for proportions. Usually the last condition is used such that \( T \) refers to a sub-tree whose root is the internal node \( t \) and \( S \) denotes the portion of the training set that refers to the node \( t \). The pessimistic pruning procedure performs top-down traversing over the internal nodes. If an internal node is pruned, then all its descendants are removed from the pruning process, resulting in a relatively fast pruning.
6.6 Error–based Pruning (EBP)
Error–based pruning is an evolution of pessimistic pruning. It is implemented in the well–known C4.5 algorithm. As in pessimistic pruning, the error rate is estimated using the upper bound of the statistical confidence interval for proportions.

\[
\varepsilon_{EBP}(T, S) = \varepsilon(T, S) + Z \sqrt{\frac{\varepsilon(T, S)(1 - \varepsilon(T, S))}{|S|}}
\]

(17)

Let \(\varepsilon(T, S)\) denotes the misclassification rate of the tree \(T\) on the training set \(S\). \(Z\) is the inverse of the standard normal cumulative distribution and \(\beta\) is the desired significance level.

6.7 Minimum Description Length (MDL) Pruning
The minimum description length can be used for evaluating the generalized accuracy of a node where \(St\) denotes the instances that have reached node \(t\). The splitting cost of an internal node is calculated based on the cost aggregation of its children.

\[
Cost(t) = \sum_{c \in \text{classes}(S)} \left[ \frac{|S|}{|S_{c} - S_{t}|} \ln \frac{|S_{c}|}{|S|} + \ln \frac{|\text{dom}(j)|}{2} - \ln |S_{c}| \right]
\]

(18)

7. Other Issues

7.1 Weighting Instances

Some decision trees inducers may give different treatments to different instances. This is performed by weighting the contribution of each instance in the analysis according to a provided weight (between 0 and 1).

7.2 Misclassification Costs

Several decision trees inducers can be provided with numeric penalties for classifying an item into one class when it really belongs in another.

7.3 Handling Missing Values

Missing values are a common experience in real-world data sets. This situation can complicate both induction (a training set where some of its values are missing) as well as classification (a new instance that miss certain values). This problem has been addressed by several researchers (Friedman, 1977; Breiman et al., 1984; Quinlan, 1989). One can handle missing values in the training set in the following way: When calculating the splitting criteria using attribute \(a_i\), simply ignore all instances their values in attribute \(a_i\) are unknown, i.e. instead of using the splitting criteria \(\varepsilon_{EBP}(ai ; S)\) it uses \(\varepsilon_{EBP}(ai ; S \mid \exists ai = ?S)\).

On the other hand, in case of missing values, the splitting criteria should be reduced proportionally as nothing has been learned from these instances (Quinlan, 1989). In other words, instead of using the splitting criteria

\[
\frac{|S - a_{ai} = ?S|}{|S|} \Delta \Phi(a_j, S - a_{ai} = ?S)
\]

(19)
Once a node is split, it is required to each one of the outgoing edges with the following corresponding weight:

\[
\text{GainRatio}(a_i, S) = \frac{\sum_{q \in Q} \text{InformationGain}(a_i, S - v_{a_i} = q^*)}{\sum_{q \in Q} \log \left( \frac{|q|}{|S|} \right) - \sum_{u \in \text{dom}(a_i)} \frac{|u|}{|S|} \log \left( \frac{|u|}{|S|} \right)}
\]

(20)

Where, yq denotes the value of the target attribute in the tuple q. If the missing attribute ai is numeric, then instead of using mode of ai it is more appropriate to use its mean.

8. Decision Trees Inducers

8.1 ID3

The ID3 algorithm is considered as a very simple decision tree algorithm value of target feature or when best information gain is not greater than zero. ID3 does not apply any pruning procedures nor does it handle numeric attributes or missing values.

8.2 C4.5

C4.5 is an evolution of ID3, presented by the same author (Quinlan, 1993). It uses gain ratio as splitting criteria. The splitting ceases when the number of instance to be split is below a certain threshold. Error–based pruning is performed after the growing phase. C4.5 can handle numeric attributes. It can induce from a training set that incorporates missing values by using corrected gain ratio criteria as presented above.

8.3 CART

CART stands for Classification and Regression Trees. It is characterized by the fact that it constructs binary trees, namely each internal node has exactly two outgoing edges. The splits are selected using the towing criteria and the obtained tree is pruned by cost–complexity pruning. When provided, CART can consider misclassification costs in the tree induction. It also enables users to provide prior probability distribution. An important feature of CART is its ability to generate regression trees. Regression trees are trees where their leaves predict a real number and not a class. In case of regression, CART looks for splits that minimize the predictions squared error (the least–squared deviation). The prediction in each leaf is based on the weighted mean for node.

8.4 CHAID

Starting from the early seventies, researchers in applied statistics developed procedures for generating decision trees, such as was originally designed to handle nominal attributes only. For each input attribute ai, CHAID finds the pair of values in V that is least significantly different with respect to the target attribute. The significant difference is measured by the p value obtained from a statistical test. The statistical test used dependson the type of target attribute. If the target attribute is continuous, an F test is used. If it is nominal, then a Pearson chi–squared test is used. If it is ordinal, then a likelihood–ratio test is used. For each selected pair, CHAID checks if the p value obtained is greater than a certain merge threshold. If the answer is positive, it merges the values and searches for an additional potential pair to be merged. The process is repeated until no significant pairs are found. The best input attribute to be used for splitting the current node is then selected, such that each child node is made of a group of homogeneous values of the selected attribute. Note that no split is performed if the adjusted p value of the best input attribute is not less than a certain split threshold. This procedure also stops when one of the following conditions is fulfilled:
1. Maximum tree depth is reached.
2. Minimum number of cases in node for being a parent is reached, so it cannot be split any further.
3. Minimum number of cases in node for being a child node is reached.

CHAID handles missing values by treating them all as a single valid category. CHAID does not perform pruning.

8.5 QUEST
The QUEST (Quick, Unbiased, Efficient, Statistical Tree) algorithm supports Univariate and linear combination splits (Loh and Shih, 1997). For each split, the association between each input attribute and the target attribute is computed using the ANOVA F-test or Levene’s test (for ordinal and continuous attributes) or Pearson’s chi-square (for nominal attributes). If the target attribute is multinomial, two-means clustering is used to create two super-classes. The attribute that obtains the highest association with the target attribute is selected for splitting. Quadratic Discriminant Analysis (QDA) is applied to find the optimal splitting point for the input attribute. QUEST has negligible bias and it yields binary decision trees. Ten-fold cross-validation is used to prune the trees.

9. ADVANTAGES AND DISADVANTAGES OF DECISION TREES
Several advantages of the decision tree as a classification tool have been pointed out in the literature:

1. Decision trees are self-explanatory and when compacted they are also easy to follow. In other words if the decision tree has a reasonable number of leaves, it can be grasped by non-professional users. Furthermore decision trees can be converted to a set of rules. Thus, this representation is considered as comprehensible.
2. Decision trees can handle both nominal and numeric input attributes.
3. Decision tree representation is rich enough to represent any discrete-value classifier.
4. Decision trees are capable of handling datasets that may have errors.
5. Decision trees are capable of handling datasets that may have missing values.
6. Decision trees are considered to be a nonparametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

DISADVANTAGES:

1. Most of the algorithms (like ID3 and C4.5) require that the target attribute will have only discrete values.
2. As decision trees use the “divide and conquer” method, they tend to perform well if a few highly relevant attributes exist, but less so if many complex interactions are present. One of the reasons for this is that other classifiers can compactly describe a classifier that would be very challenging to represent using a decision tree. A simple illustration of this phenomenon is the replication problem of decision trees (Pagallo and Huassler, 1990). Since most decision trees divide the instance space into mutually exclusive regions to represent a concept, in some cases the trees should contain several duplications of the same sub-tree in order to represent the classifier. For instance if the concept follows the following binary function: \( y = (A1 \setminus A2) \setminus (A3 \setminus A4) \) then the minimal Univariate decision tree that represents this function. Note that the tree contains two copies of the same sub-tree.
3. The greedy characteristic of decision trees leads to another disadvantage that should be pointed out. This is its over-sensitivity to the training set, to irrelevant attributes and to noise (Quinlan, 1993).

10. DECISION TREE EXTENSIONS
In the following sub-sections, we discuss some of the most popular extensions to the classical decision tree induction paradigm.

10.1 OBLIVIOUS DECISION TREES
Oblivious decision trees are decision trees for which all nodes at the same level test the same feature. Despite its restriction, oblivious decision trees are found to be effective for feature selection. It has been shown that oblivious decision trees can be converted to a decision table.
The principal difference between the oblivious decision tree and a regular decision tree structure is the constant ordering of input attributes at every terminal node of the oblivious decision tree, the property which is necessary for minimizing the overall subset of input attributes (resulting in dimensionality reduction). An oblivious decision tree is usually built by a greedy algorithm, which tries to maximize the mutual information measure in every layer. The recursive search for explaining attributes is terminated when there is no attribute that explains the target with statistical significance.

10.2 Fuzzy Decision Trees

Fuzzy decision trees (FDT) may simultaneously assign more than one branch to the same instance with gradual certainty. FDTs preserve the symbolic structure of the tree and its comprehensibility. Nevertheless, FDT can represent concepts with graduated characteristics by producing real-valued outputs with gradual shifts. Janikow (1998) presented a complete framework for building a fuzzy tree including several inference procedures based on conflict resolution in rule-based systems and efficient approximate reasoning methods. Olaru and Wehenkel (2003) presented a new fuzzy decision trees called soft decision trees (SDT). This approach combines tree-growing and pruning, to determine the structure of the soft decision tree, with refitting and back fitting, to improve its generalization capabilities. They empirically showed that soft decision trees are significantly more accurate than standard decision trees. Moreover, a global model variance study shows a much lower variance for soft decision trees than for standard trees as a direct cause of the improved accuracy.

10.3 Decision Trees Inducers for Large Datasets

With the recent growth in the amount of data collected by information systems, there is a need for decision trees that can handle large datasets. Has examined two methods for efficiently growing decision trees from a large database by reducing the computation complexity required for induction. However, the Catlett method requires that all data will be loaded into the main memory before induction. That is to say, the largest dataset that can be induced is bounded by the memory size. Suggest partitioning the datasets into several disjointed datasets, so that each dataset is loaded separately into the memory and used to induce a decision tree. The decision trees are then combined to create a single classifier. However, the experimental results indicate that partition may reduce the classification performance, meaning that the classification accuracy of the combined decision trees is not as good as the accuracy of a single decision tree induced from the entire dataset. The SLIQ algorithm does not require loading the entire dataset into the main memory, instead it uses a secondary memory (disk). In other words, a certain instance is not necessarily resident in the main memory all the time. SLIQ creates a single decision tree from the entire dataset.

10.4 Incremental Induction

Most of the decision trees inducers require rebuilding the tree from scratch for reflecting new data that has become available. Several researches have addressed the issue of updating decision trees incrementally.

REFERENCES


Author:

Mr. Pramod Patil is working as Associate Professor at D. Y. Patil Institute of Engineering & Technology, Pune(M.S.), India. He is pursuing Ph.D. from University of Pune, Pune. His research areas include Data Mining, Decision Making. He has completed his M.Tech in Computer Engineering from University of Pune, Pune. He has couple of renowned publications on his name and papers presented in various conferences. He is in the education field and teaching at UG& PG