Decision Trees

Chapter · January 2005

DOI: 10.1007/0-387-25465-X_9

CITATIONS
63

READS
3,095

2 authors:

Lior Rokach
Ben-Gurion University of the Negev
308 PUBLICATIONS **10,999** CITATIONS

SEE PROFILE

Oded Maimon
Tel Aviv University
232 PUBLICATIONS **5,802** CITATIONS

SEE PROFILE

Chapter 9

DECISION TREES

Lior Rokach
Department of Industrial Engineering
Tel-Aviv University
liorr@eng.tau.ac.il

Oded Maimon
Department of Industrial Engineering
Tel-Aviv University
maimon@eng.tau.ac.il

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 paper presents an updated survey of current methods for constructing decision tree classifiers in a top-down manner. The chapter suggests a unified algorithmic framework for presenting these algorithms and describes various splitting criteria and pruning 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 instance space. 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 fre-

quent 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.

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 9.1 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.

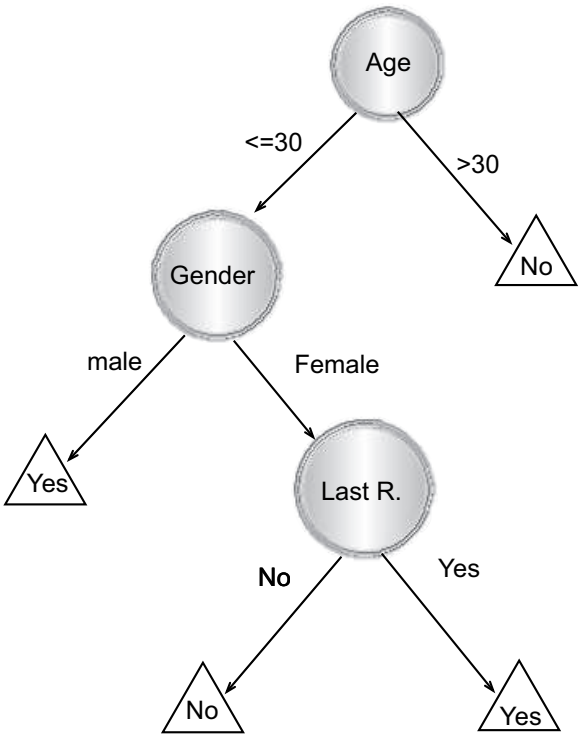

Figure 9.1. Decision Tree Presenting Response to Direct Mailing.

In case of numeric attributes, decision trees can be geometrically interpreted as a collection of hyperplanes, 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. For example, one of the paths in Figure 9.1 can be transformed into the rule: "If customer age is is less than or equal to or equal to 30, and the gender of the customer is "Male" – then the customer will respond to the mail". 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.

There are various top–down decision trees inducers such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), CART (Breiman *et al*., 1984). Some consist of two conceptual phases: growing and pruning (C4.5 and CART). Other inducers perform only the growing phase.

Figure 9.2 presents a typical algorithmic framework for top–down inducing of a decision tree using growing and pruning. Note that these algorithms are greedy by nature and construct the decision tree in a top–down, recursive manner (also known as "divide and conquer"). In each iteration, the algorithm considers the partition of the training set using the outcome of a discrete function of the input attributes. The selection of the most appropriate function is made according to some splitting measures. After the selection of an appropriate split, 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:

- According to the origin of the measure: information theory, dependence, and distance.
- According to the measure structure: impurity based criteria, normalized 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 ϕ:[0, 1]$^k \rightarrow R$ that satisfies the following conditions:

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

```
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,...,v_n) gains
        the best splitting metric.
    IF best splitting metric > treshold THEN
        Label t with f(A)
        FOR each outcome v_i of f(A):
            Set Subtree_i= TreeGrowing (σ_{f(A)=v_i} S,A,y).
            Connect the root node of t_T to Subtree_i with
                    an edge that is labelled as v_i
        END FOR
    ELSE
        Mark the root node in T as a leaf with the most
            common value of y in S as 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≠Ø THEN T=pruned(T,t)
UNTIL t=Ø
RETURN T
```

Figure 9.2. Top-Down Algorithmic Framework for Decision Trees Induction.

Note that if the probability vector has a component of 1 (the variable x gets only 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:

$$P_y(S) = \left(\frac{|\sigma_{y=c_1} S|}{|S|}, \ldots, \frac{\left|\sigma_{y=c_{|dom(y)|}} S\right|}{|S|} \right)$$

The goodness–of–split due to discrete attribute a_i is defined as reduction in impurity of the target attribute after partitioning S according to the values $v_{i,j} \in dom(a_i)$:

$$\Delta\Phi(a_i, S) = \phi(P_y(S)) - \sum_{j=1}^{|dom(a_i)|} \frac{|\sigma_{a_i=v_{i,j}} S|}{|S|} \cdot \phi(P_y(\sigma_{a_i=v_{i,j}} S))$$

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).

$$InformationGain(a_i, S) = Entropy(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i=v_{i,j}} S|}{|S|} \cdot Entropy(y, \sigma_{a_i=v_{i,j}} S)$$

where:

$$Entropy(y, S) = \sum_{c_j \in dom(y)} -\frac{\left|\sigma_{y=c_j} S\right|}{|S|} \cdot \log_2 \frac{\left|\sigma_{y=c_j} S\right|}{|S|}$$

3.4 Gini Index

Gini index is an impurity-based criterion that measures the divergences between the probability distributions of the target attribute's values. The Gini index has been used in various works such as (Breiman *et al*., 1984) and (Gelfand *et al*., 1991) and it is defined as:

$$Gini(y, S) = 1 - \sum_{c_j \in dom(y)} \left(\frac{\left|\sigma_{y=c_j} S\right|}{|S|} \right)^2$$

Consequently the evaluation criterion for selecting the attribute a_i is defined as:

$$GiniGain(a_i, S) = Gini(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{\left|\sigma_{a_i=v_{i,j}} S\right|}{|S|} \cdot Gini(y, \sigma_{a_i=v_{i,j}} S)$$

3.5 Likelihood-Ratio Chi–Squared Statistics

The likelihood–ratio is defined as (Attneave, 1959)

$$G^2(a_i, S) = 2 \cdot \ln(2) \cdot |S| \cdot InformationGain(a_i, S)$$

This ratio is useful for measuring the statistical significance of the information gain criterion. The zero hypothesis (H_0) is that the input attribute and the target attribute are conditionally independent. If H_0 holds, the test statistic is distributed as χ^2 with degrees of freedom equal to: $(dom(a_i) - 1) \cdot (dom(y) - 1)$.

3.6 DKM Criterion

The DKM criterion is an impurity-based splitting criterion designed for binary class attributes (Dietterich *et al*., 1996) and (Kearns and Mansour, 1999). The impurity-based function is defined as:

$$DKM(y, S) = 2 \cdot \sqrt{\left(\frac{|\sigma_{y=c_1} S|}{|S|}\right) \cdot \left(\frac{|\sigma_{y=c_2} S|}{|S|}\right)}$$

It has been theoretically proved (Kearns and Mansour, 1999) 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, as described in the following sections.

3.8 Gain Ratio

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

$$GainRatio(a_i, S) = \frac{InformationGain(a_i, S)}{Entropy(a_i, S)}$$

Note that this ratio is not defined when the denominator is zero. Also the ratio may tend to favor attributes for which the denominator is very small. Consequently, it is suggested in two stages. First the information gain is calculated for all attributes. As a consequence, taking into consideration only attributes

that have performed at least as good as the average information gain, the attribute that has obtained the best ratio gain is selected. It has been shown that the gain ratio tends to outperform simple information gain criteria, both from the accuracy aspect, as well as from classifier complexity aspects (Quinlan, 1988).

3.9 Distance Measure

The distance measure, like the gain ratio, normalizes the impurity measure. However, it suggests normalizing it in a different way (Lopez de Mantras, 1991):

$$\frac{\Delta\Phi(a_i,S)}{-\sum_{v_{i,j}\in dom(a_i)}\sum_{c_k\in dom(y)}\frac{|\sigma_{a_i=v_{i,j}\ AND\ y=c_k}S|}{|S|}\cdot\log_2\frac{|\sigma_{a_i=v_{i,j}\ AND\ y=c_k}S|}{|S|}}$$

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.

Let $\beta(a_i, dom_1(a_i), dom_2(a_i), S)$ denote the binary criterion value for attribute a_i over sample S when $dom_1(a_i)$ and $dom_2(a_i)$ are its corresponding subdomains. The value obtained for the optimal division of the attribute domain into two mutually exclusive and exhaustive sub-domains is used for comparing attributes.

3.11 Twoing Criterion

The gini index may encounter problems when the domain of the target attribute is relatively wide (Breiman *et al.*, 1984). In this case it is possible to employ binary criterion called twoing criterion. This criterion is defined as:

$$twoing(a_i, dom_1(a_i), dom_2(a_i), S) = 0.25\cdot\frac{\left|\sigma_{a_i\in dom_1(a_i)}S\right|}{|S|}\cdot\frac{\left|\sigma_{a_i\in dom_2(a_i)}S\right|}{|S|}\cdot\left(\sum_{c_i\in dom(y)}\left|\frac{\left|\sigma_{a_i\in dom_1(a_i)\ AND\ y=c_i}S\right|}{\left|\sigma_{a_i\in dom_1(a_i)}S\right|}-\frac{\left|\sigma_{a_i\in dom_2(a_i)\ AND\ y=c_i}S\right|}{\left|\sigma_{a_i\in dom_2(a_i)}S\right|}\right|\right)^2$$

When the target attribute is binary, the gini and twoing criteria are equivalent. For multi–class problems, the twoing 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:

$$ORT(a_i, dom_1(a_i), dom_2(a_i), S) = 1 - cos\theta(P_{y,1}, P_{y,2})$$

where $\theta(P_{y,1}, P_{y,2})$ is the angle between two vectors $P_{y,1}$ and $P_{y,2}$. These vectors represent the probability distribution of the target attribute in the partitions $\sigma_{a_i \in dom_1(a_i)} S$ and $\sigma_{a_i \in dom_2(a_i)} S$ respectively.

It has been shown that this criterion performs better than the information gain and the Gini index for specific problem constellations.

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 $dom(y) = \{c_1, c_2\}$, the criterion is defined as:

$$KS(a_i, dom_1(a_i), dom_2(a_i), S) =$$

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

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 in (Ferri *et al.*, 2002). The attribute that obtains the maximal area under the convex hull of the ROC curve is selected. It has been shown that the AUC–based splitting criterion outperforms other splitting criteria both with respect to classification accuracy and area under the ROC curve. It is important to note that unlike impurity criteria, this criterion does not perform a comparison between the impurity of the parent node with the weighted impurity of the children after splitting.

3.15 Other Univariate Splitting Criteria

Additional univariate splitting criteria can be found in the literature, such as permutation statistics (Li and Dubes, 1986), mean posterior improvements (Taylor and Silverman, 1993) and hypergeometric distribution measures (Martin, 1997).

3.16 Comparison of Univariate Splitting Criteria

Comparative studies of the splitting criteria described above, and others, have been conducted by several researchers during the last thirty years, such as (Baker and Jain, 1976; BenBassat, 1978; Mingers, 1989; Fayyad and Irani, 1992; Buntine and Niblett, 1992; Loh and Shih, 1997; Loh and Shih, 1999; Lim *et al.*, 2000). Most of these comparisons are based on empirical results, although there are some theoretical conclusions.

Many of the researchers point out that in most of the cases, the choice of splitting criteria will not make much difference on the tree performance. Each criterion is superior in some cases and inferior in others, as the "No–Free–Lunch" theorem suggests.

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, although this type of criteria may dramatically improve the tree's performance, these criteria are much less popular than the univariate criteria.

Most of the multivariate splitting criteria are based on the linear combination of the input attributes. Finding the best linear combination can be performed using a greedy search (Breiman *et al.*, 1984; Murthy, 1998), linear programming (Duda and Hart, 1973; Bennett and Mangasarian, 1994), linear discriminant analysis (Duda and Hart, 1973; Friedman, 1977; Sklansky and Wassel, 1981; Lin and Fu, 1983; Loh and Vanichsetakul, 1988; John, 1996) and others (Utgoff, 1989a; Lubinsky, 1993; Sethi and Yoo, 1994).

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

Employing tightly stopping criteria tends to create small and under–fitted decision trees. On the other hand, using loosely stopping criteria tends to generate large decision trees that are over–fitted to the training set. Pruning methods originally suggested in (Breiman *et al.*, 1984) were developed for solving this dilemma. According to this methodology, a loosely stopping criterion is used, letting the decision tree to overfit 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.

Another key motivation of pruning is "trading accuracy for simplicity" as presented in (Bratko and Bohanec, 1994). 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. The following subsections describe the most popular techniques.

6.2 Cost–Complexity Pruning

Cost-complexity pruning (also known as weakest link pruning or error-complexity pruning) proceeds in two stages (Breiman *et al.*, 1984). In the first stage, a sequence of trees T_0, T_1, ..., 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:

$$\alpha = \frac{\varepsilon(pruned(T,t),S) - \varepsilon(T,S)}{|leaves(T)| - |leaves(pruned(T,t))|}$$

where $\varepsilon(T,S)$ indicates the error rate of the tree T over the sample S and |leaves(T)| denotes the number of leaves in T. *pruned(T,t)* denotes the tree obtained by replacing the node t in T with a suitable leaf.

In the second phase the generalization error of each pruned tree T_0, T_1, ..., T_k is estimated. 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, has been suggested by Quinlan (1987). While traversing over the internal nodes from the bottom to the top, 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.

In order to estimate the accuracy, Quinlan (1987) proposes to use a pruning set. It can be shown that this procedure ends with the smallest accurate sub–tree with respect to a given pruning set.

6.4 Minimum Error Pruning (MEP)

The minimum error pruning has been proposed in (Olaru and Wehenkel, 2003). It 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:

$$\varepsilon'(t) = 1 - \max_{c_i \in dom(y)} \frac{|\sigma_{y=c_i} S_t| + l \cdot p_{apr}(y = c_i)}{|S_t| + l}$$

where $p_{apr}(y = c_i)$ is the *a–priori* probability of y getting the value c_i, and l 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. The calculation is performed recursively up to the leaves.

If an internal node is pruned, then it becomes a leaf and its error rate is calculated directly using the last equation. Consequently, we can compare the error rate before and after pruning a certain internal node. If pruning this node does not increase the error rate, the pruning should be accepted.

6.5 Pessimistic Pruning

Pessimistic pruning avoids the need of pruning set or cross validation and uses the pessimistic statistical correlation test instead (Quinlan, 1993).

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:

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

However, this correction still produces an optimistic error rate. Consequently, one should consider pruning an internal node t if its error rate is within one standard error from a reference tree, namely (Quinlan, 1993):

$$\varepsilon'(pruned(T,t),S) \leq \varepsilon'(T,S) + \sqrt{\frac{\varepsilon'(T,S) \cdot (1-\varepsilon'(T,S))}{|S|}}$$

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.

$$\varepsilon_{UB}(T,S) = \varepsilon(T,S) + Z_\alpha \cdot \sqrt{\frac{\varepsilon(T,S) \cdot (1-\varepsilon(T,S))}{|S|}}$$

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

Let *subtree(T,t)* denote the subtree rooted by the node t. Let *maxchild(T,t)* denote the most frequent child node of t (namely most of the instances in S reach this particular child) and let S_t denote all instances in S that reach the node t.

The procedure performs bottom–up traversal over all nodes and compares the following values:

1. $\varepsilon_{UB}(subtree(T,t),S_t)$
2. $\varepsilon_{UB}(pruned(subtree(T,t),t),S_t)$

3. $\varepsilon_{UB}(subtree(T, maxchild(T,t)), S_{maxchild(T,t)})$

According to the lowest value the procedure either leaves the tree as is, prune the node t, or replaces the node t with the subtree rooted by *maxchild(T,t)*.

6.7 Optimal Pruning

The issue of finding optimal pruning has been studied in (Bratko and Bohanec, 1994) and (Almuallim, 1996). The first research introduced an algorithm which guarantees optimality, known as OPT. This algorithm finds the optimal pruning based on dynamic programming, with the complexity of $\Theta(|leveas(T)|^2)$, where T is the initial decision tree. The second research introduced an improvement of OPT called OPT–2, which also performs optimal pruning using dynamic programming. However, the time and space complexities of OPT–2 are both $\Theta(|leveas(T^*)| \cdot |\text{internal}(T)|)$, where T^* is the target (pruned) decision tree and T is the initial decision tree.

Since the pruned tree is habitually much smaller than the initial tree and the number of internal nodes is smaller than the number of leaves, OPT–2 is usually more efficient than OPT in terms of computational complexity.

6.8 Minimum Description Length (MDL) Pruning

The minimum description length can be used for evaluating the generalized accuracy of a node (Rissanen, 1989; Quinlan and Rivest, 1989; Mehta *et al.*, 1995). This method measures the size of a decision tree by means of the number of bits required to encode the tree. The MDL method prefers decision trees that can be encoded with fewer bits. The cost of a split at a leaf t can be estimated as (Mehta *et al.*, 1995):

$$\text{Cost(t)} = \sum_{c_i \in dom(y)} |\sigma_{y=c_i} S_t| \cdot \ln \frac{|S_t|}{|\sigma_{y=c_i} S_t|} + \frac{|dom(y)|-1}{2} \ln \frac{|S_t|}{2} + \ln \frac{\pi^{\frac{|dom(y)|}{2}}}{\Gamma(\frac{|dom(y)|}{2})}$$

where S_t 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.

6.9 Other Pruning Methods

There are other pruning methods reported in the literature, such as the MML (Minimum Message Length) pruning method (Wallace and Patrick, 1993) and Critical Value Pruning (Mingers, 1989).

6.10 Comparison of Pruning Methods

Several studies aim to compare the performance of different pruning techniques (Quinlan, 1987; Mingers, 1989; Esposito *et al.*, 1997). The results indicate that some methods (such as cost–complexity pruning, reduced error pruning) tend 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 toward under–pruning. Most of the comparisons concluded that the "no free lunch" theorem applies in this case also, namely there is no pruning method that in any case outperforms other pruning methods.

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: let $\sigma_{a_i=?}S$ indicate the subset of instances in S whose a_i values are missing. When calculating the splitting criteria using attribute a_i, simply ignore all instances their values in attribute a_i are unknown, that is, instead of using the splitting criteria $\Delta\Phi(a_i, S)$ it uses $\Delta\Phi(a_i, S - \sigma_{a_i=?}S)$.

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 $\Delta\Phi(a_i, S)$, it uses the following correction:

$$\frac{|S - \sigma_{a_i=?}S|}{|S|}\Delta\Phi(a_i, S - \sigma_{a_i=?}S).$$

In a case where the criterion value is normalized (as in the case of gain ratio), the denominator should be calculated as if the missing values represent an

additional value in the attribute domain. For instance, the Gain Ratio with missing values should be calculated as follows:

$$GainRatio(a_i, S) = \frac{\frac{\left|S-\sigma_{a_i=?}S\right|}{|S|} InformationGain(a_i, S-\sigma_{a_i=?}S)}{-\frac{\left|\sigma_{a_i=?}S\right|}{|S|}\log(\frac{\left|\sigma_{a_i=?}S\right|}{|S|}) - \sum_{v_{i,j}\in dom(a_i)} \frac{\left|\sigma_{a_i=v_{i,j}}S\right|}{|S|}\log(\frac{\left|\sigma_{a_i=v_{i,j}}S\right|}{|S|})}$$

Once a node is split, it is required to add $\sigma_{a_i=?}S$ to each one of the outgoing edges with the following corresponding weight:

$$\left|\sigma_{a_i=v_{i,j}}S\right| / |S - \sigma_{a_i=?}S|$$

The same idea is used for classifying a new instance with missing attribute values. When an instance encounters a node where its splitting criteria can be evaluated due to a missing value, it is passed through to all outgoing edges. The predicted class will be the class with the highest probability in the weighted union of all the leaf nodes at which this instance ends up.

Another approach known as *surrogate splits* was presented by Breiman *et al*. (1984) and is implemented in the CART algorithm. The idea is to find for each split in the tree a surrogate split which uses a different input attribute and which most resembles the original split. If the value of the input attribute used in the original split is missing, then it is possible to use the surrogate split. The resemblance between two binary splits over sample S is formally defined as:

$$res(a_i, dom_1(a_i), dom_2(a_i), a_j, dom_1(a_j), dom_2(a_j), S) = \frac{\left|\sigma_{a_i\in dom_1(a_i) \;\; AND \;\; a_j\in dom_1(a_j)}S\right|}{|S|} + \frac{\left|\sigma_{a_i\in dom_2(a_i) \;\; AND \;\; a_j\in dom_2(a_j)}S\right|}{|S|}$$

When the first split refers to attribute a_i and it splits *dom*(a_i) into *dom*$_1(a_i)$ and *dom*$_2(a_i)$. The alternative split refers to attribute a_j and splits its domain to *dom*$_1(a_j)$ and *dom*$_2(a_j)$.

The missing value can be estimated based on other instances (Loh and Shih, 1997). On the learning phase, if the value of a nominal attribute a_i in tuple q is missing, then it is estimated by its mode over all instances having the same target attribute value. Formally,

$$estimate(a_i, y_q, S) = \underset{v_{i,j}\in dom(a_i)}{\text{argmax}} \left|\sigma_{a_i=v_{i,j} \;\; AND \;\; y=y_q}S\right|$$

where y_q denotes the value of the target attribute in the tuple q. If the missing attribute a_i is numeric, then instead of using mode of a_i 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 (Quinlan, 1986). ID3 uses information gain as splitting criteria. The growing stops when all instances belong to a single 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 instances 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 (Breiman *et al*., 1984). 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 twoing 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 prediction 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: AID (Sonquist *et al*., 1971), MAID (Gillo, 1972), THAID (Morgan and Messenger, 1973) and CHAID (Kass, 1980). CHAID (Chisquare–Automatic–Interaction–Detection) was originally designed to handle nominal attributes only. For each input attribute a_i, CHAID finds the pair of values in V_i 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 depends on 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 can not 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.

8.6 Reference to Other Algorithms

Table 9.1 describes other decision trees algorithms available in the literature. Obviously there are many other algorithms which are not included in this table. Nevertheless, most of these algorithms are a variation of the algorithmic framework presented above. A profound comparison of the above algorithms and many others has been conducted in (Lim *et al*., 2000).

Table 9.1. Additional Decision Tree Inducers.

Algorithm	Description	Reference
CAL5	Designed specifically for numerical–valued attributes	Muller and Wysotzki (1994)
FACT	An earlier version of QUEST. Uses statistical tests to select an attribute for splitting each node and then uses discriminant analysis to find the split point.	Loh and Vanichsetakul (1988)
LMDT	Constructs a decision tree based on multivariate tests are linear combinations of the attributes.	Brodley and Utgoff (1995)
T1	A one–level decision tree that classifies instances using only one attribute. Missing values are treated as a "special value". Support both continuous an nominal attributes.	Holte (1993)
PUBLIC	Integrates the growing and pruning by using MDL cost in order to reduce the computational complexity.	Rastogi and Shim (2000)
MARS	A multiple regression function is approximated using linear splines and their tensor products.	Friedman (1991)

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.

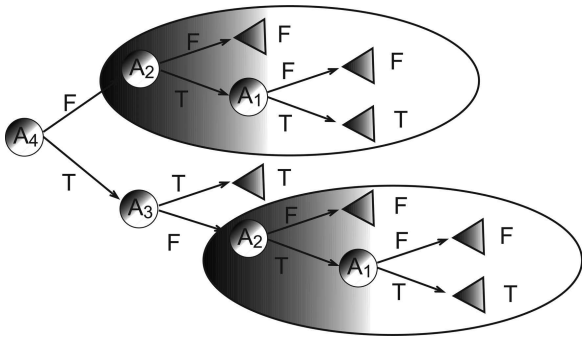

Figure 9.3. Illustration of Decision Tree with Replication.

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.

On the other hand, decision trees have such disadvantages as:

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 tree 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: $y = (A_1 \cap A_2) \cup (A_3 \cap A_4)$ then the minimal univariate decision tree that represents this function is illustrated in Figure 9.3. Note that the tree contains two copies of the same subt-ree.

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. Almuallim and Dietterich (1994) as well as Schlimmer (1993) have proposed forward feature selection procedure by constructing oblivious decision trees. Langley and Sage (1994) suggested backward selection using the same means. It has been shown that oblivious decision trees can be converted to a decision table (Kohavi and Sommerfield, 1998). Recently Last *et al*. (2002) have suggested a new algorithm for constructing oblivious decision trees, called IFN (Information Fuzzy Network) that is based on information theory.

Figure 9.4 illustrates a typical oblivious decision tree with four input features: glucose level (G), age (A), hypertension (H) and pregnant (P) and the Boolean target feature representing whether that patient suffers from diabetes. Each layer is uniquely associated with an input feature by representing the interaction of that feature and the input features of the previous layers. The number that appears in the terminal nodes indicates the number of instances that fit this path. For example, regarding patients whose glucose level is less than 107 and their age is greater than 50, 10 of them are positively diagnosed with diabetes while 2 of them are not diagnosed with diabetes.

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). The arcs that connect the terminal nodes and the nodes of the target layer are labelled with the number of records that fit this path.

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

In classical decision trees, an instance can be associated with only one branch of the tree. Fuzzy decision trees (FDT) may simultaneously assign more than one branch to the same instance with gradual certainty.

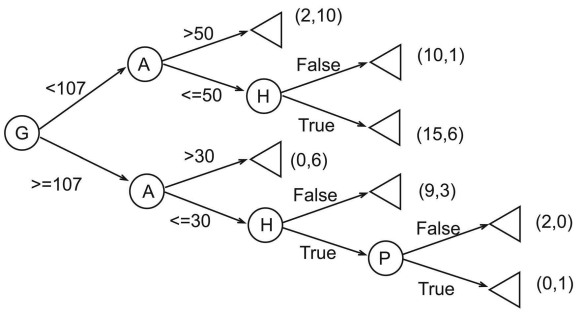

Figure 9.4. Illustration of Oblivious Decision Tree.

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 backfitting, 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.

Peng (2004) has used FDT to improve the performance of the classical inductive learning approach in manufacturing processes. Peng (2004) proposed to use soft discretization of continuous-valued attributes. It has been shown that FDT can deal with the noise or uncertainties existing in the data collected in industrial systems.

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. Catlett (1991) 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. Fifield (1992) suggests parallel implementation of the ID3 Algorithm. However, like Catlett, it assumes that all dataset can fit in the main memory. Chan and Stolfo (1997) 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 (Mehta *et al.*, 1996) 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. However, this method also has an upper limit for the largest dataset that can be processed, because it uses a data structure that scales with the dataset size and this data structure must be resident in main memory all the time. The SPRINT algorithm uses a similar approach (Shafer *et al.*, 1996). This algorithm induces decision trees relatively quickly and removes all of the memory restrictions from decision tree induction. SPRINT scales any impurity based split criteria for large datasets. Gehrke *et al* (2000) introduced RainForest; a unifying framework for decision tree classifiers that are capable of scaling any specific algorithms from the literature (including C4.5, CART and CHAID). In addition to its generality, RainForest improves SPRINT by a factor of three. In contrast to SPRINT, however, RainForest requires a certain minimum amount of main memory, proportional to the set of distinct values in a column of the input relation. However, this requirement is considered modest and reasonable.

Other decision tree inducers for large datasets can be found in the literature (Alsabti *et al.*, 1998; Freitas and Lavington, 1998; Gehrke *et al.*, 1999).

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. Utgoff (1989b, 1997) presents several methods for updating decision trees incrementally. An extension to the CART algorithm that is capable of inducing incrementally is described in (Crawford *et al.*, 2002).

References

Almuallim H., An Efficient Algorithm for Optimal Pruning of Decision Trees. Artificial Intelligence 83(2): 347-362, 1996.

Almuallim H,. and Dietterich T.G., Learning Boolean concepts in the presence of many irrelevant features. Artificial Intelligence, 69: 1-2, 279-306, 1994.

Alsabti K., Ranka S. and Singh V., CLOUDS: A Decision Tree Classifier for Large Datasets, Conference on Knowledge Discovery and Data Mining (KDD-98), August 1998.

Attneave F., Applications of Information Theory to Psychology. Holt, Rinehart and Winston, 1959.

Baker E., and Jain A. K., On feature ordering in practice and some finite sample effects. In Proceedings of the Third International Joint Conference on Pattern Recognition, pages 45-49, San Diego, CA, 1976.

BenBassat M., Myopic policies in sequential classification. IEEE Trans. on Computing, 27(2):170-174, February 1978.

Bennett X. and Mangasarian O.L., Multicategory discrimination via linear programming. Optimization Methods and Software, 3:29-39, 1994.

Bratko I., and Bohanec M., Trading accuracy for simplicity in decision trees, Machine Learning 15: 223-250, 1994.

Breiman L., Friedman J., Olshen R., and Stone C.. Classification and Regression Trees. Wadsworth Int. Group, 1984.

Brodley C. E. and Utgoff. P. E., Multivariate decision trees. Machine Learning, 19:45-77, 1995.

Buntine W., Niblett T., A Further Comparison of Splitting Rules for Decision-Tree Induction. Machine Learning, 8: 75-85, 1992.

Catlett J., Mega induction: Machine Learning on Vary Large Databases, PhD, University of Sydney, 1991.

Chan P.K. and Stolfo S.J, On the Accuracy of Meta-learning for Scalable Data Mining, J. Intelligent Information Systems, 8:5-28, 1997.

Crawford S. L., Extensions to the CART algorithm. Int. J. of ManMachine Studies, 31(2):197-217, August 1989.

Dietterich, T. G., Kearns, M., and Mansour, Y., Applying the weak learning framework to understand and improve C4.5. Proceedings of the Thirteenth International Conference on Machine Learning, pp. 96-104, San Francisco: Morgan Kaufmann, 1996.

Duda, R., and Hart, P., Pattern Classification and Scene Analysis, New-York, Wiley, 1973.

Esposito F., Malerba D. and Semeraro G., A Comparative Analysis of Methods for Pruning Decision Trees. EEE Transactions on Pattern Analysis and Machine Intelligence, 19(5):476-492, 1997.

Fayyad U., and Irani K. B., The attribute selection problem in decision tree generation. In proceedings of Tenth National Conference on Artificial Intelligence, pp. 104–110, Cambridge, MA: AAAI Press/MIT Press, 1992.

Ferri C., Flach P., and Hernández-Orallo J., Learning Decision Trees Using the Area Under the ROC Curve. In Claude Sammut and Achim Hoffmann, ed-

itors, Proceedings of the 19th International Conference on Machine Learning, pp. 139-146. Morgan Kaufmann, July 2002

Fifield D. J., Distributed Tree Construction From Large Datasets, Bachelor's Honor Thesis, Australian National University, 1992.

Freitas X., and Lavington S. H., Mining Very Large Databases With Parallel Processing, Kluwer Academic Publishers, 1998.

Friedman J. H., A recursive partitioning decision rule for nonparametric classifiers. IEEE Trans. on Comp., C26:404-408, 1977.

Friedman, J. H., "Multivariate Adaptive Regression Splines", The Annual Of Statistics, 19, 1-141, 1991.

Gehrke J., Ganti V., Ramakrishnan R., Loh W., BOAT-Optimistic Decision Tree Construction. SIGMOD Conference 1999: pp. 169-180, 1999.

Gehrke J., Ramakrishnan R., Ganti V., RainForest - A Framework for Fast Decision Tree Construction of Large Datasets,Data Mining and Knowledge Discovery, 4, 2/3) 127-162, 2000.

Gelfand S. B., Ravishankar C. S., and Delp E. J., An iterative growing and pruning algorithm for classification tree design. IEEE Transaction on Pattern Analysis and Machine Intelligence, 13(2):163-174, 1991.

Gillo M. W., MAID: A Honeywell 600 program for an automatised survey analysis. Behavioral Science 17: 251-252, 1972.

Hancock T. R., Jiang T., Li M., Tromp J., Lower Bounds on Learning Decision Lists and Trees. Information and Computation 126(2): 114-122, 1996.

Holte R. C., Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11:63-90, 1993.

Hyafil L. and Rivest R.L., Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 5(1):15-17, 1976

Janikow, C.Z., Fuzzy Decision Trees: Issues and Methods, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 28, Issue 1, pp. 1-14. 1998.

John G. H., Robust linear discriminant trees. In D. Fisher and H. Lenz, editors, Learning From Data: Artificial Intelligence and Statistics V, Lecture Notes in Statistics, Chapter 36, pp. 375-385. Springer-Verlag, New York, 1996.

Kass G. V., An exploratory technique for investigating large quantities of categorical data. Applied Statistics, 29(2):119-127, 1980.

Kearns M. and Mansour Y., A fast, bottom-up decision tree pruning algorithm with near-optimal generalization, in J. Shavlik, ed., 'Machine Learning: Proceedings of the Fifteenth International Conference', Morgan Kaufmann Publishers, Inc., pp. 269-277, 1998.

Kearns M. and Mansour Y., On the boosting ability of top-down decision tree learning algorithms. Journal of Computer and Systems Sciences, 58(1): 109-128, 1999.

Kohavi R. and Sommerfield D., Targeting business users with decision table classifiers, in R. Agrawal, P. Stolorz & G. Piatetsky-Shapiro, eds, 'Proceed-

ings of the Fourth International Conference on Knowledge Discovery and Data Mining', AAAI Press, pp. 249-253, 1998.

Langley, P. and Sage, S., Oblivious decision trees and abstract cases. in Working Notes of the AAAI-94 Workshop on Case-Based Reasoning, pp. 113-117, Seattle, WA: AAAI Press, 1994.

Last, M., Maimon, O. and Minkov, E., Improving Stability of Decision Trees, International Journal of Pattern Recognition and Artificial Intelligence, 16: 2,145-159, 2002.

Li X. and Dubes R. C., Tree classifier design with a Permutation statistic, Pattern Recognition 19:229-235, 1986.

Lim X., Loh W.Y., and Shih X., A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms . Machine Learning 40:203-228, 2000.

Lin Y. K. and Fu K., Automatic classification of cervical cells using a binary tree classifier. Pattern Recognition, 16(1):69-80, 1983.

Loh W.Y.,and Shih X., Split selection methods for classification trees. Statistica Sinica, 7: 815-840, 1997.

Loh W.Y. and Shih X., Families of splitting criteria for classification trees. Statistics and Computing 9:309-315, 1999.

Loh W.Y. and Vanichsetakul N., Tree-structured classification via generalized discriminant Analysis. Journal of the American Statistical Association, 83: 715-728, 1988.

Lopez de Mantras R., A distance-based attribute selection measure for decision tree induction, Machine Learning 6:81-92, 1991.

Lubinsky D., Algorithmic speedups in growing classification trees by using an additive split criterion. Proc. AI&Statistics93, pp. 435-444, 1993.

Martin J. K., An exact probability metric for decision tree splitting and stopping. An Exact Probability Metric for Decision Tree Splitting and Stopping, Machine Learning, 28, 2-3):257-291, 1997.

Mehta M., Rissanen J., Agrawal R., MDL-Based Decision Tree Pruning. KDD 1995: pp. 216-221, 1995.

Mehta M., Agrawal R. and Rissanen J., SLIQ: A fast scalable classifier for Data Mining: In Proc. If the fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, March 1996.

Mingers J., An empirical comparison of pruning methods for decision tree induction. Machine Learning, 4(2):227-243, 1989.

Morgan J. N. and Messenger R. C., THAID: a sequential search program for the analysis of nominal scale dependent variables. Technical report, Institute for Social Research, Univ. of Michigan, Ann Arbor, MI, 1973.

Muller W., and Wysotzki F., Automatic construction of decision trees for classification. Annals of Operations Research, 52:231-247, 1994.

Murthy S. K., Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. Data Mining and Knowledge Discovery, 2(4):345-389, 1998.

Naumov G.E., NP-completeness of problems of construction of optimal decision trees. Soviet Physics: Doklady, 36(4):270-271, 1991.

Niblett T. and Bratko I., Learning Decision Rules in Noisy Domains, Proc. Expert Systems 86, Cambridge: Cambridge University Press, 1986.

Olaru C., Wehenkel L., A complete fuzzy decision tree technique, Fuzzy Sets and Systems, 138(2):221–254, 2003.

Pagallo, G. and Huassler, D., Boolean feature discovery in empirical learning, Machine Learning, 5(1): 71-99, 1990.

Peng Y., Intelligent condition monitoring using fuzzy inductive learning, Journal of Intelligent Manufacturing, 15 (3): 373-380, June 2004.

Quinlan, J.R., Induction of decision trees, Machine Learning 1, 81-106, 1986.

Quinlan, J.R., Simplifying decision trees, International Journal of Man-Machine Studies, 27, 221-234, 1987.

Quinlan, J.R., Decision Trees and Multivalued Attributes, J. Richards, ed., Machine Intelligence, V. 11, Oxford, England, Oxford Univ. Press, pp. 305-318, 1988.

Quinlan, J. R., Unknown attribute values in induction. In Segre, A. (Ed.), Proceedings of the Sixth International Machine Learning Workshop Cornell, New York. Morgan Kaufmann, 1989.

Quinlan, J. R., C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, 1993.

Quinlan, J. R. and Rivest, R. L., Inferring Decision Trees Using The Minimum Description Length Principle. Information and Computation, 80:227-248, 1989.

Rastogi, R., and Shim, K., PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning,Data Mining and Knowledge Discovery, 4(4):315-344, 2000.

Rissanen, J., Stochastic complexity and statistical inquiry. World Scientific, 1989.

Rounds, E., A combined non-parametric approach to feature selection and binary decision tree design, Pattern Recognition 12, 313-317, 1980.

Schlimmer, J. C. , Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In Proceedings of the 1993 International Conference on Machine Learning: pp 284-290, San Mateo, CA, Morgan Kaufmann, 1993.

Sethi, K., and Yoo, J. H., Design of multicategory, multifeature split decision trees using perceptron learning. Pattern Recognition, 27(7):939-947, 1994.

Shafer, J. C., Agrawal, R. and Mehta, M. , SPRINT: A Scalable Parallel Classifier for Data Mining, Proc. 22nd Int. Conf. Very Large Databases, T. M.

Vijayaraman and Alejandro P. Buchmann and C. Mohan and Nandlal L. Sarda (eds), 544-555, Morgan Kaufmann, 1996.

Sklansky, J. and Wassel, G. N., Pattern classifiers and trainable machines. SpringerVerlag, New York, 1981.

Sonquist, J. A., Baker E. L., and Morgan, J. N., Searching for Structure. Institute for Social Research, Univ. of Michigan, Ann Arbor, MI, 1971.

Taylor P. C., and Silverman, B. W., Block diagrams and splitting criteria for classification trees. Statistics and Computing, 3(4):147-161, 1993.

Utgoff, P. E., Perceptron trees: A case study in hybrid concept representations. Connection Science, 1(4):377-391, 1989.

Utgoff, P. E., Incremental induction of decision trees. Machine Learning, 4: 161-186, 1989.

Utgoff, P. E., Decision tree induction based on efficient tree restructuring, Machine Learning 29, 1):5-44, 1997.

Utgoff, P. E., and Clouse, J. A., A Kolmogorov-Smirnoff Metric for Decision Tree Induction, Technical Report 96-3, University of Massachusetts, Department of Computer Science, Amherst, MA, 1996.

Wallace, C. S., and Patrick J., Coding decision trees, Machine Learning 11: 7-22, 1993.

Zantema, H., and Bodlaender H. L., Finding Small Equivalent Decision Trees is Hard, International Journal of Foundations of Computer Science, 11(2): 343-354, 2000.