POLYTECHNIC UNIVERSITY
Department of Computer Science / Finance and Risk Engineering

Decision Trees and Decision Rules

K. Ming Leung

Abstract: The logic-based decision trees and decision rules methodology performs well across a wide range of data mining problems. It also has the advantage that the results can be expressed in terms that the end-user can easily understand.

Directory

- **Table of Contents**
- **Begin Article**

Table of Contents

1. Introduction

1.1. Motivation and Background

The logic-based decision trees and decision rules methodology is the most powerful type of off-the-shelf classifiers that performs well across a wide range of data mining problems. These classifiers adopt a top-down approach and use supervised learning to construct decision trees from a set of given training data set. A decision tree consists of nodes where attributes are tested. The outgoing branches of the node correspond to all the possible outcomes of the test at the node.

For example, consider samples having two features, X and Y, where X has continuous real values and Y is a categorical variable having three possible values, A, B, and C. The figure shows an example of a simple decision tree for classifying these samples. Nodes are denoted by circles, and the decision tree (actually an inverted tree) ends at one of the leaves, denoted by rectangles. Each leaf identifies a particular class. In this example, all samples with features values $X > 1$ and $Y = A$ belong to Class 1. Samples with $X > 1$ and $Y = B$ or C belong to Class 2. Samples with $X \leq 1$ belong to Class

1, regardless of their value for Y.

Notice that in this example, at each node a test is performed based on the value of a single attribute. Decision trees that use univariate splits have a simple representational form, making it easy for the end-user to understand the inferred model. However at the same time, they represent a restriction on the expressiveness of the model and thus the approximation power of the model.

2. C4.5 Algorithm: Generating a Decision Tree

We will consider the C4.5 algorithm for determining the best decision tree based on univariate splits. It works with both categorical and numeric feature values. The algorithm was developed by Ross Quinlan. It is an extension of his earlier ID3 algorithm. He went on to create C5.0 and See5 (C5.0 for Unix/Linux, See5 for Windows) which he markets commercially.

There are two stages of the C4.5 algorithm. The first part, which is discussed in this section, deals with generating the decision tree based on the training data set. The second part has to do with pruning the decision tree based on the validating samples.

We assume that we have a set T of training samples. Let the possible classes be denoted as $\{C_1, C_2, \ldots, C_k\}$. There are three possibilities depending on the content of the set T.

1. T contains one or more samples, all belonging to a single class C_j. The decision tree for T is a leaf identifying class C_j.
2. T contains no samples. The decision tree is again a leaf but the class to be associated with the leaf must be determined from information other than T, such as the overall majority class in T. The C4.5 algorithm uses as a criterion the most frequent class as the parent of the given node.
3. T contains samples that belong to a mixture od classes. It must be refined into subsets of samples that are closer to being a single-class collection of samples. Based on the value of a single attribute, an appropriate test that has a certain number of mutually exclusive outcomes $\{O_1, O_2, \ldots, O_n\}$ is chosen. T is partitioned into subsets $T_1, T_2, \ldots, T_n$, where T_i contains all the samples in T that have outcome O_i of the chosen test. The de-

cision tree for T consists of a decision node identifying the test and one branch for each possible outcome.

The same tree-building procedure is applied recursively to each subset of training samples, so that the i-th branch leads to the decision tree constructed from the subset T_i of training samples. The successive division of the set of training samples proceeds until all the subsets consist of sample belonging to a single class.

2.1. Choice of Test at a Node

The decision tree structure is determined by the tests that we choose to perform at each of the nodes. Different tests, or different order of their application, will yield different trees. The choice of test at a given node is based on information theory to minimize the number of test that will allow a sample to be classified. In other words, we are looking for a compact decision tree that is consistent with the training set.

Exploring all possible trees and selecting the simplest one is infeasible since the problem is NP-complete. Therefore most decision construction methods are non-backtracking, greedy algorithms. At any given node, the algorithm used by C4.5 basically chooses to test the attribute that provides the maximum degree of discrimination between classes locally.

Suppose we have the task of selecting a possible test with n outcomes (n values for a given categorical feature) that partition the set T of training samples into subsets $\{T_1, T_2, \dots, T_n\}$. The only information available for guidance is the distribution of classes in T and in its subsets T_i. If S is any set of samples, let $\mathrm{freq}(C_i, S)$ stand for the number of samples in S that belong to class C_i, and let $|S|$ denote the number of samples in the set S. The entropy of the set S is defined as

$$\mathrm{Info}(S) = -\sum_{i=1}^{k} \frac{\mathrm{freq}(C_i, S)}{|S|} \log_2 \left(\frac{\mathrm{freq}(C_i, S)}{|S|} \right).$$

The unit for entropy is bits.

Now we can measure the information content of T by computing $\mathrm{Info}(T)$. We can also compute the total information content after T has been partitioned in accordance with the outcomes of a chosen attribute test, say x. It is given by the weighted sum of the entropies of each of the subsets:

$$\mathrm{Info}_x(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|}\,\mathrm{Info}(T_i).$$

The quantity

$$\mathrm{Gain}(x) = \mathrm{Info}(T) - \mathrm{Info}_x(T)$$

measures the information that is gained by partitioning T in accordance with the test based on x. The criterion is to select a test x to maximize $\mathrm{Gain}(x)$ *i.e.* with the highest information gain.

2.2. Dealing with Features with Numeric Values

The above works with categorical attributes. How can one deal with features that have numeric values? The C4.5 algorithm performs test on numeric features by comparing the values to a certain threshold value, z. If Y is such a numeric feature, and if it is chosen at a given node, then the samples will be partitioned into two groups: those whose values for Y are less than or equal to z, and those whose values are greater than z. The algorithm chooses the threshold z as follows. The values of the attribute Y in the training set are first sorted in ascending order as $\{v_1, v_2, \ldots, v_n\}$. Any threshold value lying between v_i and v_{i+1} will divide the samples into those whose value for Y lies in $\{v_1, v_2, \ldots, v_i\}$ and those whose values are in $\{v_{i+1}, v_{i+2}, \ldots, v_n\}$. Thus there are $m-1$ possible splits on Y, all of which should be used to compute the potential information gain to determine an optimal split.

It is common to choose the midpoint of each interval , $(v_i+v_{i+1})/2$, as the representative threshold value. The C4.5 algorithm chooses as the threshold the smaller value v_i for every interval $\{(v_i, v_{i+1}\}$, rather than the midpoint. This ensures that the threshold value appearing in either the final decision tree or rules actually occur in the original data set.

2.3. An Example: Generating a Decision Tree

We now consider as an example the training set T given in the following table. T has 14 samples, and each sample has 3 attributes and belongs to one of 2 possible classes. Attributes x_1 and x_3 have nominal values, while attribute x_2 has numeric values.

Sample	x_1	x_2	x_3	Class
1	A	70	true	C_1
2	A	90	true	C_2
3	A	85	false	C_2
4	A	95	false	C_2
5	A	70	false	C_1
6	B	90	true	C_1
7	B	78	false	C_1
8	B	65	true	C_1
9	B	75	false	C_1
10	C	80	true	C_2
11	C	70	true	C_2
12	C	80	false	C_1
13	C	80	false	C_1
14	C	96	false	C_1

Starting at the root level, given this training set T, we need to determine which of the three attributes should be tested to form the node at the root. Since 9 samples belong to class C_1 and the remaining 5 samples to C_2, the entropy before splitting is [4]

$$\text{info}(T) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}.$$

First we consider using attribute x_1 to split T into 3 subsets T_1, T_2, and T_3, containing samples with x_1 equal to A, B, and C, respectively. T_1 has 5 samples, 2 are in C_1 and 3 in C_2, and so its entropy is

$$\text{Info}(T_1) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971 \text{ bits}.$$

T_2 has 4 samples, all are in C_1 and none in C_2, and so its entropy is

$$\text{Info}(T_2) = -\frac{4}{4}\log_2\left(\frac{4}{4}\right) - \frac{0}{4}\log_2\left(\frac{0}{4}\right) = 0 \text{ bits}.$$

There is no randomness in T_2 since all samples belong to C_1. Lastly, T_3 has 5 samples, 3 are in C_1 and 2 in C_2, and so its entropy is

$$\text{Info}(T_3) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971 \text{ bits}.$$

Thus after this potential split, the resulting entropy is

$$\mathrm{Info}_{x_1}(T) = \frac{5}{14}\mathrm{Info}(T_1) + \frac{4}{14}\mathrm{Info}(T_2) + \frac{5}{14}\mathrm{Info}(T_3) = 0.694 \text{ bits.}$$

The information gain (or loss in entropy) if the set is split using attribute x_1 is

$$\mathrm{Gain}(x_1) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Next we consider using attribute x_2 to split T. We need to handle the numeric values associated with x_2. Since we are interested to find the optimal threshold value, we only need to sort the distinct values for x_2 in the set T. The result is $\{65, 70, 75, 78, 80, 85, 90, 95, 96\}$. The potential threshold values z is $\{65, 70, 75, 78, 80, 85, 90, 95\}$ (by taking the left end point of each interval). For every one of these values, we use it to split T into 2 subsets, and compute the resulting information gain. We choose the threshold value to be the one that gives the highest information gain. It turns out that the optimal threshold value is $z = 80$. This threshold value partition T into 2 subsets T_1 and T_2, containing samples with $x_2 \leq 80$ and with $x_2 > 80$, respectively.

T_1 as 9 samples, 7 are in C_1 and 2 in C_2, and so its entropy is

$$\mathrm{Info}(T_1) = -\frac{7}{9}\log_2\left(\frac{7}{9}\right) - \frac{2}{9}\log_2\left(\frac{2}{9}\right) = 0.764 \text{ bit.}$$

T_2 has 5 samples, 2 are in C_1 and 3 in C_2, and so its entropy is

$$\mathrm{Info}(T_2) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971 \text{ bits.}$$

Thus after this potential split at x_2, the resulting entropy is

$$\mathrm{Info}_{x_2}(T) = \frac{9}{14}\mathrm{Info}(T_1) + \frac{5}{14}\mathrm{Info}(T_2) = 0.838 \text{ bits.}$$

Therefore the information gain (or loss in entropy) if the set is split using attribute x_2 is

$$\mathrm{Gain}(x_2) = 0.940 - 0.838 = 0.102 \text{ bits.}$$

Finally we consider using attribute x_3 to split T into 2 subsets T_1 and T_2, containing samples with x_3 equal to True, and False, respectively. T_1 has 6 samples, 3 are in C_1 and 3 in C_2, and so its entropy is

$$\text{Info}(T_1) = -\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right) = 1 \text{ bit}.$$

T_2 has 8 samples, 6 are in C_1 and 2 in C_2, and so its entropy is

$$\text{Info}(T_2) = -\frac{6}{8}\log_2\left(\frac{6}{8}\right) - \frac{2}{8}\log_2\left(\frac{2}{8}\right) = 0.811 \text{ bits}.$$

Thus after this potential split at x_3, the resulting entropy is

$$\text{Info}_{x_3}(T) = \frac{6}{14}\text{Info}(T_1) + \frac{8}{14}\text{Info}(T_2) = 0.892 \text{ bits}.$$

Therefore the information gain (or loss in entropy) if the set is split using attribute x_3 is

$$\text{Gain}(x_3) = 0.940 - 0.892 = 0.048 \text{ bits}.$$

By comparing the information gain for the three attributes, we see that x_1 gives the highest gain of 0.246 bits, and therefore this attribute is selected for the first splitting in the construction of a decision tree. The root node will test for the value of x_1, and three branches will be created, one for each attribute values. Samples having those corresponding values are passed to each of these branches as subsets T_1, T_2, and T_3. The entire process of test selection and optimization will be repeated for every child node.

Next, we consider each of the three subnodes separately. We first consider splitting T_1 which has 2 C_1 samples and 3 C_2 samples, and so its entropy is

$$\text{Info}(T_1) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971 \text{ bits}.$$

If we choose to test attribute x_1, it turns out the optimal threshold value is $z = 70$. Let us denote this optimal test as x_4. This choice of z splits T_1 into 2 subsets. The first subset, consisting of 2 samples with $x_2 \leq 70$, has all the 2 samples in C_1 and none in C_2. The second subset, consisting of 3 samples with $x_2 > 70$, has all the 3 samples in

C_2 and none in C_1. The resulting information is

$$\begin{aligned}\text{Info}_{x_4}(T_1) &= \frac{2}{5}\left[-\frac{2}{2}\log_2\left(\frac{2}{2}\right)-\frac{0}{2}\log_2\left(\frac{0}{2}\right)\right]\\ &+ \frac{3}{5}\left[-\frac{0}{3}\log_2\left(\frac{0}{3}\right)-\frac{3}{3}\log_2\left(\frac{3}{3}\right)\right]\\ &= 0 \text{ bits.}\end{aligned}$$

The information gained by this test is

$$\text{Gain}(x_4) = 0.971 - 0 = 0.971 \text{ bits.}$$

The 2 branches created by this split will be the final leaf nodes since the subsets of samples in each of the branches all belong to their separate classes.

Actually there is no need to search for a better split since we know that x_4 provides the highest information gain. Here, just for fun we consider choosing to split T_1 using attribute x_3. This test on x_3, which we denote by x', divides T_1 into 2 subsets. The first subset, consisting of 2 samples with $x_3 = True$, has 1 sample in C_1 and 1 in C_2. The second subset, consisting of 3 samples with $x_3 = False$, has all the 1 sample in C_1 and 2 in C_2. The resulting information is

$$\begin{aligned}\text{Info}_{x'}(T_1) &= \frac{2}{5}\left[-\frac{1}{2}\log_2\left(\frac{1}{2}\right)-\frac{1}{2}\log_2\left(\frac{1}{2}\right)\right]\\ &+ \frac{3}{5}\left[-\frac{1}{3}\log_2\left(\frac{1}{3}\right)-\frac{2}{3}\log_2\left(\frac{2}{3}\right)\right]\\ &= 0.951 \text{ bits.}\end{aligned}$$

The information gained by this test turns out to be rather small

$$\text{Gain}(x') = 0.971 - 0.951 = 0.020 \text{ bits.}$$

In any case, the conclusion is that T_1 should be split according to test x_4. This subnode will then branch into 2 final leaf nodes, one for each of the 2 classes.

Next we consider splitting T_2. However since all 4 samples in T_2 belong to C_1, thus this node will be a leaf node, and no additional tests are necessary for this branch.

We now turn to the last subset T_3, whose entropy is

$$\text{Info}(T_3) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right)-\frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971 \text{ bits.}$$

It is clear that it will be better to test on attribute x_3 which split T_2 into 2 subsets. The first subset, consisting of 2 samples with $x_3 = True$, has no sample in C_1 and all 2 samples in C_2. The second subset, consisting of 3 samples with $x_3 = False$, has all the 3 samples in C_1 and none in C_2. The resulting information is

$$\begin{aligned}\mathrm{Info}_{x_5}(T_3) \quad = \quad & \frac{2}{5}\left[-\frac{2}{2}\log_2\left(\frac{0}{2}\right) - \frac{2}{2}\log_2\left(\frac{2}{2}\right)\right] + \\ & \frac{3}{5}\left[-\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right] = 0 \text{ bits},\end{aligned}$$

where we have denoted this test as x_5. The information gained by x_5

$$\mathrm{Gain}(x_5) = 0.971 - 0 = 0.971 \text{ bits}$$

is clearly the best. This test results in 2 uniform subsets of samples of the 2 separate classes, and therefore yields 2 final leaf nodes for this branch.

The final decision tree for T is now determined. It can then be used to classify any new unseen sample.

The resulting tree can be represented in the form of pseudocode with if-then-else constructions for branching into a tree structure. Our decision above can be represented as

```
If (attribute1 = A) Then
    If (attributes <= 70) Then
        Classification = Class1
    Else
        Classification = Class2
    EndIf
ElseIf (attribute = B) Then
        Classification = Class1
Else
    If (attribute3 = True) Then
        Classification = Class2
    Else
        Classification = Class1
    EndIf
EndIf
```

The manner in which classification decision is made according to the decision tree is now expressed in plain English and can be easily understood by the end-user.

2.4. The Split-Information parameter

The information-gain criterion unfortunately has a serious deficiency in that there is a strong bias in favor of tests with a lot of outcomes. A remedy comes in the form of a proper normalization. In analogy with the definition of the entropy of a set, an additional parameter is introduced for each test x that splits a given set T into subsets $T_i, i = 1, 2, \ldots, n$:

$$\text{Split-Info}(x) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \log_2 \left(\frac{|T_i|}{|T|} \right).$$

This represents the potential information generated by dividing set T into n subsets T_i. Now, a new gain measure can be defined:

$$\text{Gain-ratio}(x) = \frac{\text{Gain}(x)}{\text{Split-info}(x)}.$$

This new gain measure expresses the proportion of information generated by the split that appears useful in classification. The gain-ratio criterion the selects a test that maximizes this ratio. This criterion is robust and typically gives a consistently better choice of a test than the previous gain criterion.

For example, if we go back to the test x_1 that splits T into 3 subsets, the additional parameter is

$$\begin{aligned}\text{Split-Info}(x_1) &= -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{4}{14}\log_2\left(\frac{4}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)\\ &= 1.577\text{bits}\end{aligned}$$

this gives

$$\text{Gain-ratio}(x_1) = \frac{0.246}{1.557} = 0.156.$$

Of course a similar procedure should be performed for all the other test in the decision tree to obtain the final decision tree. It turns out that for this example, this modification in the algorithm does not alter the decision tree.

2.5. Handling Unknown Attribute Values

The algorithm here can be modified to handle samples with missing attribute values. It is assumed that these missing values are distributed probabilistically according to the relative frequency of known values of the other samples.

Info(T) and Info$_x(T)$ are calculated as before, except that only samples with known attribute values are taken into account. Then the gain parameter is corrected with a factor F,which represents the probability that a given attribute is known. Specifically F is define as the ratio of the number of samples in the database with a known value for a given attribute and the total number of samples in a data set. The new gain criterion now has the form

$$\text{Gain}(x) = F\left(\text{Info}(T) - \text{Info}_x(T)\right).$$

Similarly, Split-info(x) is altered by regarding the samples with unknown values as an additional group in splitting. If the test x has n outcomes, its Split-info(x) is computed as if the test divided the data set into $n+1$ subsets. This modification has a direct influence on the final value of the modified criterion Gain-ratio(x).

2.6. Pruning Decision Trees

Generating a decision to function best with a given of training data set often creates a tree that over-fits the data and is too sensitive on the sample noise. Such decision trees do not perform well with new unseen samples.

We need to prune the tree in such a way to reduce the prediction error rate. Pruning a decision tree means that one or more subtrees are discarded and replaced with leaves thus simplifying the tree. One possibility to estimate the prediction error rate is to use the cross-validation techniques. This technique divides initially available samples into roughly equal-sized blocks and, for each block, the tree is constructed from all sample except this block and tested with that block of samples. With the available training and testing samples, the basic idea is to remove parts of the tree that do not contribute to the classification accuracy of unseen testing samples. producing a less complex and thus more comprehensible tree.

2.7. Advantages and disadvantages

Decision trees in general has several important advantages:

1. creating decision trees need no tuning parameters
2. no assumptions about distribution of attribute values or independence of attributes
3. no need for transformation of variables (any monotonic transformation of the variable will result in the same trees)
4. the method automatically finds a subset of the features that are relevant to the classification
5. decision trees are robust to outliers as the choices of a split depends on the ordering of feature values and not on the absolute magnitudes of these values
6. it can easily be extended to handle samples with missing values

Decision trees also has several important disadvantages:

1. If data samples are represented graphically in an n-dimensional space, then a decision tree divides the space into regions. each

region is labeled with a corresponding class. An unseen sample is classified by determining the region in which the given lies. Decision tree is constructed by successive refinement, splitting existing regions into smaller ones that contain highly concentrated points of one class. The number of training samples needed to construct a good classifier is proportional to the number of regions.

2. Decision rules yield orthogonal hyperplanes in the n-dimensional space, thus each region has the form of a hyper-rectangle. But if in reality many of these decision hyperplanes are not perpendicular to the coordinates (because certain deciding factors are the results of combinations of different attributes), decision trees and rules tend to be much more complex. Of course a solution is to better transform the data in the pre-processing step.

3. There is a type of classification problem where the classification criterion has the form: a given class is supported if n out of m conditions are present. Decision trees are not the appropriate tool for modeling this type of problems. Medical diagnostic

decisions are a typical example of this kind of classification. If 4 out of 11 symptoms support diagnosis of a given disease, then the corresponding classifier will generate 330 regions in an 11-dimensional space for positive diagnosis only. That corresponds to 330 decision rules.

References

[1] M. Kantardzic, *Data Mining - Concepts, Models, Methods, and Algorithms*, IEEE Press, Wiley-Interscience, 2003, ISBN 0-471-22852-4. This is our adopted textbook, from which this set of lecture notes are derived primarily.

[2] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[3] J. R. Quinlan, "Improved use of continuous attributes in c4.5", *Journal of Artificial Intelligence Research*, 4:77-90, 1996.

[4] Computationally there is actually no need to know the information content before performing any test to attempt to split the samples

in a group. Finding a test x that maximizes

$$\text{Gain}(x) = \text{Info}(T) - \text{Info}_x(T)$$

is the same as finding a test x that minimizes $\text{Info}_x(T)$.

11