6 Decision Trees

Learning Objectives

After reading this chapter, you will understand

- How decision trees are used to choose the course of action
- How decision trees are used for classification
- The strengths and weakness of decision trees
- The splitting criterion used at the nodes
- What is meant by induction of decision trees
- Why pruning of decision trees is sometimes necessary

The decision tree is one of the most popularly used data structures in pattern classification. This is because of its highly transparent and user-friendly characteristics.

6.1 Introduction

A *decision tree* is a tree where each non-leaf or internal node is associated with a decision and the leaf nodes are generally associated with an outcome or class label. Each internal node tests one or more attribute values leading to two or more links or branches. Each link in turn is associated with a possible value of the decision. These links are mutually distinct and collectively exhaustive. This means that it is possible to follow only one of the links and all possibilities will be taken care of—there is a link for each possibility.

Decision trees are excellent tools for choosing between several courses of action. They provide a highly effective structure within which you can lay out options and investigate the possible outcome of choosing these options.

In the case of a binary decision tree, each node gives the statement of the decision to be taken or the comparison to be made. There are two outgoing edges from the nodes. One edge represents the outcome "yes" or "true" and the other edge, the outcome "no" or "false".

Example 1

There are four coins a, b, c, d out of which three coins are of equal weight and one coin is heavier. Find the heavier coin.

Figure 6.1 gives the decision tree corresponding to this problem. The root node compares the weight of $a+b$ with $c+d$. If $a+b$ is heavier than $c+d$, then the outcome is "yes" and it takes the left branch. Otherwise, $c+d$ is heavier and it takes the right branch. The node in the left branch compares the weight of a with b. If the outcome of this comparison is "yes", then it chooses a as the heavier coin. If the outcome is "no", then it chooses b as the heavier coin. This same procedure is done with c and d if the outcome is "no" for the root node.

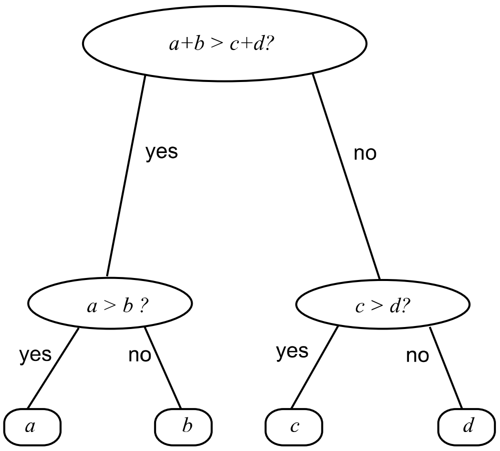


Figure 6.1 Decision tree

The example depicted in Figure 6.1 represents a simple decision tree in the context of decision making. Some of the points illustrated are:

1. There are *four leaf nodes* corresponding to the *four possible outcomes*. Each leaf node corresponds to one of the coins being heavier.
2. It requires two weighings to make the final decision or to reach a leaf node. Note that each decision node (non-terminal or internal node) corresponds to a weighing operation. Further, there are two decision nodes from the root to any leaf.
3. Each path from the root to a leaf corresponds to a rule. For example, the rule corresponding to the left-most path is "if $a+b > c+d$ and if $a > b$ then a is heavier".

In pattern classification, an internal node in the decision tree is associated with a test based on the values of one or more features. These tests may be categorised as follows:

1. *Axis-parallel test* In this case, the test is of the form $x > a_0$, where x is a feature and a_0 is a threshold value. For example, "height" > 5 ft tests whether the height of an object is above 5 feet or not. In other words, this test is associated with a hyper-plane parallel to all the feature axes other than that of "height" which splits the pattern space into two parts—one part corresponds to patterns having "height" greater than 5 feet and the other part has patterns of "height" less than or equal to 5 feet. Essentially, this test involves only one feature. Such a split is called the *axis-parallel split*.
2. *Test based on linear combination of features* Here, the test is of the form

   $$\sum_{i=1}^{d} a_i x_i > a_0$$

   where x_i is the ith feature and a_i is the weight associated with it. This test involves a linear combination of feature values and the corresponding hyper-plane need not be parallel to any axis. For example, "0.4 height + 0.3 weight > 38" tests whether a linear combination of "height" and "weight" of an object exceeds 38. This test splits the space into two parts based on an *oblique* split. Note that the axis-parallel test is a special case of the oblique split where a_i is 0 for all but one i.
3. *Test based on a non-linear combination of features* This is the most general form of the test possible and it is of the form

   $$f(x) > 0$$

   where $f(x)$ is any non-linear function of the components of x. It is easy to see that axis-parallel and oblique splits are special cases of this test. However, this form of the test is computationally the most expensive.

It is possible to employ different types of tests at different nodes of a decision tree. However, it is computationally prohibitive to explore this possibility on large data sets. We consider only the axis-parallel and oblique tests in this chapter as they are relatively simpler to implement.

6.2 Decision Trees for Pattern Classification

Patterns can be classified using decision trees where the nodes in the tree represent the status of the problem after making some decision. The leaf nodes give the class label of the classification rule corresponding to the path from the root node to that leaf node.

EXAMPLE 2

Let us consider a set of animals described in Table 6.1. Figure 6.2 gives a decision tree for the classification of the set of animals.

Table 6.1 Description of a set of animals

	Legs	Horns	Size	Colour	Beak	Sound
Cow	4	yes	medium	white	no	moo
Crow	2	no	small	black	yes	caw
Elephant	4	no	large	black	no	trumpet
Goat	4	yes	small	brown	no	bleat
Mouse	4	no	small	black	no	squeak
Parrot	2	no	small	green	yes	squawk
Sparrow	2	no	small	brown	yes	chirp
Tiger	4	no	medium	orange	no	roar

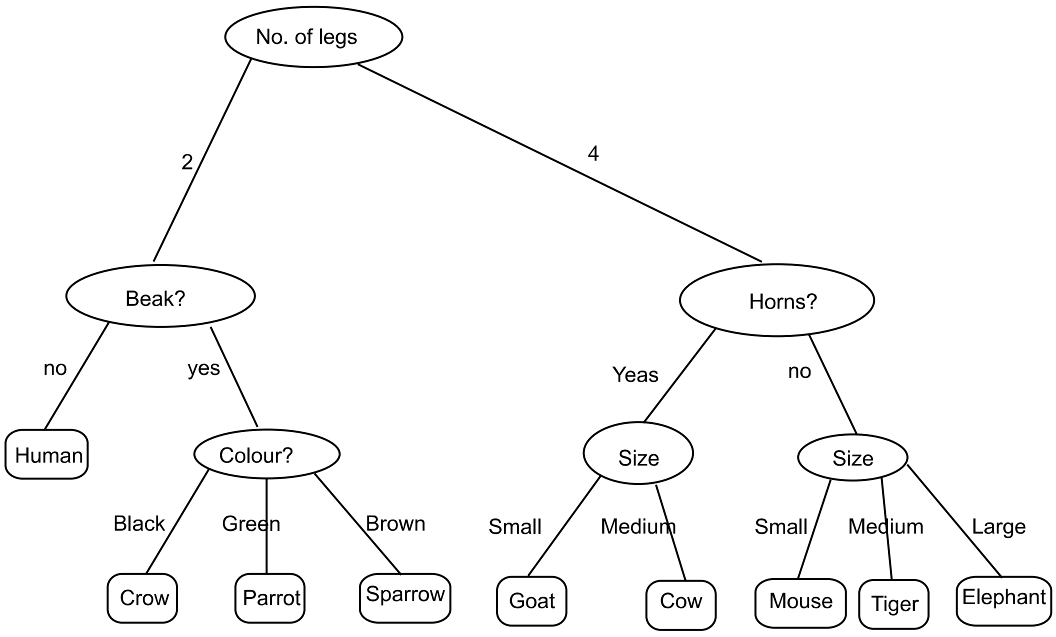


Figure 6.2 Decision tree for classification of animals

Looking at the decision tree, we can make the following observations:

- *Class labels are associated with leaf nodes* In general, leaf nodes are associated with class labels. In Figure 6.2, the leaf nodes are associated with animal names.
- *Root to leaf represents a rule* For example, looking at the decision tree in Figure 6.2, a classification rule would be:

  if (no. of legs = 4) and (has horns = false) and (size = small) then (mouse), where "mouse" is the class label.
- *Classification involves making a decision at every node* Classification involves making a decision at each node and moving down the appropriate branch till we reach a leaf node where the class label is determined.
- *Irrelevant features* Features that are not relevant to the classification problem at hand do not occur in the decision tree. In Table 6.1, the sound made by the animals is an attribute but does not occur in the decision tree, since it is not needed for the discrimination of the patterns.
- *Numerical and categorical features* Both numerical and categorical features can be used. We can see from the example that the colour is a categorical feature, whereas the number of legs is a numerical feature.
- *The tree can be binary or non-binary* We could have two decisions say "yes" and "no" at each node or we can have a k-ary tree, where a "many-way decision" may be made at a node. The colour feature has a number of branches depending on the colours considered. The horns feature has just two decisions—"yes" or "no". A point to be noted here is that a many-way decision can be converted into a number of yes/no decisions and therefore *a binary tree is sufficient to represent a decision tree*.
- *A set of patterns is associated with each node* This set of patterns is larger at the top nodes and keeps reducing in size at subsequent levels. In Figure 6.2, once the decision is made on the number of legs, the node reached when "four legs" is the decision will contain only patterns which exclude birds and human beings.
- *The rules are simple and easy to understand* It consists of a conjunction of a number of antecedents and a single outcome. Each antecedent consists of a simple boolean function.

The animal classification problem is not a typical case of pattern classification since each pattern in the database refers to a different class. A more typical example is given below.

EXAMPLE 3

Figure 6.3 pertains to the output of a manager of a company. There are three managers, Ram, Shyam and Mohan. Table 6.2 gives the employee records of these three managers.

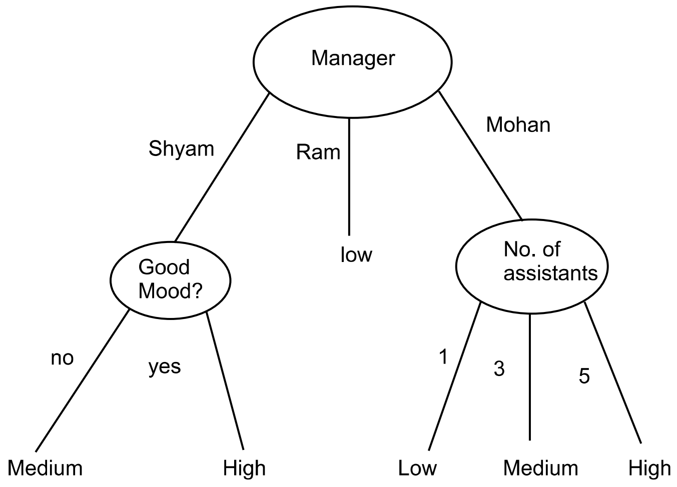


Figure 6.3 Decision tree for pattern classification

Table 6.2 Employee records in a company

Name	Age	Educational qualification	Position
Ram	55	B Com	Manager
Shyam	30	BE	Manager
Mohan	40	MSc	Manager

The decision tree in Figure 6.3 may have been induced from the patterns in Table 6.3.

Table 6.3 Patterns from which the decision tree in Figure 6.3 is induced

Manager	Number of assistants	Mood	Output
Shyam	3	No	Medium
Shyam	5	No	Medium
Shyam	1	Yes	High
Ram	1	Yes	Low
Ram	5	No	Low
Ram	5	Yes	Low
Mohan	1	No	Low
Mohan	3	Yes	Medium
Mohan	5	No	High

The output of these managers is the class labels. There are three classes: low (output), medium (output) and high (output). "Manager" is a categorical attribute since it pertains to the name of the manager. The "number of assistants" is a numerical attribute while "mood" is a boolean attribute which takes the values "yes" and "no". The points to be noted here are

1. *Class labels are associated with leaf nodes* The leaf nodes give one of the classes: low, medium or high.
2. *Root to leaf represents a rule* For example, looking at the decision tree in Figure 6.3, a classification rule would be

   if (Manager = Mohan) and (No. of assistants = 3) then (Output = medium)
3. *Every node involves making a decision* Every node involves making a decision which lead to two or more branches.
4. *Irrelevant features* Features that are not relevant to the classification problem at hand do not occur in the decision tree. For example, if each pattern had a feature which is the age of the manager, this feature would not appear in the decision tree—they are eliminated.
5. *Numerical and categorical features* Both numerical and categorical features appear in the decision tree.
6. *Classification is simple and efficient* It involves as many comparisons as the number of nodes, from root to leaf, in the branch followed.

Weaknesses of Decision Trees

- *Design time could be large* For example, the number of oblique splits possible could be exponential in the number of training patterns. At a single node of the tree, the complexity of selecting an optimal oblique hyper-place is exponential. If there are n training instances, each having d attributes, there are at most $2^d \times \begin{pmatrix} n \\ d \end{pmatrix}$ distinct d-dimensional oblique splits. In other words, hyper-planes can dichotomise a set of n d-dimensional vectors in at most $2 \times \sum_{k=0}^{d} \begin{pmatrix} n-1 \\ d \end{pmatrix}$ ways if $n > d+1$ and 2^n ways if $n \leq d+1$. This problem is exponential in the number of training patterns.
- *Simple (axis-parallel) decision trees do not represent non-rectangular regions well* Most decision tree algorithms only examine a single attribute at a time. This leads to rectangular classification that may not correspond well with the actual distribution of the data. In the simple decision tree, all the decision boundaries

are hyper-planes orthogonal to the axis of the feature being considered. When the regions are non-rectangular, the decision tree approximates the regions by hyper-rectangles.

EXAMPLE 4

Consider the example given in Figure 6.4. Here the decision boundary is non-rectangular. Figure 6.5 shows the approximate division performed by a simple decision tree.

- *Exponentially large decision trees are needed for some functions like the parity function and majority function*

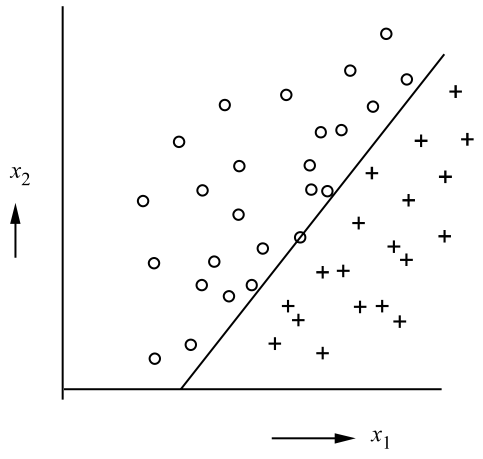


Figure 6.4 Non-rectangular decision boundary

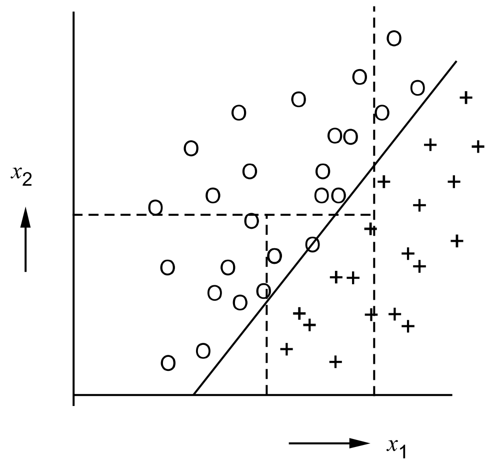


Figure 6.5 Decision tree performed on a non-rectangular region

Example 5

A parity function takes input $x \in \{0, 1\}^d$ and outputs 1 if the number of 1s in x is odd, 0 otherwise. It is evident that it is necessary to use all the d bits to decide the output. This makes the decision tree very large.

6.3 Construction of Decision Trees

A decision tree is induced from examples. We have a set of labelled patterns which forms the training set. The tree is constructed so that it agrees with the training set. The decision tree corresponding to a training set helps us to describe a large number of cases in a concise way. It is an example of inductive learning. To construct the decision tree, one or more attributes have to be chosen at each node to make a decision. The most important attribute is used at the earliest level in the decision tree. This attribute is that which makes the most difference to the classification; it is the most discriminative attribute. At each node, the set of examples is split up and each outcome is a new decision tree learning problem by itself. A really good attribute divides the examples into sets that are all positive or all negative. An attribute that leaves the example set with roughly the same proportion of positive and negative examples as the original is not good. In other words, if many of the outcomes result in a definitive answer, it is a good attribute to choose.

Example 6

In Figure 6.6, the decision $f_1 \geq a$ divides both class 1 and class 2 so that all the patterns of each class is on one side of the boundary. The decision $f_2 \geq b$ divides the patterns of both Class 1 and Class 2 so that there are two patterns on one side and two patterns on the other side. So it can be seen that the decision $f_1 \geq a$ is a better option as it directly classifies the patterns as belonging to Class 1 or Class 2.

Once we have chosen the correct attribute, the different outcomes represent new decision trees and, in each case, the most important attribute is chosen. This is done till all nodes give a final classification.

At each node, the query which makes data to the subsequent nodes as *pure* as possible is chosen. Thus, instead of measuring how pure the node is, we minimise the impurity of the resulting nodes. There are many measures of impurity which we shall discuss below.

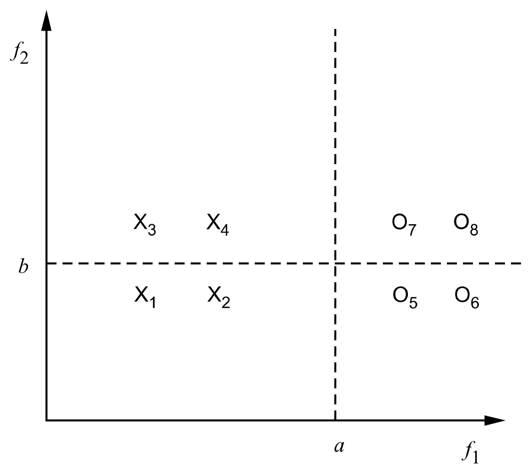


Figure 6.6 Classification into two classes: An illustration

6.3.1 Measures of Impurity

1. *Entropy impurity or information impurity*

   The entropy impurity at a node N is $i(N)$ and is given by

   $$i(N) = -\sum_j P(w_j)\log_2 P(w_j)$$

   Here $P(w_j)$ is the fraction of patterns at node N of category w_j.

   EXAMPLE 7

   Consider the case when the patterns are split equally into two sub-sets. Then $P(w_j)$ is 0.5 and we get

   $$i(N) = -0.5\log 0.5 - 0.5\log 0.5 = 1$$

   When all the patterns go along one branch and there are no patterns in the other branch, $i(N) = 0$. Consider 10 examples which split into three classes. The first class gets 4 examples, the second class gets 5 and the third class gets 1 example.

   $$P(w_1) = \tfrac{4}{10} = 0.4$$
   $$P(w_2) = \tfrac{5}{10} = 0.5$$
   $$P(w_3) = \tfrac{1}{10} = 0.1$$

   and $i(N) = -0.4\log 0.4 - 0.5\log 0.5 - 0.1\log 0.1 = 1.36$

2. *Variance impurity*

   $i(N) = P(w_1)P(w_2)$ in a two-category case.

When $P(w_1)$ is 0.5 and $P(w_2)$ is 0.5, $i(N) = 0.25$. When $P(w_1)$ is 1.0 and $P(w_2)$ is 0, $i(N) = 0$

Generalising this to more categories, we get the *Gini impurity.*

$$i(N) = \sum_{i \neq j} P(w_i)P(w_j) = \frac{1}{2}[1 - \sum_j P^2 i(w_j)]$$

EXAMPLE 8

Again, let us we consider 10 examples which split into three classes. The first class gets 4 examples, the second class gets 5 and the third class gets 1 example. We get

$$P(w_1) = \tfrac{4}{10} = 0.4$$

$$P(w_2) = \tfrac{5}{10} = 0.5$$

$$P(w_3) = \tfrac{1}{10} = 0.1$$

We can then calculate the Gini impurity as

$$i(N) = \tfrac{1}{2}[1 - 0.4^2 - 0.5^2 - 0.1^2] = 0.29$$

3. *Misclassification impurity*

$$i(N) = 1 - \max_j P(w_j)$$

EXAMPLE 9

In the example we have considered with three branches, the misclassification impurity will be

$$i(N) = 1 \max(0.4, 0.5, 0.1) = 0.5$$

This measures the minimum probability that a training pattern would be misclassified.

6.3.2 Which Attribute to Choose?

The attribute to be chosen should decrease the impurity as much as possible. The drop in impurity is defined as

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

This is in the case when there are only two branches at the node. Here P_L and N_L

correspond to the left branch and N_R corresponds to the right branch. If there are more than two branches, we get

$$\Delta i(N) = i(N) - \sum_j P_j \times i(N_j)$$

j takes on the value for each of the outcomes of the decision made at the node. $\Delta i(N)$ can also be called the gain in information at the node.

The attribute which maximises $\Delta i(N)$ is to be chosen.

Example 10

Consider a case where there are 100 examples with 40 belonging to C_1, 30 belonging to C_2 and 30 belonging to C_3. Let some attribute X split these examples into two branches, the left branch containing 60 examples and the right branch containing 40 examples. The left branch contains 40 examples of C_1, 10 examples of C_2 and 10 examples of C_3. The right branch contains 0 examples of C_1, 20 examples of C_2 and 20 examples of C_3. This is shown in Table 6.4.

Table 6.4 Number of examples in the decision tree after a split using feature X

X = a left branch	X = b right branch	Total	Class
40	0	40	1
10	20	30	2
10	20	30	3

Using entropy impurity

The entropy impurity at the split is

$$\begin{aligned} i(N) &= -\frac{40}{100}\log\frac{40}{100} - \frac{30}{100}\log\frac{30}{100} - \frac{30}{100}\log\frac{30}{100} \\ &= -0.4\log 0.4 - 0.3\log 0.3 - 0.3\log 0.3 = 1.38 \end{aligned}$$

The entropy of the left branch $i(N_L)$ is

$$i(N_L) = -\frac{40}{60}\log\frac{40}{60} - \frac{10}{60}\log\frac{10}{60} - \frac{10}{60}\log\frac{10}{60} = 1.25$$

The entropy of the right branch $i(N_R)$ is

$$i(N_R) = -\frac{20}{40}\log\frac{20}{40} - \frac{20}{40}\log\frac{20}{40} = 1.0$$

The drop in impurity is therefore

$$\Delta i(N) = 1.38 - (\frac{60}{100} \times 1.25) - (\frac{40}{100} \times 1.0)$$
$$= 1.38 - 0.75 - 0.4 = 0.23$$

Using variance impurity

The impurity at the node is

$$i(N) = \frac{1}{2}(1 - 0.4^2 - 0.3^2 - 0.3^2)$$
$$= 0.5 \times 0.66 = 0.33$$

The impurity at the left node is

$$i(N_L) = \frac{1}{2}(1 - 0.6667^2 - 0.1667^2 - 0.1667^2)$$
$$= 0.5 \times 0.5 = 0.25$$

The impurity at the right node is

$$i(N_R) = \frac{1}{2}(1 - 0.5^2 - 0.5^2)$$
$$= 0.5 \times 0.5 = 0.25$$

The drop in impurity is therefore

$$\Delta i(N) = 0.33 - (\frac{60}{100} \times 0.25) - (\frac{40}{100} \times 0.25) = 0.08$$

Using misclassification impurity

The impurity at the node is

$$i(N) = 1 \max(0.4, 0.3, 0.3) = 0.6$$

Impurity at the left node is

$$i(N_L) = 1 \max(\frac{40}{60}, \frac{10}{60}, \frac{10}{60})$$
$$= 1 \max(0.6667, 0.1667, 0.1667) = 0.333$$

Impurity at the right node is

$$i(N_L) = 1\max(\frac{20}{40}, \frac{20}{40})$$
$$= 1\max(0.5, 0.5) = 0.5$$

The drop in impurity is therefore

$$\Delta i(N) = 0.6 - (\frac{60}{100} \times 0.33) - (\frac{40}{100} \times 0.5)$$
$$= 0.6 - 0.198 - 0.2 = 0.202$$

6.4 Splitting at the Nodes

Each decision outcome at a node is called a split since the training data is split into sub-sets. The root node splits the full training data. Each successive decision splits the data at the node into proper sub-sets.

The decision rule at each non-leaf node is of the form

$$f_i(x) > a_0$$

This is called the split rule or test at the node. Depending on this split rule, we can have different types of splitting.

1. *Axis-parallel split* It pertains to splitting depending on the value of a single attribute and is of the form

   $$x_j > a_0$$

   Here the decision is taken depending on the attribute x_j. If it is greater than a_0, one branch is taken, otherwise the other branch is taken.

   EXAMPLE 11

   Axis parallel split is illustrated in Figure 6.7. Since each decision involves only one feature, the division of the coordinate space is parallel to each axis.

   $X_1 = (1,1)$; $X_2 = (2,1)$
   $X_3 = (1,2)$; $X_4 = (2,2)$
   $O_5 = (6,1)$; $O_6 = (7,1)$
   $O_7 = (6,2)$; $O_8 = (7,2)$
   $X_9 = (6,7)$; $X_{10} = (7,7)$

   The rule at the first node in the axis-parallel split would be $f_1 > 4$ and the rule at the node at the next level would be $f_2 > 4$ resulting in the decision tree shown in Figure 6.8.

Figure 6.7 shows 10 points whose coordinates are as follows:

2. *Oblique split* In the case of the oblique split (Figure 6.9), with the general form of the equations being $a.f_1 + b.f_2 + c > 0$, we get the following equations:

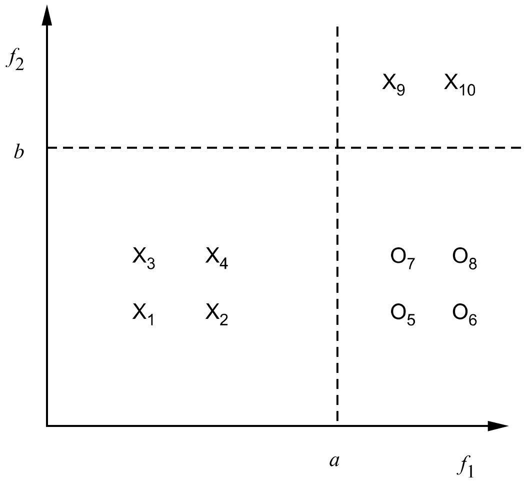


Figure 6.7 Axis-parallel split

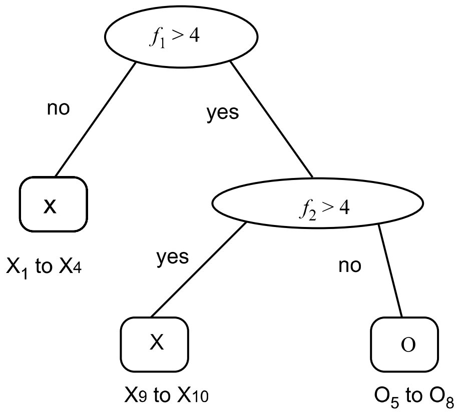


Figure 6.8 Final decision tree for the patterns shown in Figure 6.7 using axis-parallel split

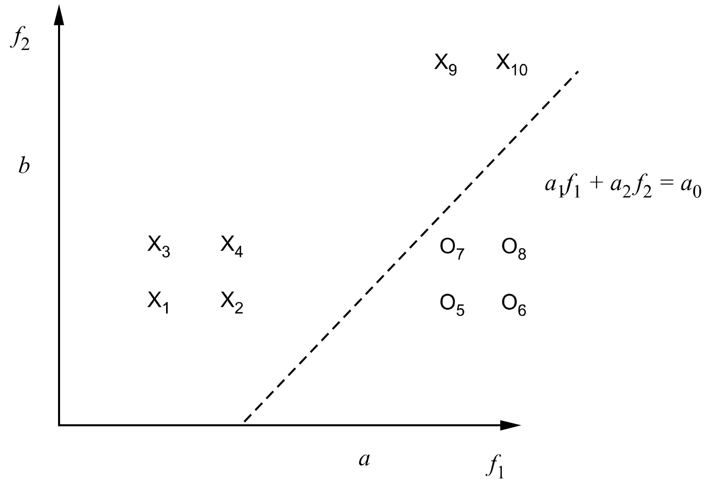
$$2a + b + c > 0$$

$$7a + 7b + c > 0$$

$$6a + 2b + c < 0$$

If we solve these inequalities, we get $a = -1$, $b = 1$ and $c = 2$.

The rule in oblique split therefore becomes $-f_1 + f_2 + 2 > 0$ or $f_1 - f_2 < 2$, giving the decision tree shown in Figure 6.10.



f_2
b
a
f_1
X_1 X_2 X_3 X_4 X_9 X_{10}
O_5 O_6 O_7 O_8
$a_1 f_1 + a_2 f_2 = a_0$

Figure 6.9 A two-class example to illustrate oblique split

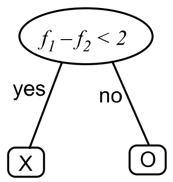


Figure 6.10 Decision tree for the patterns shown in Figure 6.9 using oblique split

3. *Multivariate split* Multivariate means the use of more than one attribute for the split at a node as opposed to univariate splitting. While a linear combination of features gives us the oblique split, it is also possible to have a non-linear combination of features to affect the split at a node.

6.4.1 When to Stop Splitting

1. *Use cross-validation*
   A part of the training data is kept aside for validation. This can also be done repeatedly keeping different sub-sets for validation and taking the average performance. The decision tree which gives the best results during validation is the tree to be considered.

2. *Reduction in impurity*
   Splitting can be stopped, when the reduction in impurity is very small. This means

   $$\text{max}_s i(s) \leq \beta,$$

   where β is a small value.

3. *Depending on the global criterion function*
   If the criterion function is of the form

   $$\alpha \times \text{size} + \sum\nolimits_{\text{leafnodes}} i(N)$$

   where α is a positive constant and size = number of nodes or links, the strategy is to stop splitting when this global criterion reaches some minimum value.

6.5 Overfitting and Pruning

Whenever there is a large set of possible hypotheses, the tree can keep growing—thus becoming too specific. An example of this is having one unique path through the tree for every pattern in the training set. This is called overfitting. The solution to this problem of overfitting is to use the technique of decision tree pruning. Pruning prevents recursive splitting using attributes that are not clearly relevant.

6.5.1 Pruning by Finding Irrelevant Attributes

In this method, the decision tree is first constructed fully and is then pruned by finding irrelevant attributes. When we split a set of examples using an attribute, the resulting sub-sets may be roughly the same proportion of each class as the original set. This results in zero information gain which is a clue to the irrelevance of the attribute. Looking at the data and the number of positive and negative examples, we can calculate the extent to which it deviates from a perfect absence of pattern. If the degree of deviation is statistically unlikely, then this is considered good evidence for the presence of a significant pattern in the data. The deviation in terms of number of positive and negative examples p_i and n_i and the expected numbers $\tilde{p}_i$ and $\tilde{n}_i$ will be

$$D = \sum_{i=1}^{v} \frac{(p_i - \tilde{p}_i)^2}{\tilde{p}_i} + \frac{(n_i - \tilde{n}_i)^2}{\tilde{n}_i}$$

where v is the sample size. D is distributed according to χ^2 (chi-squared) distribution. The probability that the attribute is really irrelevant can be calculated with the help of the standard χ^2 table. This is called χ^2 pruning.

6.5.2 Use of Cross-Validation

Overfitting can be prevented by cross-validation explained in Section 6.4.1. This is the process of estimating how well the current hypothesis will predict unseen data. Using cross-validation, it is possible to select a tree with a good prediction performance.

6.6 Example of Decision Tree Induction

A simple problem on pattern classification is considered and the decision tree is induced.

EXAMPLE 12

Table 6.5 Example training data set for induction of a decision tree

Cook	Mood	Cuisine	Tasty
Sita	Bad	Indian	Yes
Sita	Good	Continental	Yes
Asha	Bad	Indian	No
Asha	Good	Indian	Yes
Usha	Bad	Indian	Yes
Usha	Bad	Continental	No
Asha	Bad	Continental	No
Asha	Good	Continental	Yes
Usha	Good	Indian	Yes
Usha	Good	Continental	No
Sita	Good	Indian	Yes
Sita	Bad	Continental	Yes

Table 6.5 gives a set of training data. In this data, there are two classes, Tasty = yes and Tasty = no. Eight examples correspond to Tasty = yes and 4 examples to Tasty = no. The information of the data set is therefore

$$i(N) = -\frac{4}{12}\log\frac{4}{12} - \frac{8}{12}\log\frac{8}{12} = 1.8089$$

Let us consider the three attributes and find the attribute with the highest gain.

1. *Cook*

   (a) Cook = Sita has four examples belonging to Tasty = yes and 0 examples belonging to Tasty = no. This branch has an entropy of 0.

(b) Cook = Asha has 2 examples belonging to tasty = yes and 2 examples belonging to tasty = no. The entropy for Cook = Asha is

$$i(N_A) = -\tfrac{2}{4}\log\tfrac{2}{4} - \tfrac{2}{4}\log\tfrac{2}{4} = 1.0$$

(c) Cook = Usha has 2 examples belonging to Tasty = yes and 2 examples belonging to Tasty = no. The entropy for Cook = Usha is therefore

$$i(N_U) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1.0$$

The gain is therefore

$$\text{Gain(Cook)} = 1.8089 - \frac{4}{12}\times 1.0 - \frac{4}{12}\times 1.0 = 1.4223$$

2. *Mood*

(a) Mood = bad has three examples belonging to Tasty = yes and 3 examples belonging to Tasty = no. The entropy for Mood = bad is therefore

$$i(N_B) = -\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6} = 1.0$$

(b) Mood = good has 5 examples belonging to Tasty = yes and 1 example belonging to Tasty = no. The entropy for Mood = good is therefore

$$i(N_G) = -\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6} = 2.4588$$

The gain is therefore

$$\text{Gain(Mood)} = 1.8089 - \frac{6}{12}\times 2.4588 - \frac{6}{12}\times 1.0 = 0.0795$$

3. *Cuisine*

(a) Cuisine = Indian has 5 examples belonging to Tasty = yes and 1 example belonging to Tasty = no. The entropy for Cuisine = Indian is therefore

$$i(N_I) = -\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6} = 2.4588$$

(b) Cuisine = Continental has 3 examples belonging to Tasty = yes and 3 examples belonging to Tasty = no. The entropy for Cuisine = Continental is therefore

$$i(N_C) = -\frac{3}{6}\log\frac{3}{6} - \frac{3}{6}\log\frac{3}{6} = 1.0$$

The gain is therefore

$$\text{Gain(Cuisine)} = 1.8079 - \frac{6}{12} \times 2.4588 - \frac{6}{12} \times 1.0 = 0.0795$$

The attribute with the largest gain is Cook and therefore Cook is chosen as the first attribute in the decision tree.

1. *Cook = Sita*

   Cook = Sita has 4 examples belonging to Tasty = yes and 0 examples for Tasty = no, giving an entropy of 0. This branch has reached the leaf node.

2. *Cook = Asha*

   Cook = Asha has 2 examples belonging to Tasty = yes and 2 examples for Tasty = no. The entropy for Cook = Asha is therefore

$$I(N) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1.0$$

   (a) Mood
   Mood = bad has 2 examples belonging to tasty = no and 0 examples belonging to Tasty = yes, giving an entropy of 0. Mood = good has 2 examples belonging to Tasty = yes and 0 examples belonging to Tasty = no, giving an entropy of 0. The gain for Mood will therefore be 1.0.

   (b) Cuisine
   Cuisine = Indian has 1 examples belonging to Tasty = yes and 1 examples belonging to Tasty = no. The entropy for Cuisine = Indian will therefore be

$$I(N) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1.0$$

   Cuisine = Continental has 1 example belonging to Tasty = yes and 1 example belonging to Tasty = no. The entropy for Cuisine = Continental will therefore be

$$I(N) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1.0$$

   The gain for Cuisine is

$$\text{Gain(Cuisine)} = 1.0 - \frac{2}{4} \times 1.0 - \frac{2}{4} \times 1.0 = 0$$

Since Mood has a higher gain, it is chosen as the next attribute after Cook = Asha.

3. *Cook = Usha*

   If a similar process is followed for Cook = Usha, the attribute with the highest gain is Cuisine and therefore Cuisine is the next attribute chosen after Cook = Usha. (Refer to Exercise Problem No. 8.)

The final decision tree is as shown in Figure 6.11. It can be seen from the figure that if Cook = Sita, we do not require information about mood or cuisine to get the right classification. If Cook = Asha, information about the type of cuisine is redundant and for Cook = Usha, information about the mood is redundant. The leaf nodes represent the class labels.

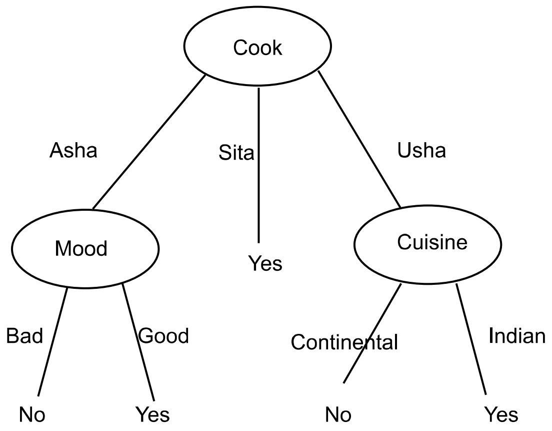


Figure 6.11 Decision tree induced from training examples

Discussion

The decision tree is an effective tool for choosing between several courses of action. It corresponds to simple rules which represent each path in the decision tree. It is induced from examples by finding which attribute gives the highest drop in impurity or highest increase in gain at each decision point.

Further Reading

Quinlan (1986) explains the basic principle of the decision tree with an example. He further describes a software for decision trees called C4.5 (1992; 1996). The various types of splitting rules for decision trees is discussed by Buntine and Niblett (1992). Murthy et al. (1994), in their paper, describe the obligue split in decision trees. The OC1 decision tree software system is also developed and described in this paper. Using a search technique, John (1994) finds multivariate splits in decision trees. Sethi and Yoo (1994) talk of a multivariate split decision tree. Wang et al. (2008) discuss fuzzy decision trees. Important papers have been written by Chandra and Varghese (2009), Kalkanis (1993), Yildiz and Dikmen (2007), Chen, Wu and Tang (2009), Ouyang et al. (2009), Chen, Hu and Tang (2009), Sieling (2008) and Twala et al. (2008)

Exercises

1. Construct a decision tree when one of three coins a, b, c is counterfeit—two are of equal weight and the third is either heavier or lighter.
2. Give an example of a decision tree where the following types of splitting criteria are used.
   (a) Axis-parallel split
   (b) Oblique split
   (c) Multivariate split
3. When a decision tree is induced from the training set, some features may not be used. Explain and give an example.
4. Induce a decision tree for the problem of choosing which restaurant to visit. The features to be used are
   (a) Expense: Expensive, reasonable, cheap
   (b) Location from home: Far, near, very close
   (c) Quality of food: Very good, fair, bad
   (d) Weather: Raining, sunny, overcast
5. Consider an axis-parallel decision tree built using n training patterns. Assume that there are C classes. What is the maximum depth of a binary decision tree corresponding to this data? What is the minimum depth of the binary decision tree?
6. Consider the following two-dimensional patterns:

Class 1	Class 2
(1, 1)	(6, 1)
(1, 2)	(6, 2)
(2, 1)	(1, 8)
	(2, 7)
	(2, 8)

(a) Obtain the best axis-parallel tree using entropy gain.

(b) Is an oblique tree simpler than the axis-parallel tree?

7. Consider a decision tree where there are a total of 25 examples. In the first node, these examples are split so that 10 examples pertain to the decision variable of the left branch and 15 to the right branch. Further, the left branch is split into three branches so that of 10, 0 pertains to the first branch, 4 examples to the second branch and 6 examples to the third branch. Find the gain of each of the branches.

8. Consider the decision tree induced for the examples given in Table 6.3. Complete the decision tree for Cook = Usha.

Computer Exercises

1. Write a program to design a decision tree and induce a decision tree for the example in Table 6.4.

2. Explore the different impurity measures to calculate gain. Implement them in the decision tree construction program and compare them.

3. Implement axis-parallel split and oblique split. Incorporate them in the decision tree construction program.

Bibliography

1. Buntine, W. and T. Niblett. A further comparison of splitting rules for decision-tree Induction. *Machine Learning* 8: 75–85. 1992.

2. Chandra, B. and P. Paul Varghese. Moving towards efficient decision tree construction. *Information Sciences* 179(8): 1059–1069. 2009.

3. Chen, Yen-Liang, Chia-Chi Wu and Kwei Tang. Building a cost-constrained decision tree with multiple condition attributes. *Information Sciences* 179(7): 967–079. 2009.

4. Chen, Yen-Liang, Hsiao-Wei Hu and Kwei Tang. Constructing a decision tree from data with hierarchical class labels. *Expert Systems with Applications* 36(3) Part 1: 4838–4847. 2009.
5. John, George H. Finding multivariate splits in decision trees using function optimization. *Proceedings of the AAAI*. 1994.
6. Kalkanis, G. The application of confidence interval error analysis to the design of decision tree classifiers. *Pattern Recognition Letters* 14(5): 355–361. 1993.
7. Murthy, Sreerama K., Simon Kasif and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2: 1–32. 1994.
8. Ouyang, Jie, Nilesh Patel and Ishwar Sethi. Induction of multiclass multifeature split decision trees from distributed data. *Pattern Recognition*. 2009.
9. Quinlan, J. R. Induction of decision trees. *Machine Learning* 1: 81–106. 1986.
10. Quinlan, J.R. *C4.5-Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann. 1992.
11. Quinlan, J.R. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research* 4: 77–90. 1996.
12. Sethi, Ishwar K. and Jae H. Yoo. Design of multicategory multifeature split decision trees using perceptron learning. *Pattern Recognition* 27(7): 939–947. 1994.
13. Sieling, Detlef. Minimization of decision trees is hard to approximate. *Journal of Computer and System Sciences* 74(3): 394–403. 2008.
14. Twala, B. E. T. H., M. C. Jones and D. J. Hand. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters* 29(7): 950–956. 2008.
15. Wang, Xi-Zhao, Jun-Hai Zhai and Shu-Xia Lu. Induction of multiple fuzzy decision trees based on rough set technique. *Information Sciences* 178(16): 3188–3202. 2008.
16. Yildiz, Olcay Taner and Onur Dikmen. Parallel univariate decision trees. *Pattern Recognition Letters* 28(7): 825–832. 2007.