CSE5230 Tutorial: The ID3 Decision Tree Algorithm

MONASH UNIVERSITY

Faculty of Information Technology

CSE5230 Data Mining

Semester 2, 2004

The aim of this exercise is to learn how one famous algorithm for constructing decision trees, ID3, works. You will do this by building a decision tree by hand for a small dataset. At the end of this exercise you should understand how ID3 constructs a decision tree using the concept of *Information Gain*. You will be able to use the decision tree you create to make a decision about new data.

1 The weather data

The weather problem is a toy data set which we will use to understand how a decision tree is built. It comes from Quinlan (1986), a paper which discusses the ID3 algorithm introduced in Quinlan (1979). It is reproduced with slight modifications in Witten and Frank (1999), and concerns the conditions under which some hypothetical outdoor game may be played. The data is shown in Table 1.

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 1: The weather data (Witten and Frank; 1999, p. 9).

In this dataset, there are five categorical attributes *outlook*, *temperature*, *humidity*, *windy*, and *play*. We are interested in building a system which will enable us to decide whether or not to play the game on the basis of the weather conditions, *i.e.* we wish to predict the value of *play* using *outlook*, *temperature*,

humidity, and *windy*. We can think of the attribute we wish to predict, *i.e. play*, as the **output** attribute, and the other attributes as **input** attributes.

2 Building a decision tree using the ID3 algorithm

As we saw in lecture 6, a decision tree consists of nodes and arcs which connect nodes. To make a decision, one starts at the root node, and asks questions to determine which arc to follow, until one reaches a leaf node and the decision is made. This basic structure is shown in Figure 1.

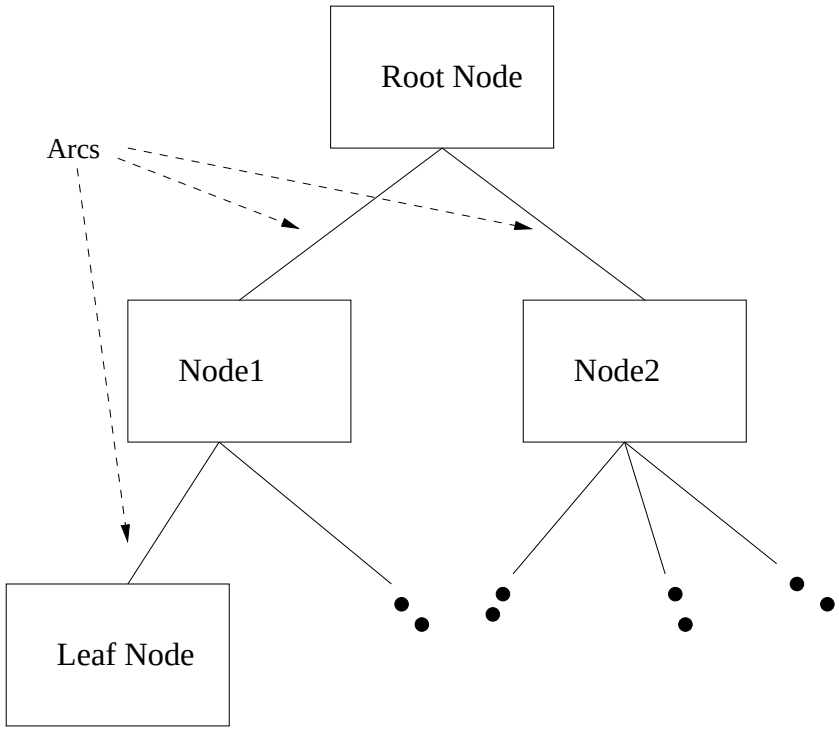

Figure 1: Basic decision tree structure

The main ideas behind the ID3 algorithm are:

- Each non-leaf node of a decision tree corresponds to an input attribute, and each arc to a possible value of that attribute. A leaf node corresponds to the expected value of the output attribute when the input attributes are described by the path from the root node to that leaf node.
- In a "good" decision tree, each non-leaf node should correspond to the input attribute which is the *most informative* about the output attribute amongst all the input attributes not yet considered in the path from the root node to that node. This is because we would like to predict the output attribute using the smallest possible number of questions on average.

- *Entropy* is used to determine how informative a particular input attribute is about the output attribute for a subset of the training data. Entropy is a measure of uncertainty in communication systems introduced by Shannon (1948). It is fundamental in modern information theory.

2.1 Entropy

In information theory, entropy is a measure of the uncertainty about a source of messages. The more uncertain a receiver is about a source of messages, the more information that receiver will need in order to know what message has been sent.
For example, if a message source always sends exactly the same message, the receiver does not need any information to know what message has been sent—it's always the same! The entropy of such a source is zero: there is no uncertainty at all. On the other hand, if a source can send n possible messages and each message occurs independently of the preceding message with equal probability, then the uncertainty of the receiver is maximised. The receiver will need to ask $\log_2 n$ yes/no questions to determine which message has been sent, *i.e.* the receiver needs $\log_2 n$ *bits* of information.

Question 1 Why does the receiver need $\log_2 n$ bits? Think about representing each message as a binary number.

The average number of bits required to identify each message is a measure of the receiver's uncertainty about the source, and is known as the entropy of the source.
Consider a source S which can produce n messages $\{m_1, m_2, \ldots, m_n\}$. All messages are produced independently of each other, and the probability of producing message m_i is p_i. For such a source S with a message probability distribution $P = (p_1, p_2, \ldots, p_n)$, the entropy $H(P)$ is

$$H(P) = -\sum_{i=1}^{n} p_i \log(p_i). \tag{1}$$

In the equation above, $\log(p_i)$ means $\log_2(p_i)$—from now on we will assume that all logarithms are to the base two.
If a set T of records from a database (*i.e.* the training set for building the decision tree) is partitioned into k classes $\{C_1, C_2, \ldots, C_k\}$ on the basis of the output attribute, then the average amount of information (measured in bits) needed to identify the class of a record is $H(P_T)$, where P_T is the probability distribution of the classes, estimated from the data as

$$P_T = \left(\frac{|C_1|}{|T|}, \frac{|C_2|}{|T|}, \ldots, \frac{|C_k|}{|T|} \right). \tag{2}$$

The notation $|C_i|$ means the number of elements in set C_i. For the weather data, where *play* is the output attribute, we have for the entire dataset T

$$P_T = \left(\frac{5}{14}, \frac{9}{14} \right),$$

where class C_1 corresponds to "no", and class C_2 to "yes". Using Equation 1, the entropy of set T is thus

$$\begin{aligned} H(T) = H(P_T) &= -\left(\frac{5}{14} \log(\frac{5}{14}) + \frac{9}{14} \log(\frac{9}{14}) \right) \\ &= 0.940. \end{aligned}$$

Note that here we identify the entropy of the set $H(T)$ with the entropy of the probability distribution of the members of the set, $H(P_T)$.

2.2 Information gain

Now consider what happens if we partition the set on the basis of an input attribute X into subsets $T_1, T_2, \ldots, T_n$. The information needed to identify the class of an element of T is the weighted average of the information needed to identify the class of an element of each subset:

$$H(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} H(T_i). \tag{3}$$

In the context of building a decision tree, we are interested in how much information about the output attribute can be gained by knowing the value of an input attribute X. This is just the difference between the information needed to classify an element of T before knowing the value of X, $H(T)$, and the information needed after partitioning the dataset T on the basis of knowing the value of X, $H(X,T)$. We define the *information gain* due to attribute X for set T as[1]

$$\mathrm{Gain}(X,T) = H(T) - H(X,T). \tag{4}$$

In order to decide which attribute to split upon, the ID3 algorithm computes the information gain for each attribute, and selects the one with the highest gain.
As an example, let us consider the information gain for attribute *temperature* in the weather data. The attribute *temperature* can have three values, *cool*, *mild* and *hot*. Partitioning on the basis of temperature will thus divide T into three subsets. This reorganisation of the weather data is shown in Table 2. We have $|T_{cool}| = 4$, $|T_{mild}| = 6$ and $|T_{hot}| = 4$. We can calculate the information needed to classify an element of T after this partitioning using Equation 3:

$$\begin{aligned}
H(temperature,T) &= \frac{4}{14}H(T_{cool}) + \frac{6}{14}H(T_{mild}) + \frac{4}{14}H(T_{hot}) \\
&= \frac{4}{14}\left(-\left(\frac{1}{4}\log(\frac{1}{4}) + \frac{3}{4}\log(\frac{3}{4})\right)\right) + \frac{6}{14}\left(-\left(\frac{2}{6}\log(\frac{2}{6}) + \frac{4}{6}\log(\frac{4}{6})\right)\right) + \\
&\quad \frac{4}{14}\left(-\left(\frac{2}{4}\log(\frac{2}{4}) + \frac{2}{4}\log(\frac{2}{4})\right)\right) \\
&= \frac{4}{14} \times 0.811 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 1.00 \\
&= 0.911
\end{aligned}$$

The information gain for attribute *temperature* for set T is thus

$$\begin{aligned}
\mathrm{Gain}(temperature,T) &= 0.940 - 0.911 \\
&= 0.029 \text{ bits.}
\end{aligned}$$

[1] Note that in information theory $\mathrm{Gain}(X,T)$ is known as the *mutual information* between the input attribute X and the output attribute.

outlook	temperature	humidity	windy	play
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	cool	normal	false	yes
rainy	mild	high	false	yes
sunny	mild	high	false	no
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
rainy	mild	high	true	no
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
overcast	hot	normal	false	yes

Table 2: The weather data partitioned on the basis of attribute *temperature*.

Question 2 Calculate the information gain for the other input attributes, *outlook*, *humidity*, and *windy*. You may use whatever tools you like to do the calculation. Note the following useful identity:

$$\log_a(x) = \frac{\log_b(x)}{\log_b(a)}.$$

An *entropy calculator* has been created to help you with this, and subsequent, questions. You can find it at:
`http://www.csse.monash.edu.au/~davids/cgi-bin/cse5230/entropy_calc.cgi`.

Question 3 Given your answer to the question above, what attribute would ID3 select to split upon for set T, *i.e.* the complete weather data from Table 1?

2.3 The ID3 algorithm

The ID3 algorithm works by recursively applying the procedure above to each of the subsets produced until "pure" nodes are found—a pure node contains elements of only one class—or until there are no attributes left to consider. It can be stated in pseudocode, as is shown in Figure 2.

Question 4 Use the ID3 algorithm given in Figure 2 to construct a decision tree for the weather data by hand. The first call to `ID3()` uses the entire set of input attributes and the entire set of training data.

Question 5 Consider a new data element with input attributes $\{overcast, cool, high, true\}$. How is it classified by the tree you constructed above?

```
function ID3 (I, 0, T) {
/*  I is the set of input attributes
*   O is the output attribute
*   T is a set of training data
*
*   function ID3 returns a decision tree
*/
    if (T is empty) {
        return a single node with the value "Failure";
    }
    if (all records in T have the same value for O) {
        return a single node with that value;
    }
    if (I is empty) {
        return a single node with the value of the most frequent value of
        O in T;
        /* Note: some elements in this node will be incorrectly classified */
    }

    /* now handle the case where we can't return a single node */
    compute the information gain for each attribute in I relative to T;
    let X be the attribute with largest Gain(X, T) of the attributes in I;
    let {x_j| j=1,2, .., m} be the values of X;
    let {T_j| j=1,2, .., m} be the subsets of T when T is partitioned
        according the value of X;
        return a tree with the root node labelled X and
        arcs labelled x_1, x_2, .., x_m, where the arcs go to the
        trees ID3(I-{X}, O, T_1), ID3(I-{X}, O, T_2), .., ID3(I-{X}, O, T_m);
}
```

Figure 2: The ID3 algorithm

2.4 The problem of attributes with many values

The simple ID3 algorithm above can have difficulties when an input attribute has many possible values, because $\mathrm{Gain}(X, T)$ tends to favour attributes which have a large number of values. It is easy to understand why if we consider an extreme case.

Imagine that our dataset T contains an attribute that has a different value for every element of T. This could arise in practice if a unique record ID was retained when extracting T from a database—for example a patient ID number in a hospital database. Such an attribute would give the maximum possible information gain, since all the training data can be correctly classified by examining its value. It would result in a decision tree in which all nodes below the root node were leaf nodes. This tree, however, would be completely useless for classifying new data: there would be no arc corresponding to the value

of the ID attribute. Moreover, there is no reason to suspect that such an attribute would have any causal relationship with the output attribute.
The problem also arises when an attribute can take on many values, even if they are not unique to each element. Quinlan (1986) suggests a solution based on considering the amount of information required to determine the value of an attribute X for a set T. This is given by $H(P_{X,T})$, where $P_{X,T}$ is the probability distribution of the values of X:

$$P_{X,T} = \left(\frac{|T_1|}{|T|}, \frac{|T_2|}{|T|}, \ldots, \frac{|T_n|}{|T|} \right). \tag{5}$$

The quantity $H(P_{X,T})$ is known as the *split information* for attribute X and set T. We will call it $\mathrm{SplitInfo}(X,T) = H(P_{X,T})$. For the weather data, the split information for the attribute *temperature* is

$$\begin{aligned} \mathrm{SplitInfo}(temperature, T) &= -\left(\frac{4}{14}\log(\frac{4}{14}) + \frac{6}{14}\log(\frac{6}{14}) + \frac{4}{14}\log(\frac{4}{14}) \right) \\ &= 1.56 \text{ bits.} \end{aligned}$$

Quinlan (1986) suggests that rather than choosing the attribute with the biggest $\mathrm{Gain}(X,T)$, we select the one with the biggest $\mathrm{GainRatio}(X,T)$, where

$$\mathrm{GainRatio}(X,T) = \frac{\mathrm{Gain}(X,T)}{\mathrm{SplitInfo}(X,T)}. \tag{6}$$

Note that $\mathrm{GainRatio}(X,T)$ might not always be defined. Quinlan (1986) specifies a *gain ratio criterion*, which says that we should select the attribute X with the highest $\mathrm{GainRatio}(X,T)$ from amongst those attributes with average-or-better $\mathrm{Gain}(X,T)$.
If you have time, attempt the following questions.

Question 6 Under what circumstances is $\mathrm{SplitInfo}(X,T)$ equal to 0?

Question 7 What is $\mathrm{GainRatio}(temperature, T)$?

Question 8 What are the gain ratios for the other input attributes?

Question 9 Construct a decision tree for the weather data using the gain ratio criterion rather than the information gain.

References

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples, *in* D. Michie (ed.), *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, pp. 168–201.

Quinlan, J. R. (1986). Induction of decision trees, *Machine Learning* **1**(1): 81–106.

Shannon, C. E. (1948). A mathematical theory of communication, *Bell System Technical Journal* **27**: 379–423 and 623–656.
`http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html`

Witten, I. H. and Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Francisco, CA, USA.
`http://www.cs.waikato.ac.nz/~ml/weka/book.html`