Decision Trees

1 Introduction

Decisions trees perform supervised learning. These are essentially classifiers that are trained on a training set with the aim of generalizing on unseen data. One of the advantages of decision trees is that they are interpretable. [1]. An example of a decision tree from [1] is illustrated in Figure 1 below.

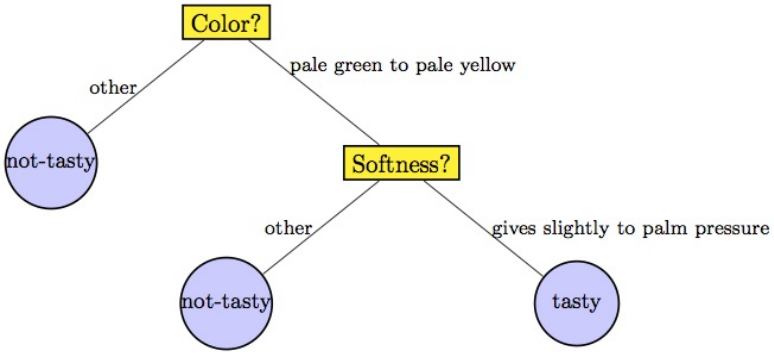

Figure 1: Example decision tree [1]

If trees are allowed to be of infinite size this could lead to overfitting. Hence, a limit is set on the size of the decision tree. Decision trees are created using induction algorithms. Various algorithms exist for generating decision trees, one of the most popular being ID3 which we will examine. Before we look at the ID3 algorithm we will firstly discuss the concept of *gain* which is used in decision tree induction algorithms.

2 Dominance Measurements

In creating a decision tree induction algorithms use certain measures to determine which is the dominant attribute or feature at each point in constructing the tree. This section discusses some of the measures that are used for this. We will use the data set in Table 1 to illustrate the different measurements. The data instances listed in the table are for making the decision to play netball (*yes*) or not to play netball (*no*). The attributes/features are:

1. Outlook - This describes the weather for the day as being either sunny, overcast or rainy.
2. Temperature - The temperature can be hot, normal or cool.
3. Humidity - Can be high or normal.
4. Wind - The wind can be weak or strong

Table 1: Example Dataset

Day	Outlook	Temperature	Humidity	Wind	Play Netball?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Two measures that are used by decision tree induction algorithms are entropy and *gain*.We will refer to the determining what class an entity belongs to as the decision represented by D. The entropy of D, $E(D)$, is calculated as follows:

$$E(D) = \sum_{i=1}^{n} -p(i)log_2 p(i) \tag{1}$$

where i is the ith class, n is the number of classes, $p(i)$ is the probability of class i being in an instance. $p(i)$ is calculated as follows:

$$p(i) = \frac{Number\ of\ instances\ i\ is\ the\ class\ for}{Total\ number\ of\ instances} \tag{2}$$

Given the dataset in Table 1 $p(yes) = \frac{9}{14}$, $p(no) = \frac{5}{14}$ and the entropy is:

$$E(D) = -(\frac{9}{14})log_2(\frac{9}{14}) - (\frac{5}{14})log_2(\frac{5}{14}) = 0.94 \tag{3}$$

Induction algorithms use the *gain* to determine which of the attributes is the dominant attribute. The following equation is used to calculate the gain G for an attribute A:

$$G(D, A) = E(D) - \sum_{j=1}^{m} p(D|A = j)E(D|A = j) \tag{4}$$

where j represents the jth value of the attribute and m is the number of values the attribute can take. $p(D|A = j)$ is:

$$p(D|A = j) = \frac{Number\ of\ instances\ containing\ j}{Total\ number\ of\ instances} \tag{5}$$

Suppose that we want to calculate the gain for the attribute *wind* in Table 1:

$$\begin{aligned} G(D, wind) = E(D) - (&p(D|wind = weak)E(D|wind = weak) + \\ &p(D|wind = strong)E(D|wind = strong)) \end{aligned} \tag{6}$$

$$p(D|wind = weak) = \frac{8}{14} = 0.57 \tag{7}$$

$$E(D|wind = weak) = -p(no)_{wind=weak} log_2 p(no)_{wind=weak} - p(yes)_{wind=weak} log_2 p(yes)_{wind=weak} \tag{8}$$

$$p(D|wind = strong) = \frac{6}{14} = 0.43 \tag{9}$$

$$\begin{aligned} E(D|wind = strong) = &-p(no)_{wind=strong} log_2 p(no)_{wind=strong} - \\ &p(yes)_{wind=strong} log_2 p(yes)_{wind=strong} \\ &= -\frac{3}{6} log_2(\frac{3}{6}) - \frac{3}{6} log_2(\frac{3}{6}) = 1 \end{aligned} \tag{10}$$

$$\begin{aligned} E(D|wind = weak) = &-p(no)_{wind=weak} log_2 p(no)_{wind=weak} - \\ &p(yes)_{wind=weak} log_2 p(yes)_{wind=weak} \\ &= -\frac{2}{8} log_2(\frac{2}{8}) - \frac{6}{8} log_2(\frac{6}{8}) = 0.81 \end{aligned} \tag{11}$$

Hence,

$$G(D, wind) = 0.94 - (0.57)(0.81) - (0.43)(1) = 0.048 \tag{12}$$

In the following section we examine ID3 induction algorithm which uses both entropy and gain to generate a decision tree.

Algorithm 1 ID3 Algorithm

```
1: procedure ID3(S, A)
2:     if all labels in S are 1 then
3:         return a leaf node of 1
4:     end if
5:     if all labels in S are 0 then
6:         return a leaf node of 0
7:     end if
8:     if A is empty then
9:         return a node with the most frequently occurring label in S
10:    else
11:        maxAttr = be the attribute with the maximum gain (equation 4)
12:        if all the instances in S have the same label then
13:            return a leaf node with the majority label in S
14:        else
15:            Make the root of the decision tree maxAttr
16:            A' = A/maxAttr
17:            for v ← 1, number of values for j do
18:                S'= instances in S containing value c_v of A
19:                if S' is empty then
20:                    Add a leaf node with the most frequently occurring label S
21:                else
22:                    Add ID3(S',A') as a subtree
23:                end if
24:            end for
25:        end if
26:    end if
27: end procedure
```

3 The ID3 Algorithm

The ID3 (Iterative Dichotomizer 3) algorithm iterative divides the attributes or features of the given dataset into groups of two at each stage to produce a decision tree. The algorithm is depicted in Algorithm 1.

The algorithm firstly checks whether all the labels in the dataset are of a particular class, if which case the decision tree is a leaf node of this class. It then finds that attribute with the highest gain $maxAttr$ and this becomes the root of the decision tree. The next level of the decision tree is connecting a child node c_v to $maxAttr$ for each value of $maxAttr$. If the value c_v does not occur in the dataset then the child of c_v is the most frequently occurring label in S. If this is not the case the subtree rooted at c_v is created by calling up $ID3(S', A')$ with S' containing those instances of S that contain value c_v and A' contain the remaining attributes excluding $maxAttr$. Applying ID3 to the dataset in Table 1 produces the decision tree in Figure 2.

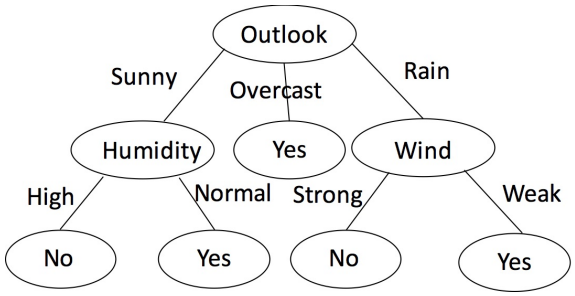

Figure 2: Example decision tree [1]

One of the shortcomings of the ID3 algorithm is that the tree that is produced by the algorithm can be large. To reduce the size of the tree either the number of iterations of ID3 can be limited or alternatively the resulting tree can be *pruned* [1].

4 Random Forests

A random forest is an *ensemble* of decision trees [1] that is used to perform classification. Each decision tree in the ensemble can be created using different induction algorithms. The process followed to perform classification is as follows:

- Choose a subset s_i from the training set T.
- Create each classifier c_i using subset s_i .
- *Majority voting* is used to classify an instance. The class that a majority of the classifiers have classified the instance as is class output by the ensemble.

One the advantages of random forests is that it overcomes overfitting.

5 Online Resources

The following free books on machine learning can be accessed online and used to supplement these notes:

Shalev-Shwartz, S. and Ben-David, S., Understanding Machine Learning. Cambridge University Press, 2014.

References

[1] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning.* Cambridge University Press, 2014.

Unsupervised Learning and K-Means

Introduction

In this section we are going to look at unsupervised learning. The main difference between unsupervised learning and supervised learning is that in unsupervised learning the dataset does not contain labels. In supervised learning we perform *classification*, in unsupervised learning *clustering* is performed instead. Clustering involves dividing the data into clusters or groups, with the most similar data instances grouped together, and the clusters being dissimilar from each other. Some measure, e.g. the Euclidean distance, is employed to measure the similarity. The measure used is dependent on the clustering algorithm used. Different clustering algorithms exist. The *K-Means* is the most commonly used algorithm which is presented in the next section.

K-Means Algorithm

The K-Means algorithm clusters a given dataset into K clusters. The standard K-Means algorithm is used for numerical data. However, there are versions of the algorithm that have been applied to mixed data. K in K-Means represents the number of clusters. The value can be selected randomly or determined using an empirical technique such as the *elbow* technique. The K-Means algorithm is depicted in Algorithm 1.

Algorithm 1 K-Means Algorithm

Given a set of data instances $D = d_1, ..., d_n$
Determine K (the number of clusters)
Randomly select centroids c_1 to c_k for each of the K clusters
while the algorithm has not converged **do**
    for $i \leftarrow 1$ *to* n **do**
        for $j \leftarrow 1$ *to* K **do**
            Calculate the Euclidean distance e_j of d_i from the centroid of cluster k_j
        end for
        Add d_i to the cluster k_j with the smallest e_j value
    end for
    for $j \leftarrow 1$ *to* K **do**
        for $l \leftarrow 1$ *to* m **do**
            Calculate the average a_l of the lth dimension of the data instances in
            the cluster j
        end for
        Update the jth centroid to the averaged values a_j for each dimension of
        the data instance
    end for
end while

The algorithm begins by selecting a value for K. K centroids are then randomly selected. The purpose of the *while* loop in line 4 is to form the clusters and move data instances between clusters when the centroids of the clusters are updated. The reason that a data instance d_i may move is when the centroid values are updated it may be closer to the centroid of another cluster than the one it is currently in. If there is no movement of data instances between clusters for an iteration of the *while* loop, then the algorithm has converged and the *while* loop is terminated. The *for* loop in line 5 is used to assign data instances d_i to a cluster based on the Euclidean distance, defined in equation (1) below, e_j between the data instance and the centroid of the cluster c_j.

$$e(x, y) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2} \tag{1}$$

Once the clusters are formed on each iteration the centroids are updated. This is depicted in lines 11 to 16 of the algorithm. The centroids are updated by taking an average of each dimension/attribute value in the data instance d_i across all data instances in the particular cluster. The Euclidean distance of each data instance d_i from the updated centroids of each cluster c_j is then calculated and if the data instance d_i has a lower Euclidean distance to a centroid of a cluster it is not in, it is moved to that cluster. This process of updating the centroids and determining which cluster centroid each data instance d_i is the closest to continues until there is no movement of data instances from one cluster to the another.

We will now look at an example of applying the K-Means algorithm. Consider the 5 data instances in Table 1:

Table 1: Example Data Instances

Entity	**Attr1**	**Attr2**
1	1	1
2	1.5	2
3	3	4
4	5	7
5	3.5	5

Each data instance has a dimension of two, i.e. there are two attribute values. We will look at the first two iterations of K-Means. Suppose that the value of K is randomly chosen to be 2 and the centroids of each cluster are chosen to be entity 2 and entity 4. Table 2 lists the Euclidean distance for the remaining data instances from the centroid for Cluster 1 and Cluster 2. Entity 2 is the centroid for Cluster 1 and entity 4 for Cluster 2.

The Euclidean distance for entities 1 and 3 is smaller for Cluster 1 and for entity 5 it is smaller for Cluster 2. Hence, Cluster 1 will contain entities 1, 2 and 3 and Cluster 2

Table 2: Euclidean distance-Iteration 1

Entity	**Attr1**	**Attr2**	**Cluster 1**	**Cluster 2**
1	1	1	1.12	7.21
3	3	4	2.5	3.61
5	3.5	5	3.61	2.5

entities 4 and 5. Let us look at one more iteration of the K-Means algorithm. Table 3 and Table 4 show the euclidean distance for Cluster 1 and Cluster 2 with the updated centroids respectively. The updated centroid value for $Attr1$ is the average of this value for entity 1, 2 and 3. Similarly, for $Attr2$ and the centroid for Cluster 2 the average of the attribute values for all the instances in the cluster is taken.

Table 3: Cluster 1

Entity	**Attr1**	**Attr2**	**Cluster 1**	**Cluster 2**
Entity 1	**1**	**1**	**1.57**	5.96
Entity 2	**1.5**	**2**	**0.46**	4.85
Entity 3	**3**	**4**	**2.04**	2.35
Updated centroid	**1.83**	**2.33**		

Table 4: Cluster 2

Entity	**Attr1**	**Attr2**	**Cluster 1**	**Cluster 2**
Entity 4	5	7	5.64	**1.25**
Entity 5	3.5	5	3.15	**1.25**
Updated centroid	4.25	6		

The lower Euclidean distance is indicated in bold. The smaller Euclidean distance is to the centroid of the cluster that the data instance is already in, hence there is no movement of data instances between clusters. Thus the algorithm has converged and will terminate.