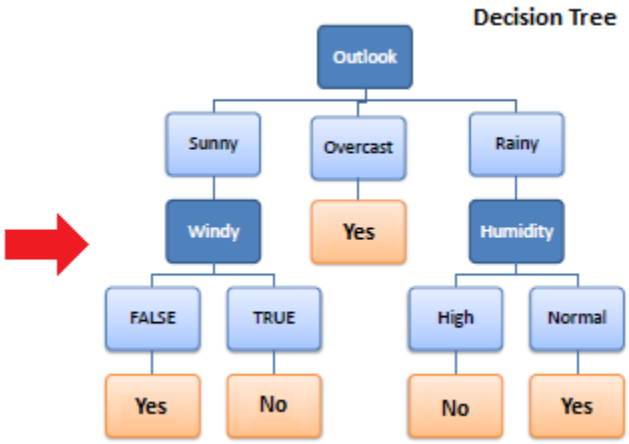
Decision Tree Classifier: Illustrative Example

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

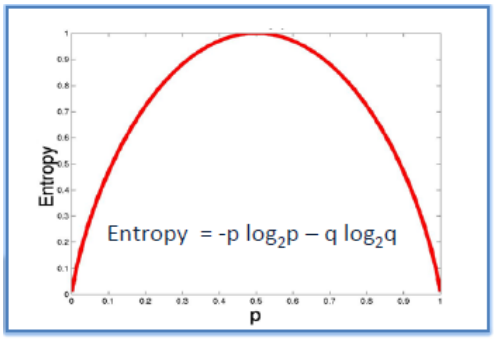


Algorithm

The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses *Entropy* and *Information Gain* to construct a decision tree.

Entropy

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous, the entropy is zero and if the sample is an equally divided it has entropy of one.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

b) Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c) E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Step 1: Calculate entropy of the target.

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

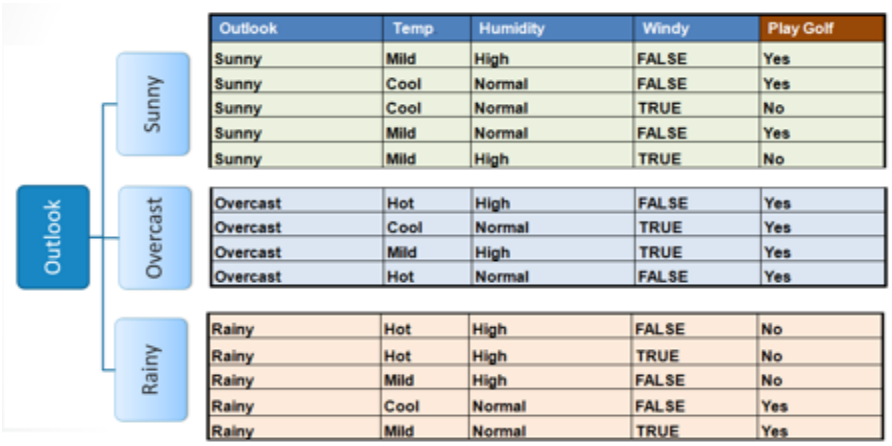
$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

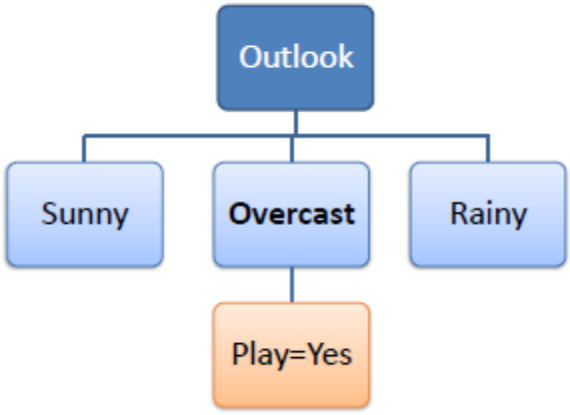
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

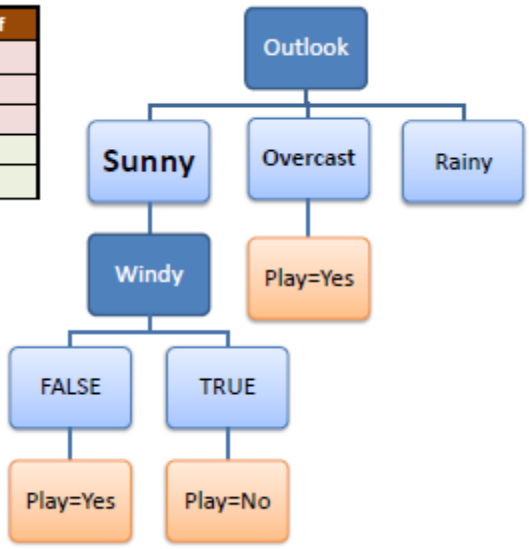
Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

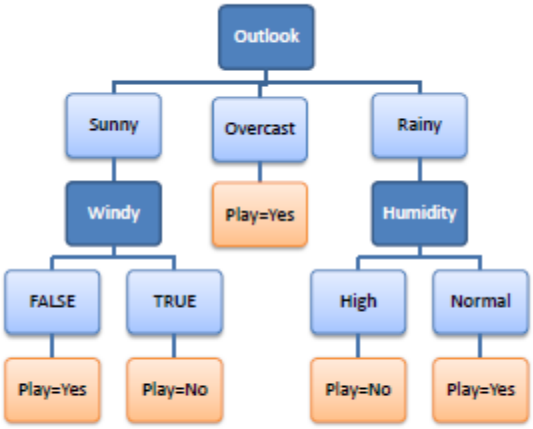
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



Example

We will stick with our weekend example. Suppose we want to train a decision tree using the following instances:

Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

The first thing we need to do is work out which attribute will be put into the node at the top of our tree: either weather, parents or money. To do this, we need to calculate:

$$\text{Entropy}(S) = -p_{cinema}\log_2(p_{cinema}) - p_{tennis}\log_2(p_{tennis}) - p_{shopping}\log_2(p_{shopping}) - p_{stay\_in}\log_2(p_{stay\_in})$$
$$= -(6/10) * \log_2(6/10) - (2/10) * \log_2(2/10) - (1/10) * \log_2(1/10) - (1/10) * \log_2(1/10)$$
$$= -(6/10) * -0.737 - (2/10) * -2.322 - (1/10) * -3.322 - (1/10) * -3.322$$
$$= 0.4422 + 0.4644 + 0.3322 + 0.3322 = 1.571$$

and we need to determine the best of:

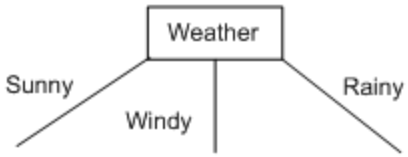
$$\text{Gain}(S, \text{weather}) = 1.571 - (|S_{sun}|/10)*\text{Entropy}(S_{sun}) - (|S_{wind}|/10)*\text{Entropy}(S_{wind}) - (|S_{rain}|/10)*\text{Entropy}(S_{rain})$$
$$= 1.571 - (0.3)*\text{Entropy}(S_{sun}) - (0.4)*\text{Entropy}(S_{wind}) - (0.3)*\text{Entropy}(S_{rain})$$
$$= 1.571 - (0.3)*(0.918) - (0.4)*(0.81125) - (0.3)*(0.918) = 0.70$$

$$\text{Gain}(S, \text{parents}) = 1.571 - (|S_{yes}|/10)*\text{Entropy}(S_{yes}) - (|S_{no}|/10)*\text{Entropy}(S_{no})$$
$$= 1.571 - (0.5) * 0 - (0.5) * 1.922 = 1.571 - 0.961 = 0.61$$

$$\text{Gain}(S, \text{money}) = 1.571 - (|S_{rich}|/10)*\text{Entropy}(S_{rich}) - (|S_{poor}|/10)*\text{Entropy}(S_{poor})$$
$$= 1.571 - (0.7) * (1.842) - (0.3) * 0 = 1.571 - 1.2894 = 0.2816$$

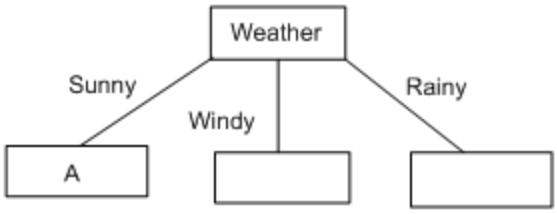
This means that the first node in the decision tree will be the weather attribute. As an exercise, convince yourself why this scored (slightly) higher than the parents attribute - remember what entropy means and look at the way information gain is calculated.

From the weather node, we draw a branch for the values that weather can take: sunny, windy and rainy:



Now we look at the first branch. S_{sunny} = {W1, W2, W10}. This is not empty, so we do not put a default categorisation leaf node here. The categorisations of W1, W2 and W10 are Cinema, Tennis and Tennis respectively. As these are not all the same, we cannot put a categorisation leaf node here. Hence we put an attribute node here, which we will leave blank for the time being.

Looking at the second branch, S_{windy} = {W3, W7, W8, W9}. Again, this is not empty, and they do not all belong to the same class, so we put an attribute node here, left blank for now. The same situation happens with the third branch, hence our amended tree looks like this:



Now we have to fill in the choice of attribute A, which we know cannot be weather, because we've already removed that from the list of attributes to use. So, we need to calculate the values for Gain(S_{sunny}, parents) and Gain(S_{sunny}, money). Firstly, Entropy(S_{sunny}) = 0.918. Next, we set S to be S_{sunny} = {W1,W2,W10} (and, for this part of the branch, we will ignore all the other examples). In effect, we are interested only in this part of the table:

Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W10	Sunny	No	Rich	Tennis

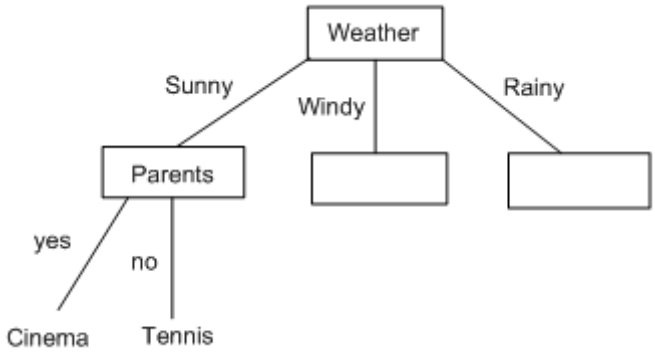
Hence we can calculate:

Gain(S_{sunny}, parents) = 0.918 - ($|S_{yes}|/|S|$)*Entropy(S_{yes}) - ($|S_{no}|/|S|$)*Entropy(S_{no})
= 0.918 - (1/3)*0 - (2/3)*0 = 0.918

Gain(S_{sunny}, money) = 0.918 - ($|S_{rich}|/|S|$)*Entropy(S_{rich}) - ($|S_{poor}|/|S|$)*Entropy(S_{poor})
= 0.918 - (3/3)*0.918 - (0/3)*0 = 0.918 - 0.918 = 0

Notice that Entropy(S_{yes}) and Entropy(S_{no}) were both zero, because S_{yes} contains examples which are all in the same category (cinema), and S_{no} similarly contains examples which are all in the same category (tennis). This should make it more obvious why we use information gain to choose attributes to put in nodes.

Given our calculations, attribute A should be taken as parents. The two values from parents are yes and no, and we will draw a branch from the node for each of these. Remembering that we replaced the set S by the set S_{Sunny}, looking at S_{yes}, we see that the only example of this is W1. Hence, the branch for yes stops at a categorisation leaf, with the category being Cinema. Also, S_{no} contains W2 and W10, but these are in the same category (Tennis). Hence the branch for no ends here at a categorisation leaf. Hence our upgraded tree looks like this:



Finishing this tree off is left as a tutorial exercise.

11.4 Avoiding Overfitting

As we discussed in the previous lecture, overfitting is a common problem in machine learning. Decision trees suffer from this, because they are trained to stop when they have perfectly classified all the training data, i.e., each branch is extended just far enough to correctly categorise the examples relevant to that branch. Many approaches to overcoming overfitting in decision trees have been attempted. As summarised by Tom Mitchell, these attempts fit into two types:

- Stop growing the tree before it reaches perfection.
- Allow the tree to fully grow, and then **post-prune** some of the branches from it.

The second approach has been found to be more successful in practice. Both approaches boil down to the question of determining the correct tree size. See Chapter 3 of Tom Mitchell's book for a more detailed description of overfitting avoidance in decision tree learning.