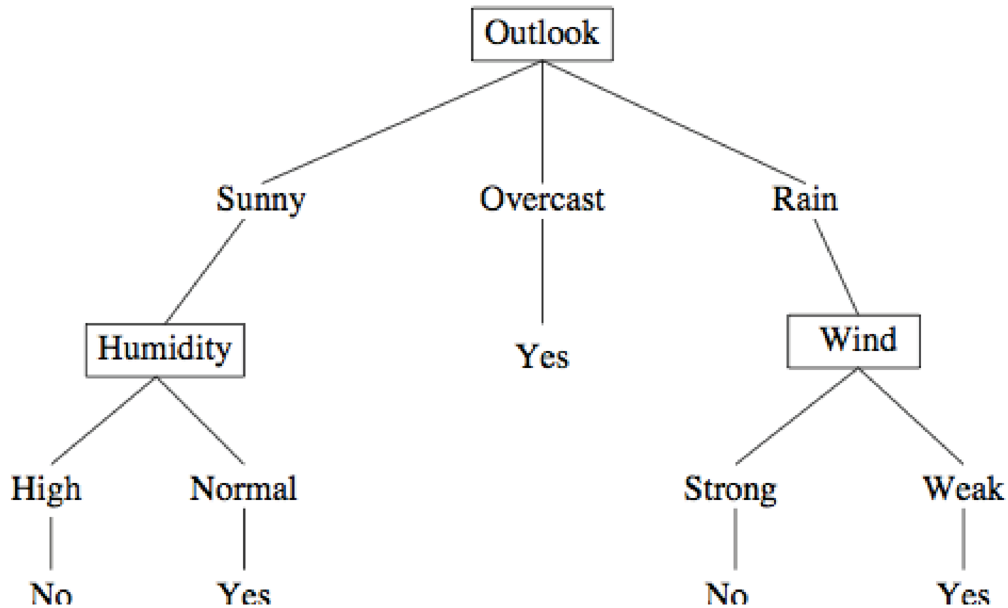
Decision Trees

Consider this binary classification data set:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

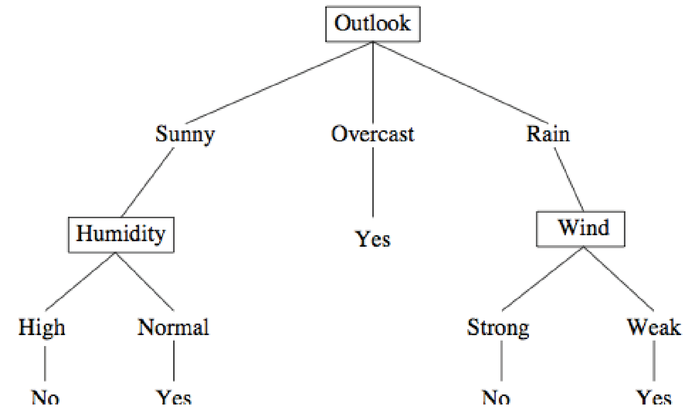
Decision Trees

We can describe this data set with the following decision tree:



Decision Trees

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



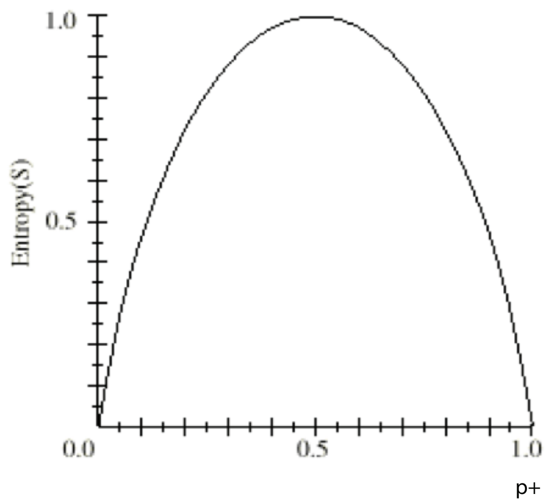
All observations in the data set are perfectly described by the tree.

Question: How do we build such trees?

Entropy

The key to decision tree induction is the notion of entropy,

$\boxed{\text{Entropy} \equiv \text{measure of randomness}}$



Observation: Entropy is at its maximum if we have a 50%-50% split among the positive and negative examples.

Observation: Entropy is zero if we have all positive or all negative examples.

Entropy

We can apply entropy to measure the “randomness" of our data set.

Let

$$D = \{(\overline{x}_1, y_1), \ldots, (\overline{x}_l, y_l)\} \subseteq A^n \times \{+1, -1\}$$

and

$$\begin{aligned} l_+ &= |\{(\overline{x}, y) \mid (\overline{x}, y) \wedge y = +1\}| \\ l_- &= |\{(\overline{x}, y) \mid (\overline{x}, y) \wedge y = -1\}| \end{aligned}$$

then

$$Entropy(D) = -\frac{l_+}{l}\log_2(\frac{l_+}{l}) - \frac{l_-}{l}\log_2(\frac{l_-}{l})$$

Now let $p_+ = l_+/l$ and $p_- = l_-/l$ then

$$\boxed{Entropy(D) = -p_+ \log_2(p_+) - p_- \log_2(p_-)}$$

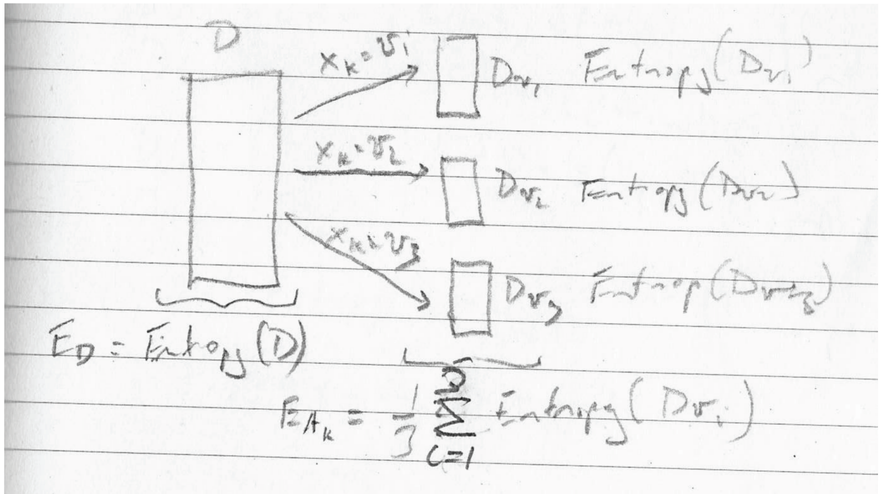
Information Gain

Def: We say that an attribute is *informative* if, when the training set is split according to its attribute values, the overall entropy in the training data is reduced.

Example: Consider the attribute $A_k = \{v_1, v_2, v_3\}$ then the split D_{v_i} of D only contains instances that have value v_i of attribute A_k,

$$D_{v_i} = \{(\overline{x}, y) \mid x_k = v_i\}$$

We can now split the data set D according to the values of attribute A_k,



If $E_{A_k} < E_D$ then attribute A_k is informative.

Information Gain

Rather than using the arithmetic mean we use the weighted mean,

$$Entropy(A_k) = \sum_{v_i \in A_k} \frac{|D_{v_i}|}{|D|} Entropy(D_{v_i})$$

Formally we define information gain as,

$$Gain(D, A_k) = Entropy(D) - Entropy(A_k)$$

or

$$\boxed{Gain(D, A_k) = Entropy(D) - \sum_{v_i \in A_k} \frac{|D_{v_i}|}{|D|} Entropy(D_{v_i})}$$

$\Rightarrow$ The larger the difference the more informative an attribute!

Information Gain

We can now use the gain to build a decision tree top-down (greedy heuristic).

Example: Consider our tennis data set with

Wind = {Weak, Strong}

Then

$$D = [9+, 5-]$$
$$D_{\text{Weak}} = [6+, 2-]$$
$$D_{\text{Strong}} = [3+, 3-]$$

Finally,

$$Gain(D, Wind) = Entropy(D) - \sum_{v_i \in A_k} \frac{|D_{v_i}|}{|D|} Entropy(D_{v_i})$$
$$= .94 - \frac{8}{14} .811 - \frac{6}{14} 1$$
$$= .048$$

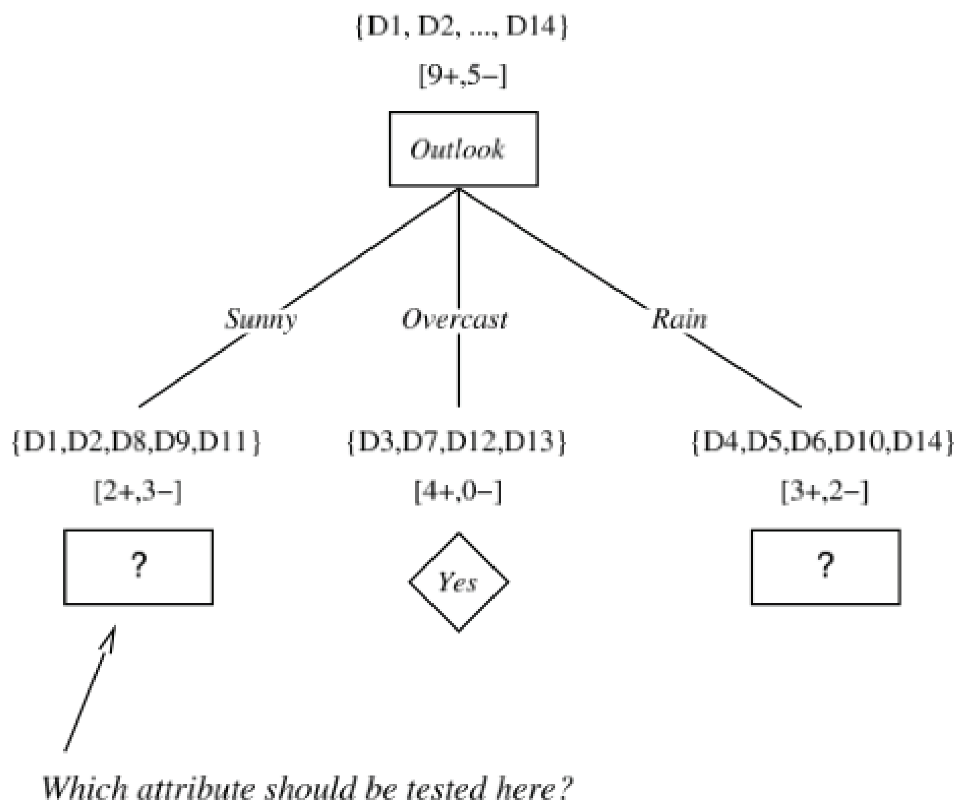
Information Gain

Similarly, for Outlook, Humidity, and Temp,

$$
\begin{aligned}
Gain(D, Outlook) &= .246 \\
Gain(D, Humidity) &= .151 \\
Gain(D, Temp) &= .029
\end{aligned}
$$

$\Rightarrow$ This means the $Outlook$ will become our root more.

Information Gain



$$S_{sunny} = \{D1,D2,D8,D9,D11\}$$

$$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$$

$$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$$

$$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$$

Information Gain

Decision Tree Induction

Basic Algorithm:

1. A $\leftarrow$ the “best" decision attribute for a node N.
2. Assign A as decision attribute for the node N.
3. For each value of A, create new descendant of the node N.
4. Sort training examples to leaf nodes.
5. IF training examples perfectly classified, THEN STOP.
   ELSE iterate over new leaf nodes