Decision Tree Learning

Decision Tree Learning

Decision trees learning is one of the most widely used and practical methods for inductive inference.

It is a method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions.

The learned function is represented by a decision tree (or if-then rules).

Widely used algorithms are:

- ID3
- ASSISTANT
- C4.5

Decision Tree learning methods search a completely expressive hypothesis spaces.

Their inductive bias is a preference for small trees over large trees.

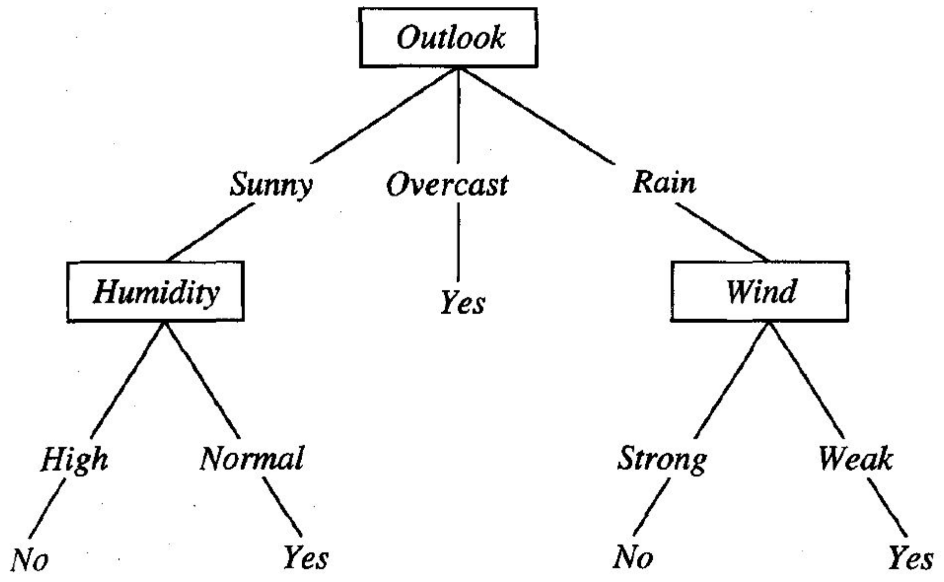
Decision Tree Representation

DTs classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.

For example, the instance

<outlook=sunny, temp=hot, humidity=high, wind=strong> will be classified as a negative instance.



Decision Tree Representation

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

For example, the DT corresponds to the expression:

$$
\begin{aligned}
& (Outlook = Sunny \;\wedge\; Humidity = Normal) \\
\vee \quad & (Outlook = Overcast) \\
\vee \quad & (Outlook = Rain \;\wedge\; Wind = Weak)
\end{aligned}
$$

Appropriate Problems for Decision Tree Learning

Decision Tree learning is generally best suited to problems with the following characteristics:-

- Instances are represented by attribute-value pairs.
- Target function has discrete output values.
- Disjunctive descriptions may be required.
- The training data may contain errors.
- The training data may contain missing attribute values.

ID3 Algorithm
Iterative Dichotomiser 3

ID3 Algorithm

ID3 algorithm learns decision trees by constructing them in a top-down manner.

ID3 algorithm uses a greedy approach to build the tree. No backtracking is performed at any step.

It begins with the question, “which attribute should be tested at the root of the tree?”

The best attribute is selected and used as the test at the root node of the tree.

A descendant of the root node is then created for each possible value of this attribute.

The entire process is then repeated using the training examples with each descendant node to select the best attribute to test at that point in the tree.

It uses information gain as the measure to select the best attribute at each step in the growing tree.

Which Attribute Is the Best Classifier?

The central choice in the ID3 algorithm is selection which attribute to test at each node in the tree.

Information Gain is used to find the best attribute. It measures how well a given attribute separates the training examples according to their target classification.

Entropy

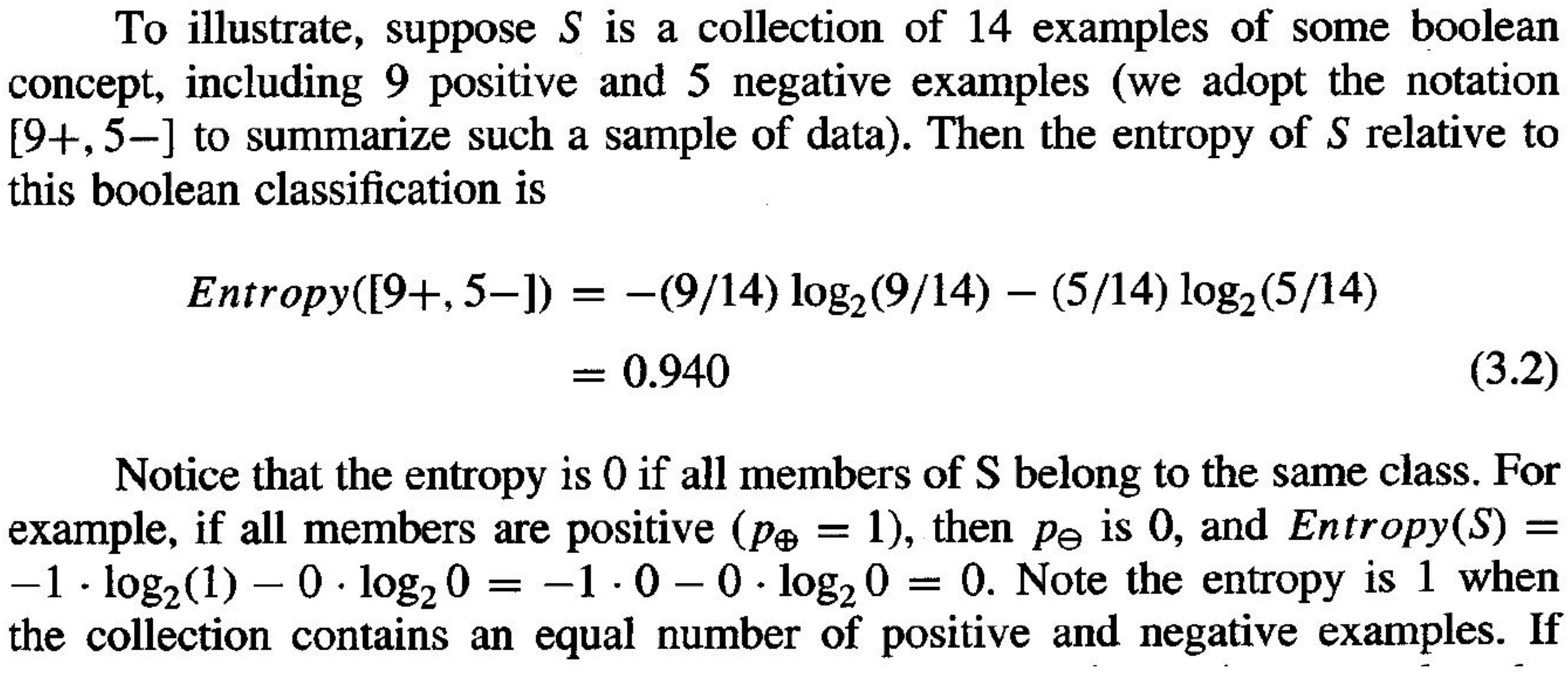
Entropy characterises the (im)purity of an arbitrary collection of examples.

Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{-} \log_2 p_{-}$$

p_+ is the proportion of positive examples in S and p_- is the proportion of negative examples in S.

Example



To illustrate, suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples (we adopt the notation $[9+, 5-]$ to summarize such a sample of data). Then the entropy of S relative to this boolean classification is

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned} \tag{3.2}$$

Notice that the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_{\oplus} = 1$), then $p_{\ominus}$ is 0, and $Entropy(S) = -1 \cdot \log_2(1) - 0 \cdot \log_2 0 = -1 \cdot 0 - 0 \cdot \log_2 0 = 0$. Note the entropy is 1 when the collection contains an equal number of positive and negative examples. If

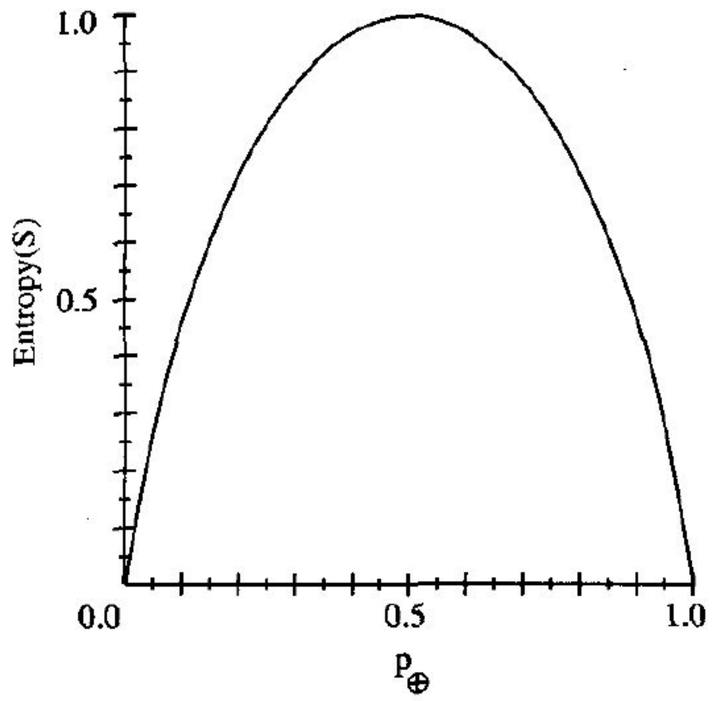
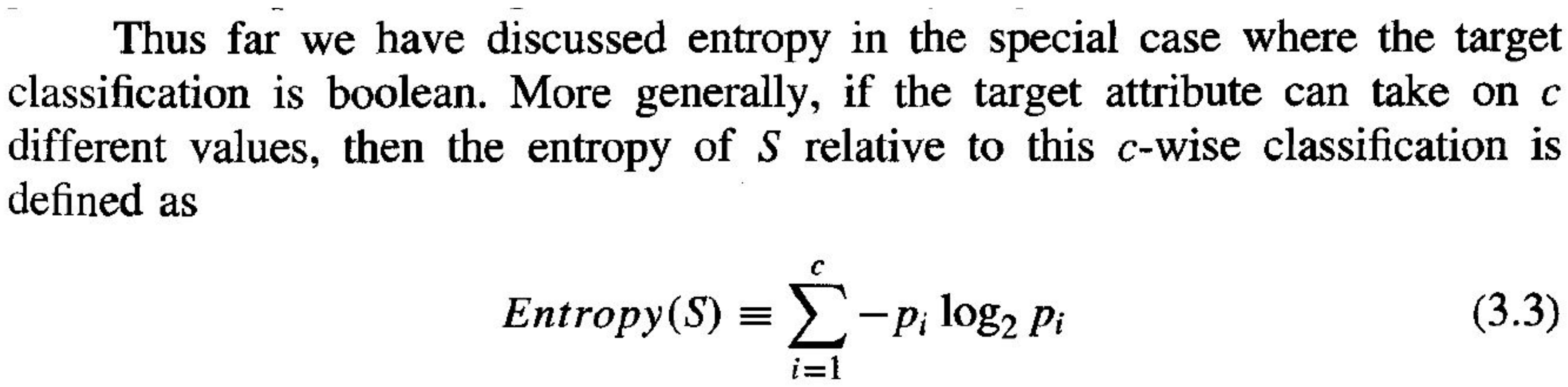


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

entropy is between 0 and 1. Figure 3.2 shows the form of the entropy function relative to a boolean classification, as $p_\oplus$ varies between 0 and 1.

Entropy (for more than two classes)



Thus far we have discussed entropy in the special case where the target classification is boolean. More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \tag{3.3}$$

Information Gain Measures the Expected Reduction in Entropy

Information gain is simply the expected reduction in entropy caused by partitioning the examples to this attribute.

The information gain Gain (S,A) of an attribute A, relative to a collection of examples S, is defined as:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$

Example

For example, suppose S is a collection of training-example days described by attributes including *Wind*, which can have the values *Weak* or *Strong*. As before, assume S is a collection containing 14 examples, $[9+, 5-]$. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have $Wind = Weak$, and the remainder have $Wind = Strong$. The information gain due to sorting the original 14 examples by the attribute *Wind* may then be calculated as

Example

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

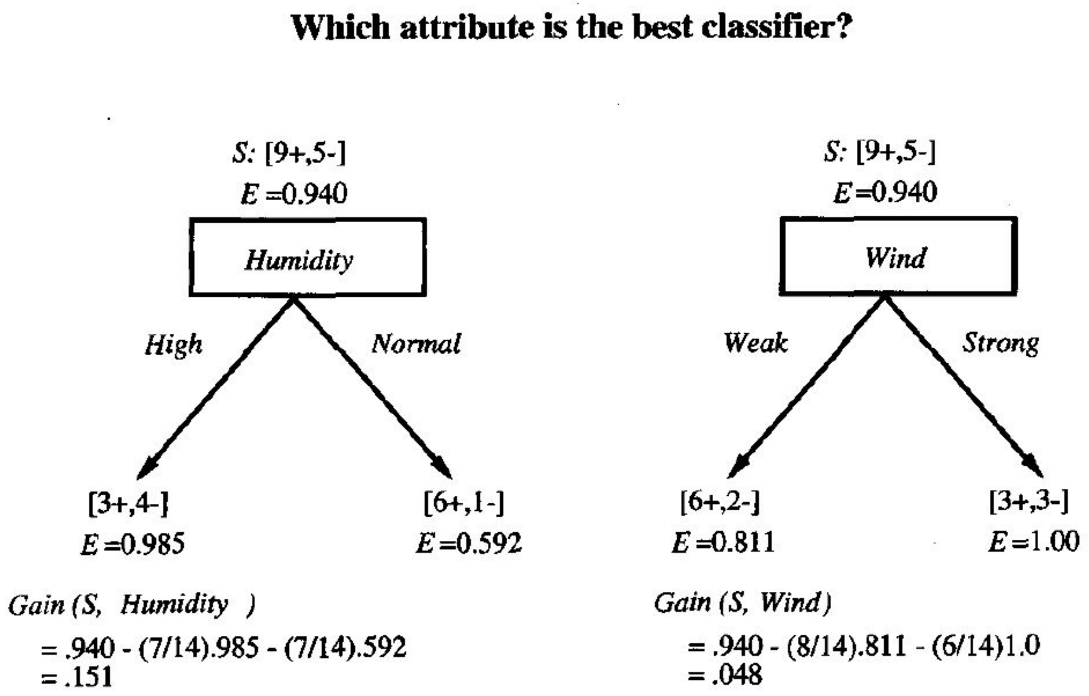


FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, $[9+, 5-]$, sorting these by their *Humidity* produces collections of $[3+, 4-]$ $(Humidity = High)$ and $[6+, 1-]$ $(Humidity = Normal)$. The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

$$\text{Gain}(S, \text{Outlook}) = Entropy(S) - \sum_{v \in Values(\text{outlook})} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= 0.94 - \left(\frac{5}{14}\right)(-0.4 \log_2 0.4 - 0.6 \log_2 0.6) - \left(\frac{4}{14}\right)(-1 \log_2 1 - 0 \log_2 0) - \left(\frac{5}{14}\right)(-0.6 \log_2 0.6 - 0.4 \log_2 0.4)$$

$$\text{Gain}(S, \text{Outlook}) = 0.246$$

$$\begin{aligned}\text{Gain}(S, \text{Temp}) = 0.94 - \Big[& \frac{4}{14} \cdot \left(-\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) + \\ & \frac{4}{14} \cdot \left(-\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right) + \\ & \frac{6}{14} \cdot \left(-\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right)\right)\Big]\end{aligned}$$

$$\text{Gain}(S, \text{Temp}) = 0.029$$

The Gain values for all the attributes are:

$$\text{Gain}\left(S, \text{Outlook}\right) = 0.246$$

$$\text{Gain}\left(S, \text{Humidity}\right) = 0.151$$

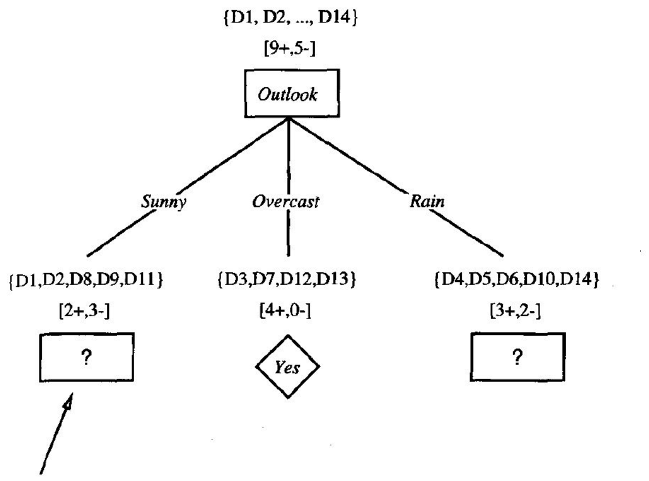
$$\text{Gain}\left(S, \text{Wind}\right) = 0.048$$

$$\text{Gain}\left(S, \text{Temp}\right) = 0.029$$

According to the information gain measure, the *Outlook* attribute provides the best prediction of the target attribute, *PlayTennis* over the training examples.

Therefore, *Outlook* is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values.

Every example for *Outlook* = *Overcast* is also a positive example. Therefore, this node of the tree becomes the leaf node with classification *PlayTennis*=*Yes*.



Which attribute should be tested here?

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

Building the Tree Further

The process of selecting a new attribute and partitioning the training examples is now repeated for each non-terminal node, using only the training examples associated with this node.

Attributes that have been incorporated higher in the tree are excluded.

This process continues for each new leaf node until either of two conditions is met:

(1) Every attribute has already been included along this path through the tree, or
(2) The training examples associated with this leaf node all have same target attribute value (i.e. their entropy is zero).

S_{sunny}

day	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

$$Entropy_{\text{sunny}} = -0.4\log_2 0.4 - 0.6\log_2 0.6 = 0.97$$

$$\text{Gain}\left(S_{\text{sunny}}, \text{Humidity}\right) = 0.97 - \left[\tfrac{3}{5}\cdot 0 + \tfrac{2}{5}\cdot 0\right] = 0.97$$

$$\text{Gain}\left(S_{\text{sunny}}, \text{Temp}\right) = 0.97 - \left[\tfrac{2}{5}\cdot 0 + \tfrac{2}{5}\cdot 1 + \tfrac{1}{5}\cdot 0\right] = 0.57$$

$$\text{Gain}\left(S_{\text{sunny}}, \text{Wind}\right) = 0.97 - \left[\tfrac{2}{5}\cdot 1 + \tfrac{3}{5}\cdot\left(-0.33\log_2 0.33 - 0.66\log_2 0.66\right)\right] = 0.021$$

For the set S_{sunny}, the attribute *Humidity* provides the best prediction for the target concept PlayTennis.

Therefore, *Humidity* is selected as the decision attribute for this node, and branches are created below this node for each of its possible values.

Every example for *Humidity = High* is also a *negative* example and every example for *Humidity=Normal* is a *positive* example. Therefore, these nodes of the tree become the leaves nodes with classification *PlayTennis=No and Yes respectively*.

S_{rain}

$$Entropy\left(S_{\text{rain}}\right) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.97$$

$$\text{Gain}\left(S_{\text{rain}}, Humidity\right) = 0.97 - [0.4 \cdot 1 + 0.6(-0.66\log_2 0.66 - 0.33\log_2 0.33)]$$

$$\text{Gain}\left(S_{\text{rain}}, Humidity\right) = 0.021$$

$$\text{Gain}\left(S_{\text{rain}}, \text{Wind}\right) = 0.97 - [0.4 \cdot 0 + 0.6 \cdot 0]$$

$$\text{Gain}\left(S_{\text{rain}}, \text{Wind}\right) = 0.97$$

S_{rain}

$$\text{Gain}\left(S_{\text{rain}}, Temp\right) = 0.97 - [0.4 \cdot 1 + 0.6(-0.66 \log_2 0.66 - 0.33 \log_2 0.33)]$$

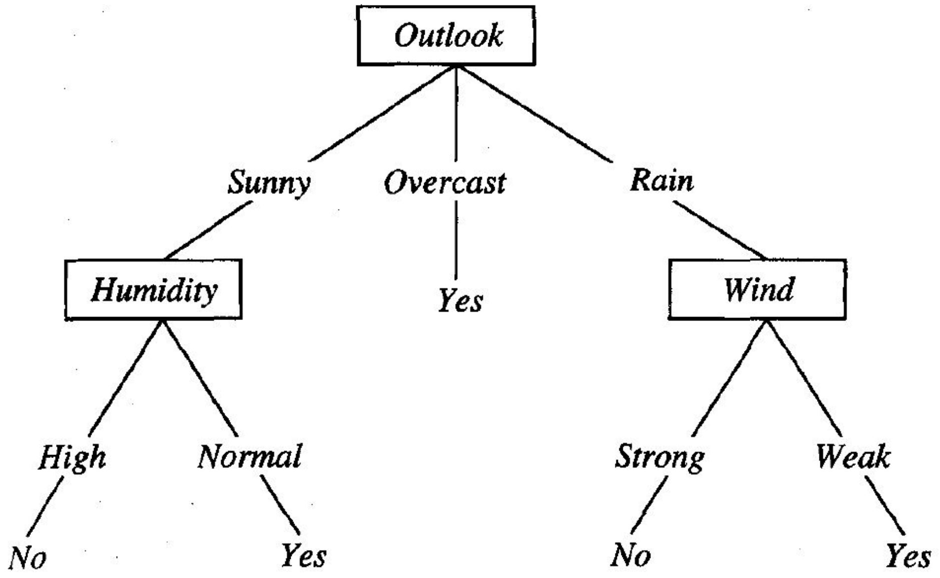
$$\text{Gain}\left(S_{\text{rain}}, Temp\right) = 0.021$$

For the set S_{rain}, the attribute *Wind* provides the best prediction for the target concept *PlayTennis*.

Therefore, *Wind* is selected as the decision attribute for this node, and branches are created below this node for each of its possible values.

Every example for *Wind = Strong* is a *negative* example and every example for *Wind=Weak* is a *positive* example. Therefore, these nodes of the tree become the leaves nodes with classification *PlayTennis=No and Yes respectively*.

Resulting Decision Tree



Inductive Bias in Decision Tree Learning

Inductive Bias is the set of assumptions that, together with training data, deductively justify the classifications assigned by the learner to future instances.

ID3 algorithm -

- Chooses the first acceptable tree it encounters in its simple-to-complex hill-climbing search through the space of possible trees.
- It selects in favour of shorter trees over longer ones.
- It selects trees that place the attributes with highest information gain closest to the root.

Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.

Why Prefer Short Hypotheses?

Occam’s Razor: Prefer the simplest hypothesis that fits the data.

Supporting Argument

There are fewer short hypotheses than long ones, it is less likely that one will find a short hypothesis that coincidentally fits the training data. In contrast, there are often many very complex hypotheses that fit the current training data but fail to generalise correctly to subsequent data.

Strengths of ID3

ID3’s hypothesis space of all decision trees is a complete space of finite discrete values functions, relative to the available attributes.

ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis. This makes the resulting search much less sensitive to errors in individual training examples.

Limitations of ID3 Algorithm

ID3 maintains only a single current hypothesis as it searches through the space of decision trees. It does not have ability to determine how many alternative decision trees are consistent with the available training data.

ID3 never performs backtracking in its search. This makes it susceptible to the risk of converging to a locally optimal solution.

References

- Chapter-3, Machine Learning by Tom M. Mitchell.