



THE UNIVERSITY
of ADELAIDE



CRICOS PROVIDER 00123M

Faculty of SET / School of Computer and Mathematical Sciences
COMP SCI 3007/7059/7659
Artificial Intelligence
Decision Tree Learning

adelaide.edu.au

seek LIGHT



Acknowledgement of Country

We acknowledge and pay our respects to the Kaurna people, the traditional custodians whose ancestral lands we gather on.

We acknowledge the deep feelings of attachment and relationship of the Kaurna people to the country and we respect and value their past, present and ongoing connection to the land and cultural beliefs.

Decision Tree Learning

AIMA C18.3

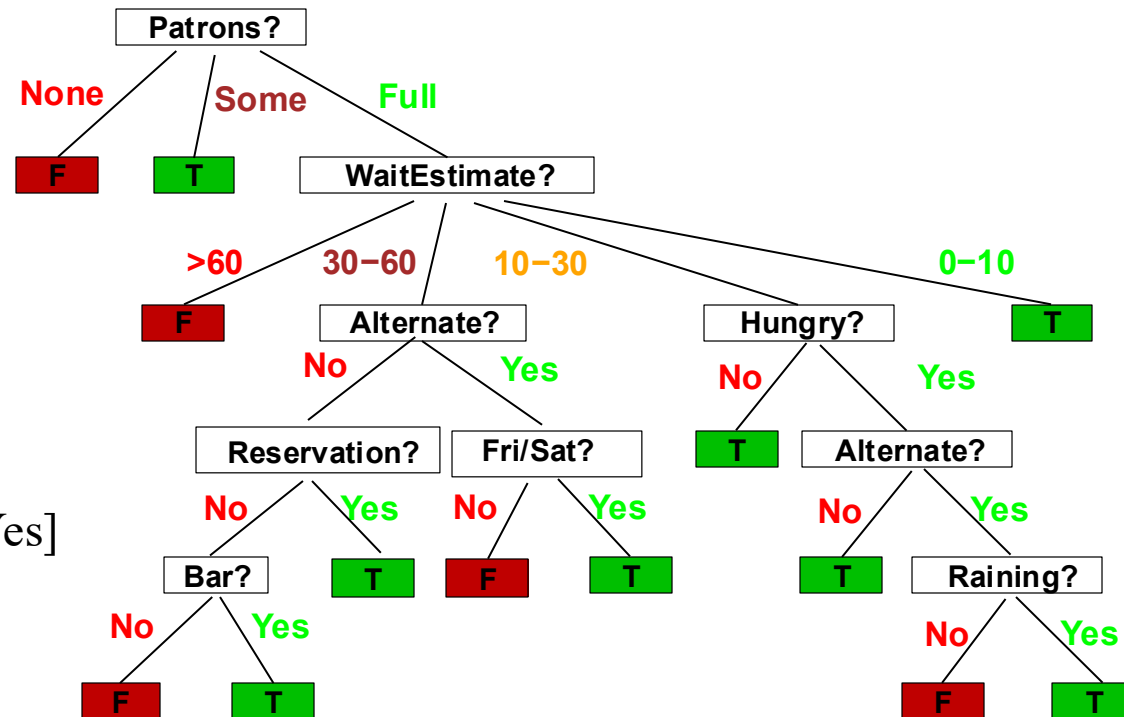
Decision Trees

Decision trees is one of the simplest methods for supervised learning. It can be applied to both regression & classification.

Example:

A decision tree for deciding whether to wait for a place at a restaurant.

Target *WillWait* (i.e., y) can be *True* or *False*.



Input X: [Full, 40, Yes, Yes]

Output y: T

Decision Trees (cont.)

At each node of a tree, a test is applied which sends the query sample down one of the branches of the node.

This continues until the query sample arrives at a terminal or leaf node. Each leaf node is associated with a value: a class label in classification, or a numeric value in regression.

The value of the leaf node reached by the query sample is returned as the output of the tree.

Example:

*Patrons = Full, Price = Expensive, Rain = Yes, Reservation = Yes,
Hungry = Yes, Fri = No, Bar = Yes, Alternate = Yes, Type = Thai,
WaitEstimate = 0 – 10*

What is *WillWait*?

The Training Set

Training set for a decision tree (each row is a sample):

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
X_1	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>0–10</i>	<i>T</i>
X_2	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>30–60</i>	<i>F</i>
X_3	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>Some</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>0–10</i>	<i>T</i>
X_4	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>10–30</i>	<i>T</i>
X_5	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>>60</i>	<i>F</i>
X_6	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Italian</i>	<i>0–10</i>	<i>T</i>
X_7	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>0–10</i>	<i>F</i>
X_8	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Thai</i>	<i>0–10</i>	<i>T</i>
X_9	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>>60</i>	<i>F</i>
X_{10}	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>Italian</i>	<i>10–30</i>	<i>F</i>
X_{11}	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>0–10</i>	<i>F</i>
X_{12}	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>30–60</i>	<i>T</i>

An **attribute** or **feature** is a characteristic of the situation that we can measure.

The samples could be **observations of previous decisions taken**.

The Training Set (cont.)

We can re-arrange the samples as follows:

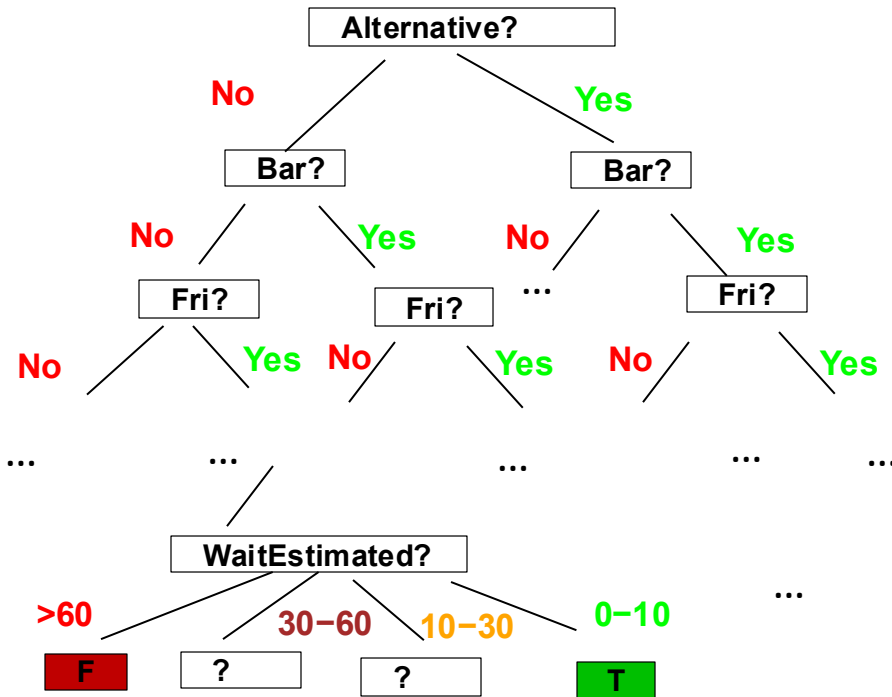
$$\underbrace{\mathbf{x}_1 = \left\{ \begin{array}{c} T \\ F \\ F \\ T \\ \textit{Some} \\ \$\$ \$ \\ F \\ T \\ \textit{French} \\ 0-10 \end{array} \right\}}_{\textit{sample 1}}, y_1 = T \quad \underbrace{\mathbf{x}_2 = \left\{ \begin{array}{c} T \\ F \\ F \\ T \\ \textit{Full} \\ \$ \\ F \\ F \\ \textit{Thai} \\ 30-60 \end{array} \right\}}_{\textit{sample 2}}, y_2 = F \quad \dots$$

i.e., each sample consists of a 10-dimensional vector \mathbf{x}_i of discrete feature values, and a target label y_i which is Boolean.

Learning Decision Trees from Observations

Trivial solution for building decision trees — construct a path to a leaf for each sample.

Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
X_1	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>0-10</i>	<i>T</i>
X_2	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>30-60</i>	<i>F</i>



If the maximum number of branches here is denoted as b , and the depth of the tree is n , the total leaves could reach b^n .

For binary tree, the number of leaves could reach: 2^n .

Learning Decision Trees from Observations

Trivial solution for building decision trees — construct a path to a leaf for each sample.

- ▶ Merely memorising the observations (not extracting any patterns from the data).
- ▶ Produces a consistent but excessively complex hypothesis (recall Occam's Razor).
- ▶ Unlikely to generalise well to new/unseen observations.

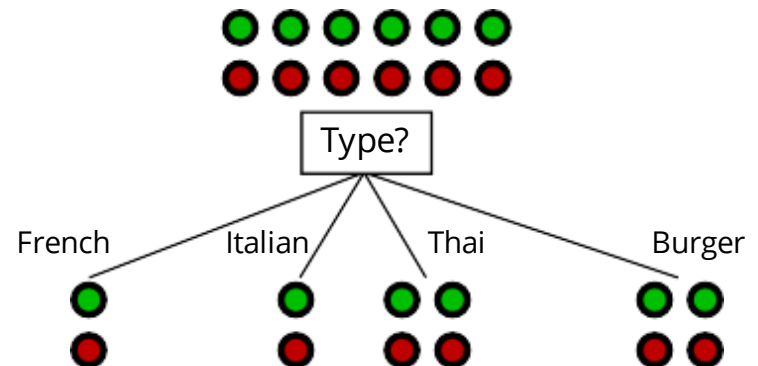
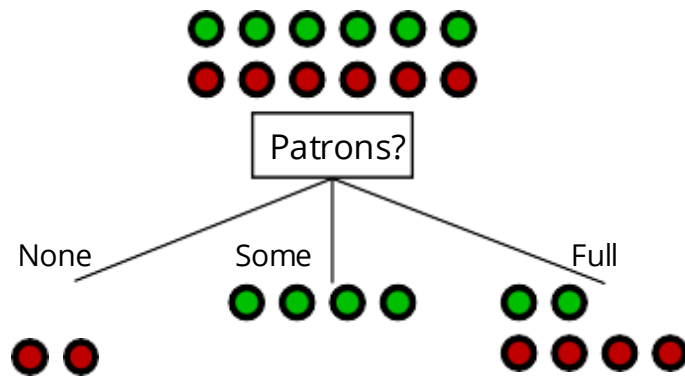
We should aim to build the smallest tree that is consistent with the observations.

One way to do this is to recursively test **the most important attributes first**.

Learning Decision Trees from Observations (cont.)

Example:

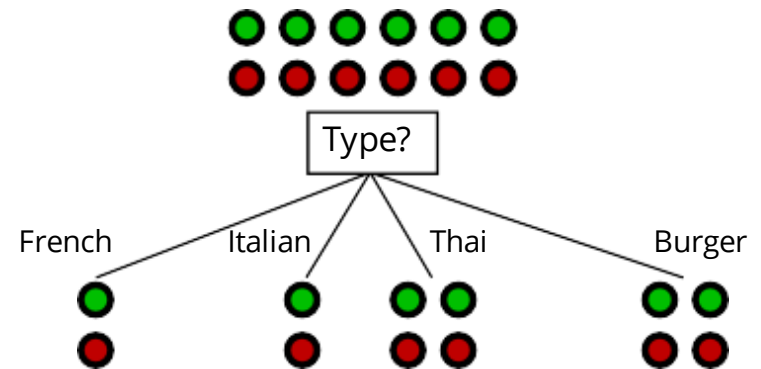
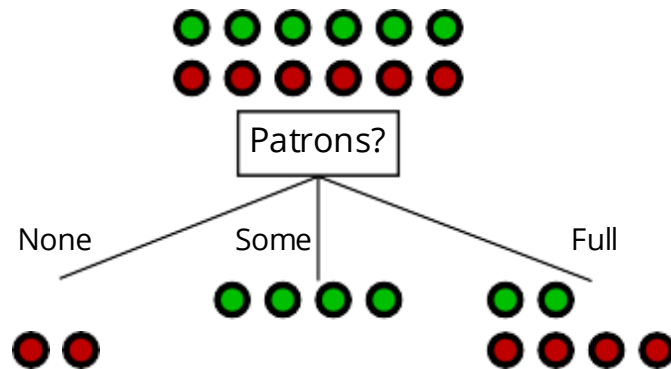
Test Patrons or Type first?



Learning Decision Trees from Observations (cont.)

Example:

Test Patrons or Type first?



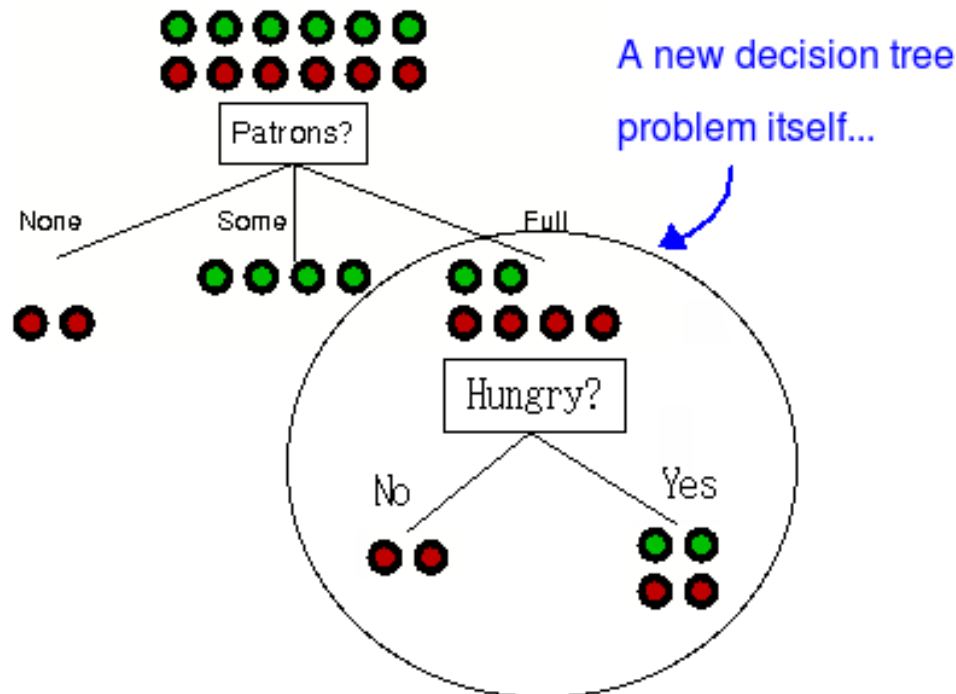
An important (or good) attribute splits samples into groups that are (ideally) all positive or negative.

Testing good attributes first allows us to minimise the tree depth.

Therefore Patrons is more important than Type.

Learning Decision Trees from Observations (cont.)

After the first attribute splits the samples, the remaining samples form decision tree problems themselves (or subtrees) but with less samples and one less attribute, e.g.,

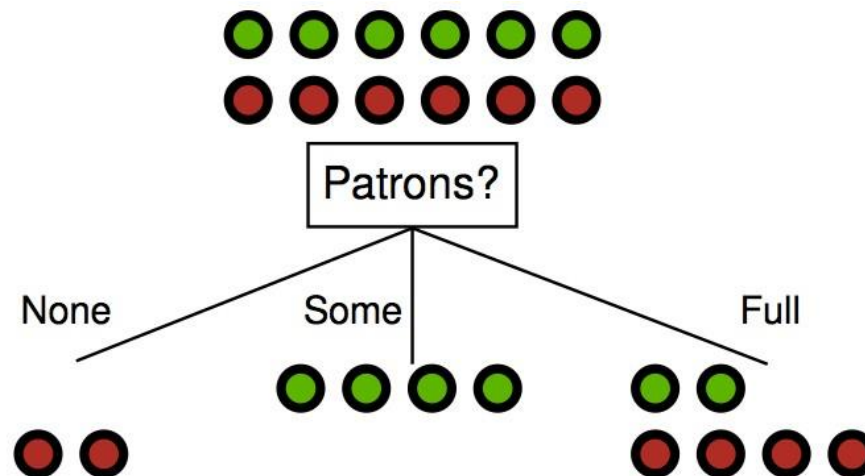


This suggests a recursive approach to build decision trees.

Decision Tree Learning (DTL) Algorithm (cont.)

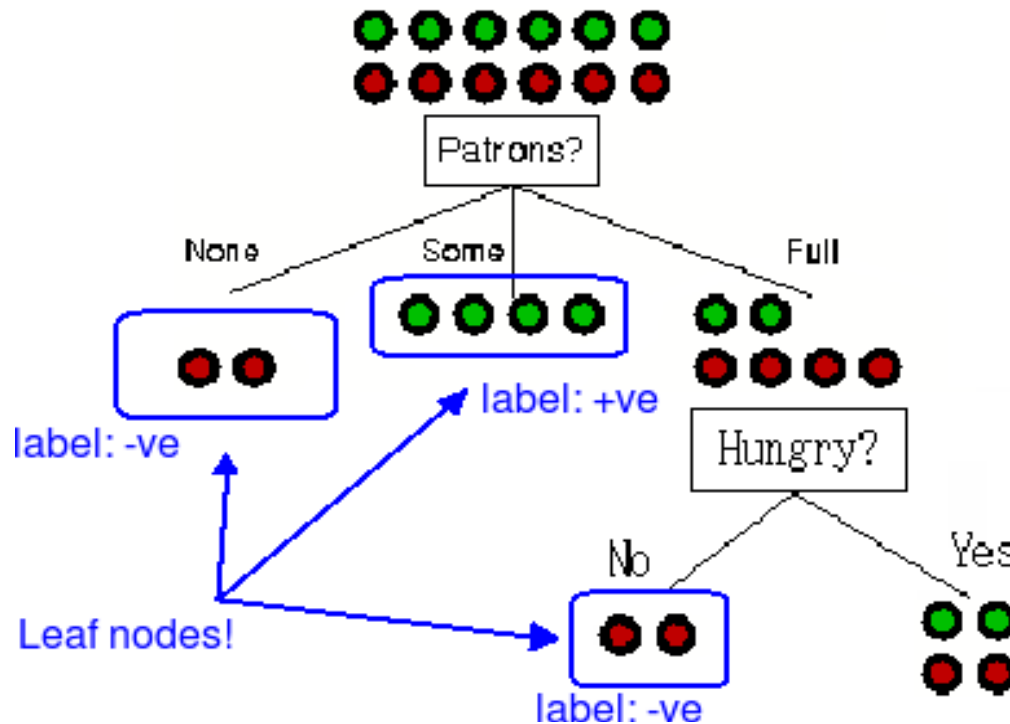
Four basic underlying ideas of the algorithm:

1. If there are some positive and negative samples, **then choose the best attribute** to split them, e.g., test Patrons at the root.



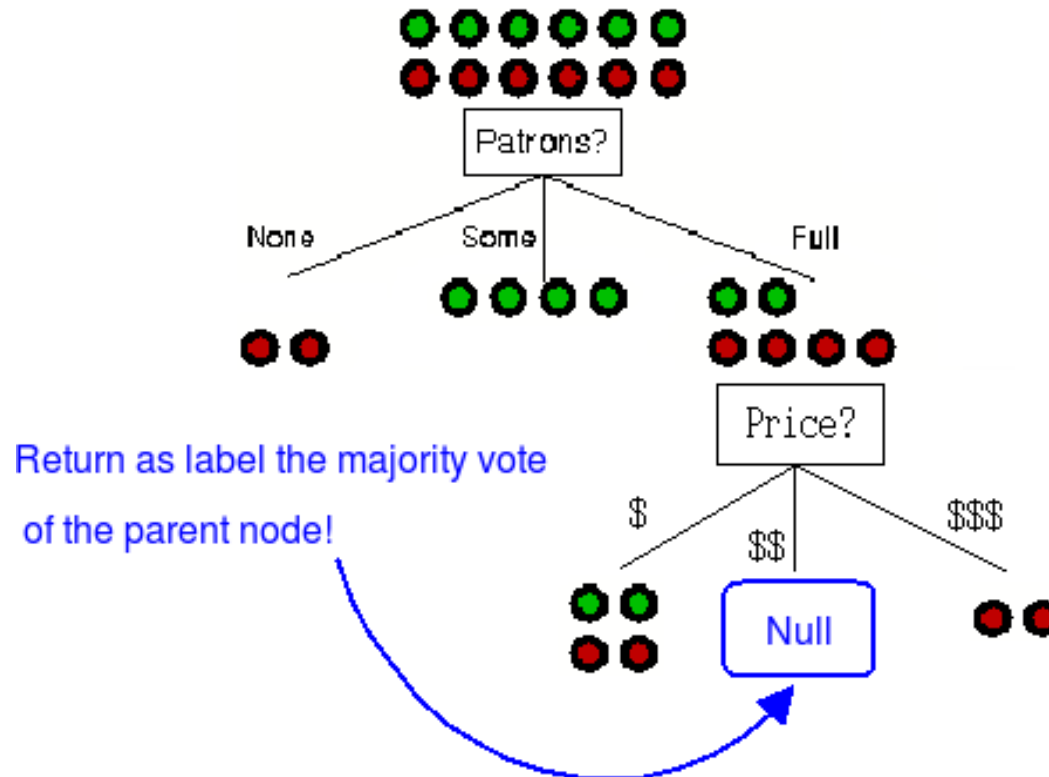
Decision Tree Learning (DTL) Algorithm (cont.)

2. If all the remaining samples are all positive or all negative, we have reached a leaf node. **Assign label** as positive (or negative), e.g.,



Decision Tree Learning (DTL) Algorithm (cont.)

3. If there are no samples left, it means that no such sample has been observed. Return a default value calculated from **the majority classification at the node's parent**, e.g.,



Decision Tree Learning (DTL) Algorithm (cont.)

4. If there are no attributes left, but both positive and negative samples, it means that these samples have exactly the same feature values but different classifications. This may happen because
 - ▲ some of the data could be incorrect, or
 - ▲ the attributes do not give enough information to describe the situation fully (i.e. we lack other useful attributes), or
 - ▲ the problem is truly non-deterministic, i.e., given two samples describing exactly the same conditions, we may make different decisions.

Solution: Call it a leaf node and assign the majority vote as the label.

Decision Tree Learning (DTL) Algorithm

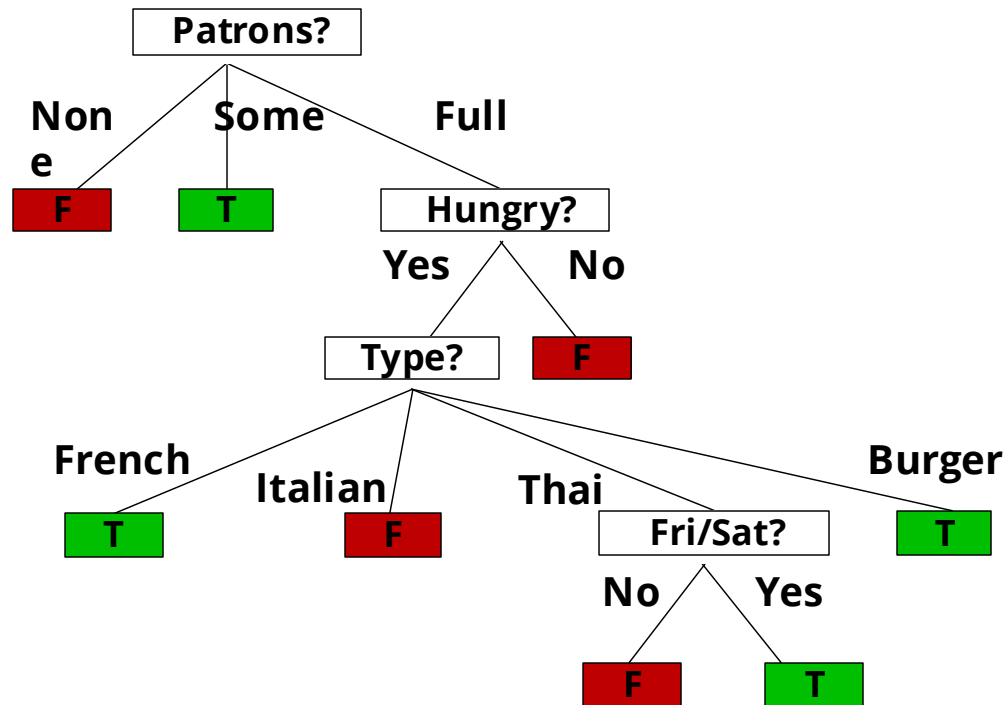
Aim: Find the smallest tree consistent with the training samples.

Idea: Recursively choose “most important” attribute as root of (sub)tree.

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
       $examples_i \leftarrow \{\text{elements of } examples \text{ with } best = v_i\}$ 
      subtree ← DTL(examplesi, attributes – best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
    return tree
```


Decision Tree Learning (DTL) Algorithm (cont.)

- ▶ Result of applying DTL on the observations:



- ▶ Simpler than the tree shown in Page 4.
-

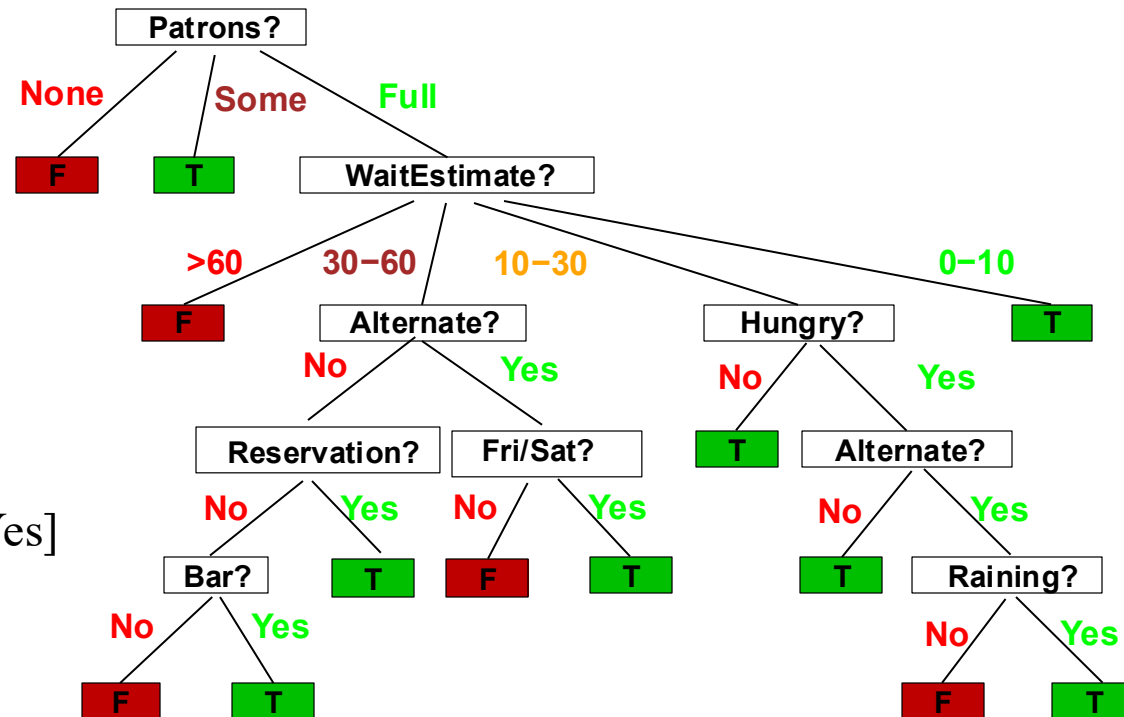
Decision Trees

Decision trees is one of the simplest methods for supervised learning. It can be applied to both regression & classification.

Example:

A decision tree for deciding whether to wait for a place at a restaurant.

Target *WillWait* (i.e., y) can be *True* or *False*.



Input X: [Full, 40, Yes, Yes]

Output y: T

Choosing the Best Attribute

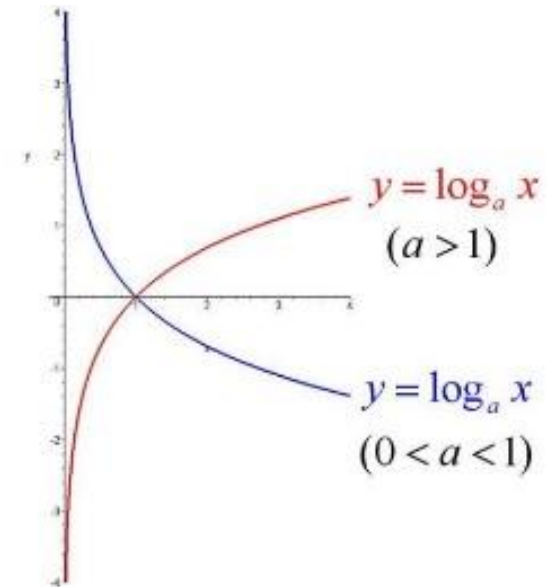
We need a measure of “good” and “bad” for attributes.

One way to do is to compute the information content at a node R :

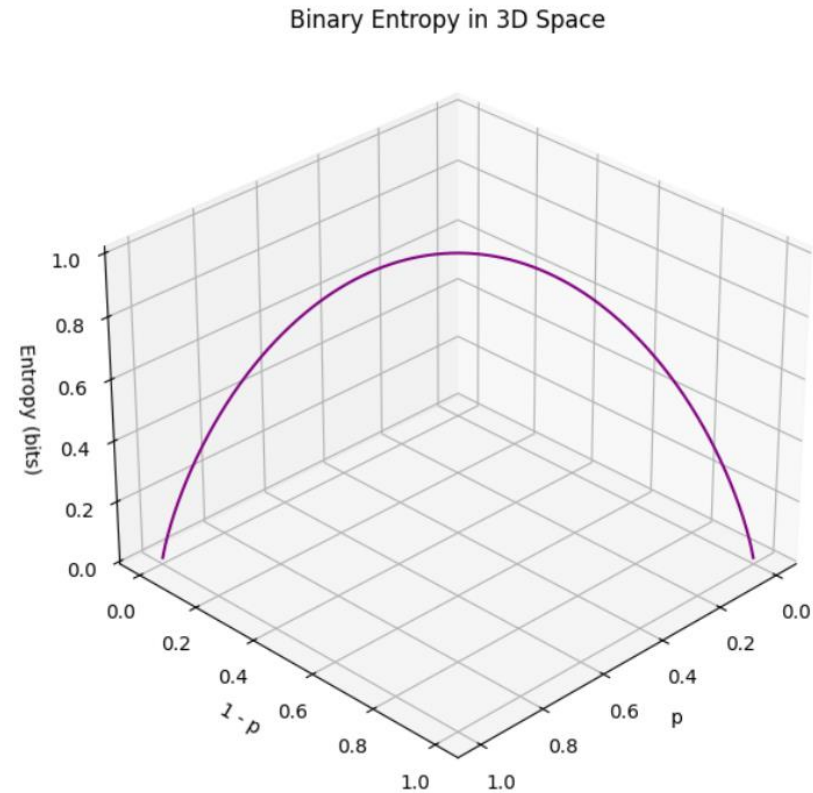
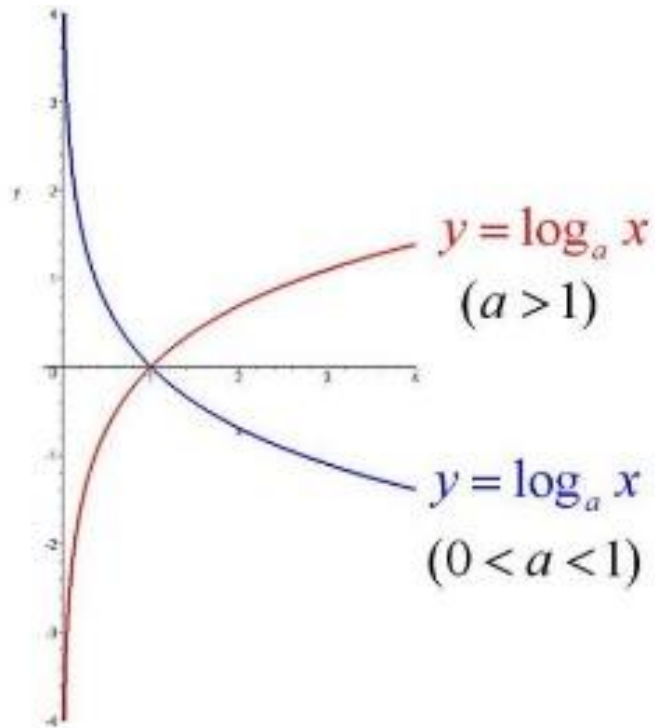
$$I(R) = \sum_{i=1}^L -P(c_i) \log_2 P(c_i)$$

where $\{c_1, \dots, c_L\}$ are the L class labels present at the node, and $P(c_i)$ is the probability of getting class c_i at the node.

A unit of information is called a “bit”.



Information Content (Shannon entropy)



$$I(R) = \sum_{i=1}^L -P(c_i) \log_2 P(c_i)$$

Choosing the Best Attribute (cont.)

Example:

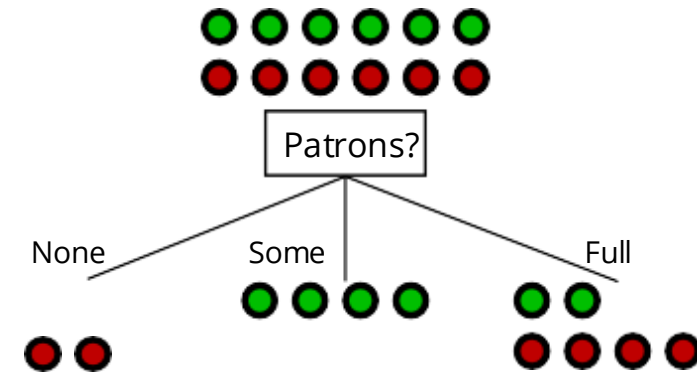
At the root node of the restaurant problem, $c_1 = \text{True}$ (i.e., will wait), $c_2 = \text{False}$ (i.e., will not wait) and there are 6 True samples and 6 False samples.

Therefore:

$$P(c_1) = \frac{\text{no. of samples} = c_1}{\text{total no. of samples}} = \frac{6}{6 + 6} = 0.5$$

$$P(c_2) = \frac{\text{no. of samples} = c_2}{\text{total no. of samples}} = \frac{6}{6 + 6} = 0.5$$

$$I(\text{Root}) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1 \text{ bit}$$



What are the maximum and minimum attainable values for information content?

In general, the amount of information will be maximum when all classes are equally likely, and be minimum when the node is homogeneous (all samples have the same labels).

Choosing the Best Attribute (cont.)

An attribute A will divide the samples at a node into different subsets (or child nodes) where A has M distinct values $\{v_1, \dots, v_M\}$:

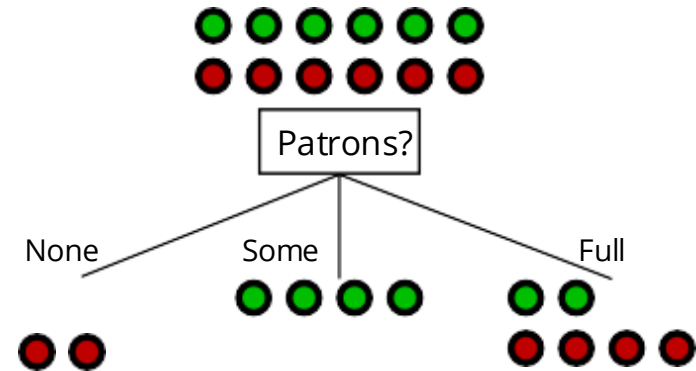
$$E_{A=v_1}, \dots, E_{A=v_M}.$$

Generally each subset $E_{A=v_i}$ will have samples of different labels, so if we go along that branch we will need an additional of $I(E_{A=v_i})$ bits of information.

Example:

In the restaurant problem, at the branch ***Patrons*** = ***Full*** of the root node, we have $c_1 = \text{True}$ with 2 samples and $c_2 = \text{False}$ with 4 samples, therefore:

$$I(E_{Patrons=Full}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.9183 \text{ bits}$$

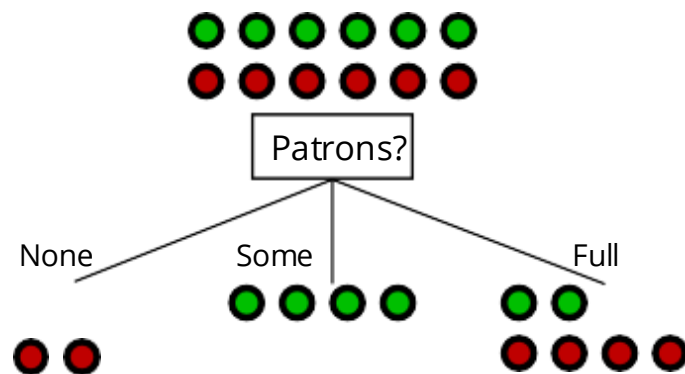


Choosing the Best Attribute (cont.)

Denote by $P(A = v_i)$ as the probability of a sample to follow the branch $A = v_i$.

Example:

For the restaurant problem, at the root node



$$\begin{aligned} P(\textit{Patrons} = \textit{Full}) &= \frac{\text{No. of samples where } \textit{Patrons} = \textit{Full}}{\text{No. of samples at the root node}} \\ &= \frac{6}{12} = 0.5 \end{aligned}$$

Choosing the Best Attribute (cont.)

After testing attribute A , we need a remainder of

$$Remainder(A) = \sum_{i=1}^M P(A = v_i) I(E_{A=v_i})$$

bits to classify the samples.

The **information gain** in testing attribute A is the difference between the original information content (on root) and the new information content (on children):

$$\begin{aligned} Gain(A) &= I(R) - Remainder(A) \\ &= I(R) - \sum_{i=1}^M P(A = v_i) I(E_{A=v_i}) \end{aligned}$$

where $\{E_{A=v_1}, \dots, E_{A=v_M}\}$ are the child nodes of R after testing attribute A .

Choosing the Best Attribute (cont.)

The key idea behind the CHOOSE-ATTRIBUTE function in the DTL algorithm is to choose the attribute that gives the maximum information gain.

Example:

In the restaurant problem, at the root node

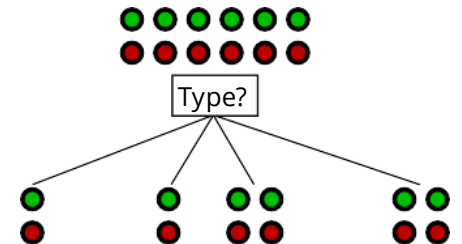
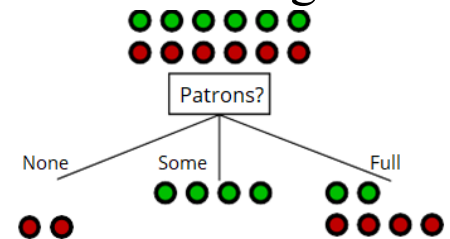
$$\begin{aligned} \text{Gain}(\textit{Patrons}) &= 1 - \frac{2}{12}(-\log_2 1) - \frac{4}{12}(-\log_2 1) - \frac{6}{12}\left(-\frac{2}{6}\log_2 \frac{2}{6} - \frac{4}{6}\log_2 \frac{4}{6}\right) \\ &\approx 0.5409 \text{ bits.} \end{aligned}$$

With similar calculations,

$$\text{Gain}(\textit{Type}) = 0$$

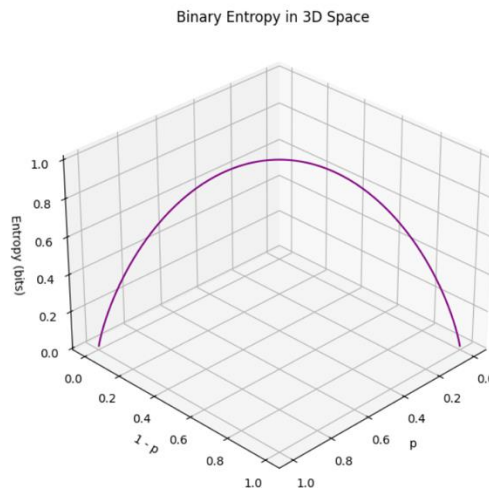
confirming that *Patrons* is a better attribute than *Type*.

In fact at the root *Patrons* gives the highest information gain.



Impurity

- A different point of view is to regard $I(\text{node})$ as a measure of **impurity**. The more “mixed” the samples are at a node (i.e. has equal proportions of all class labels), the higher the impurity value. On the other hand, a homogeneous node (i.e. has samples of one class only) will have zero impurity.
- The $\text{Gain}(A)$ value can thus be viewed as the amount of reduction in impurity if we split according to A .
- This affords the intuitive idea that we grow trees by **recursively trying to obtain leaf nodes which are as pure as possible**.



Concluding Remarks

- Decision tree learning use information gain to select the best splitting attributes.
 - For categorical attributes, do not repeat the check on the same attributes.
 - DTL could be used for both classification and regression.
 - Limitation of DTL is easy to overfit.
-