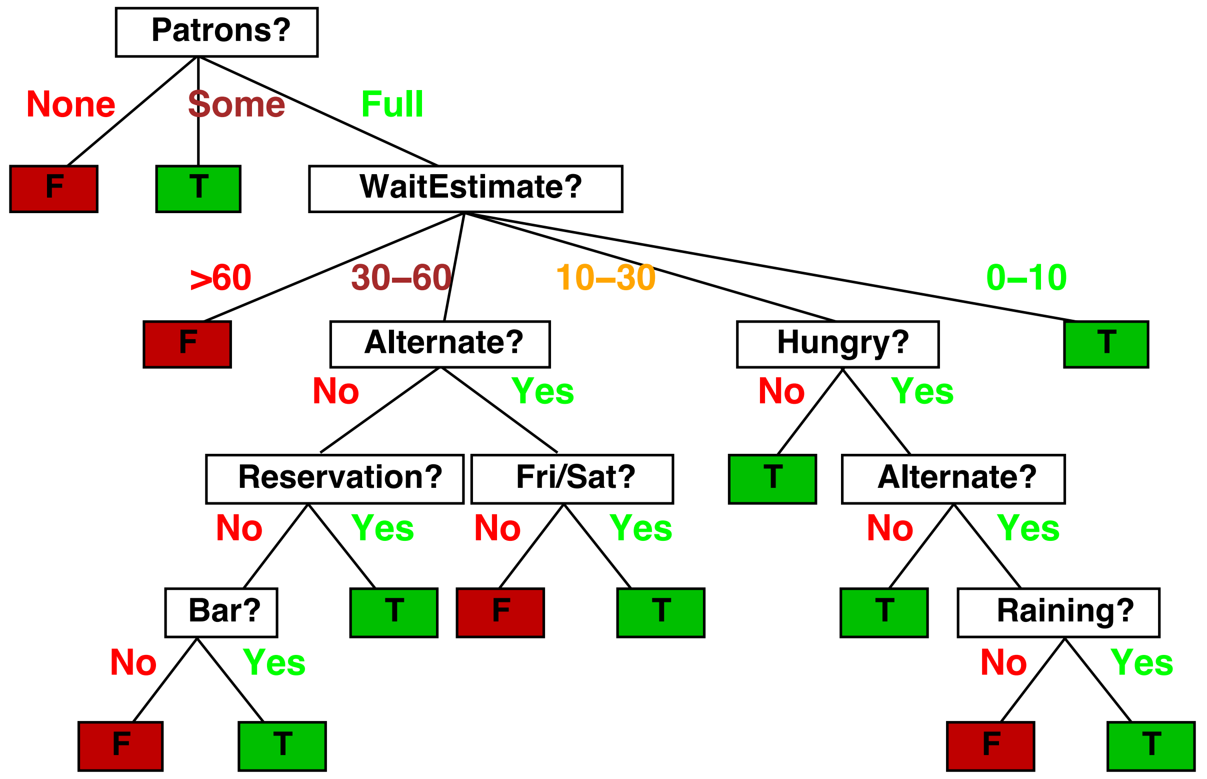
Decision Trees

[Many slides adapted from those created by colleagues at Adelaide University

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 restaurant. Target $WillWait$ can be $True$ or $False$.



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

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:

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

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.

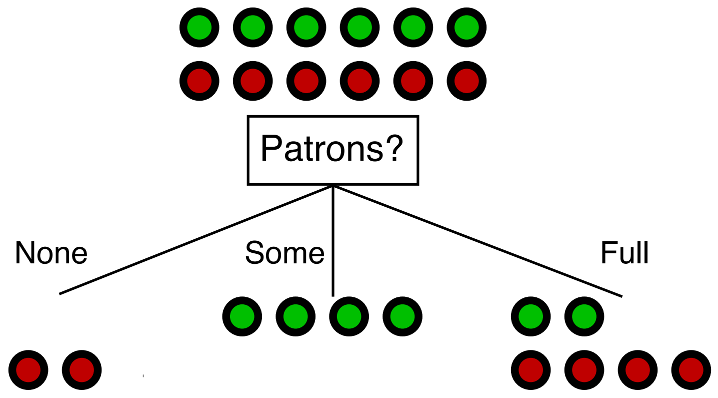
- Merely memorising the observations (not extracting any patterns from the data).
- Produces a consistent but excessively complex hypothesis (recall **Occam's Razor**).
- Unlikely to generalise well to new/unseen observations.

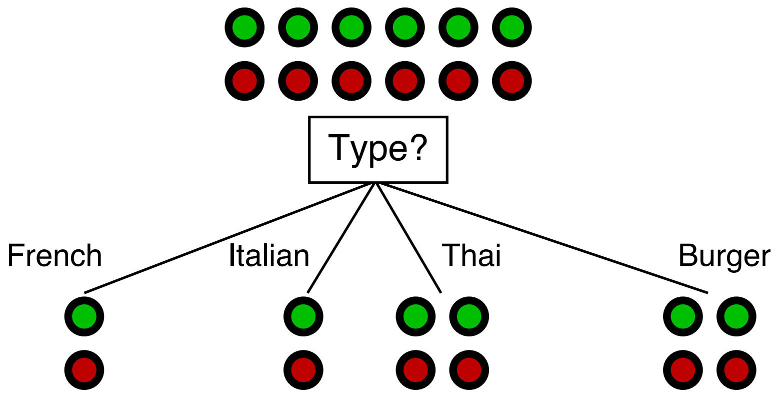
We should aim to build the smallest tree that is consistent with the observations. One way to do this is to **recursively test the most important attributes first**.

Learning Decision Trees from Observations (cont.)

Example:

Test $Patrons$ or $Type$ first?

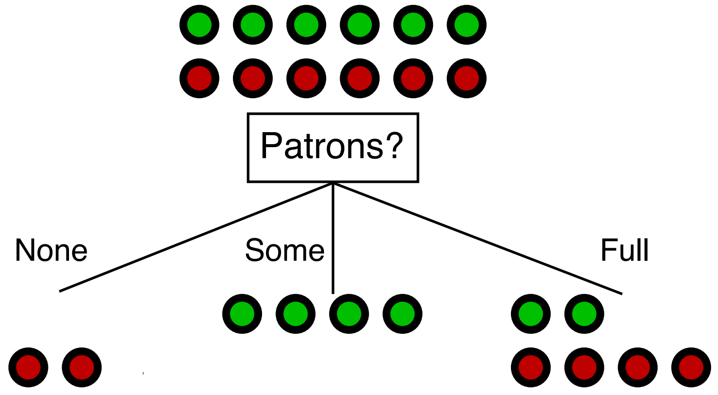


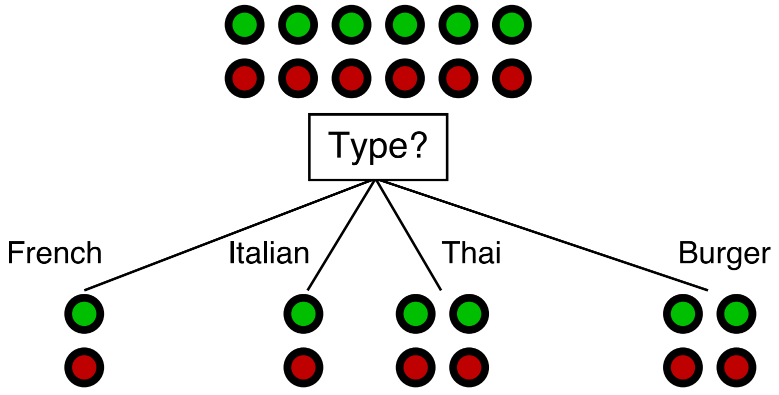


Learning Decision Trees from Observations (cont.)

Example:

Test $Patrons$ or $Type$ first?





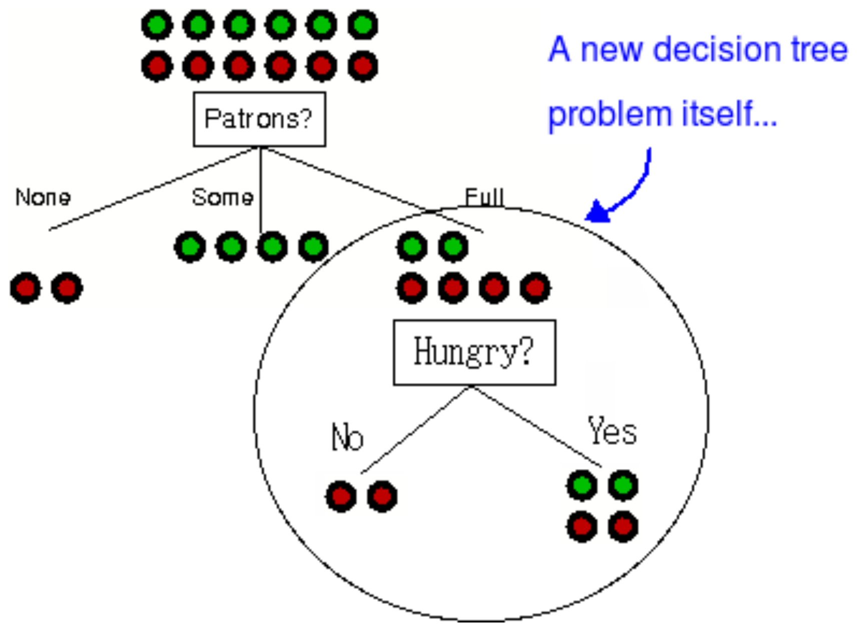
An important (or good) attribute splits samples into groups that are (ideally) all positive or negative.

Therefore $Patrons$ is more important than $Type$.

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

Learning Decision Trees from Observations (cont.)

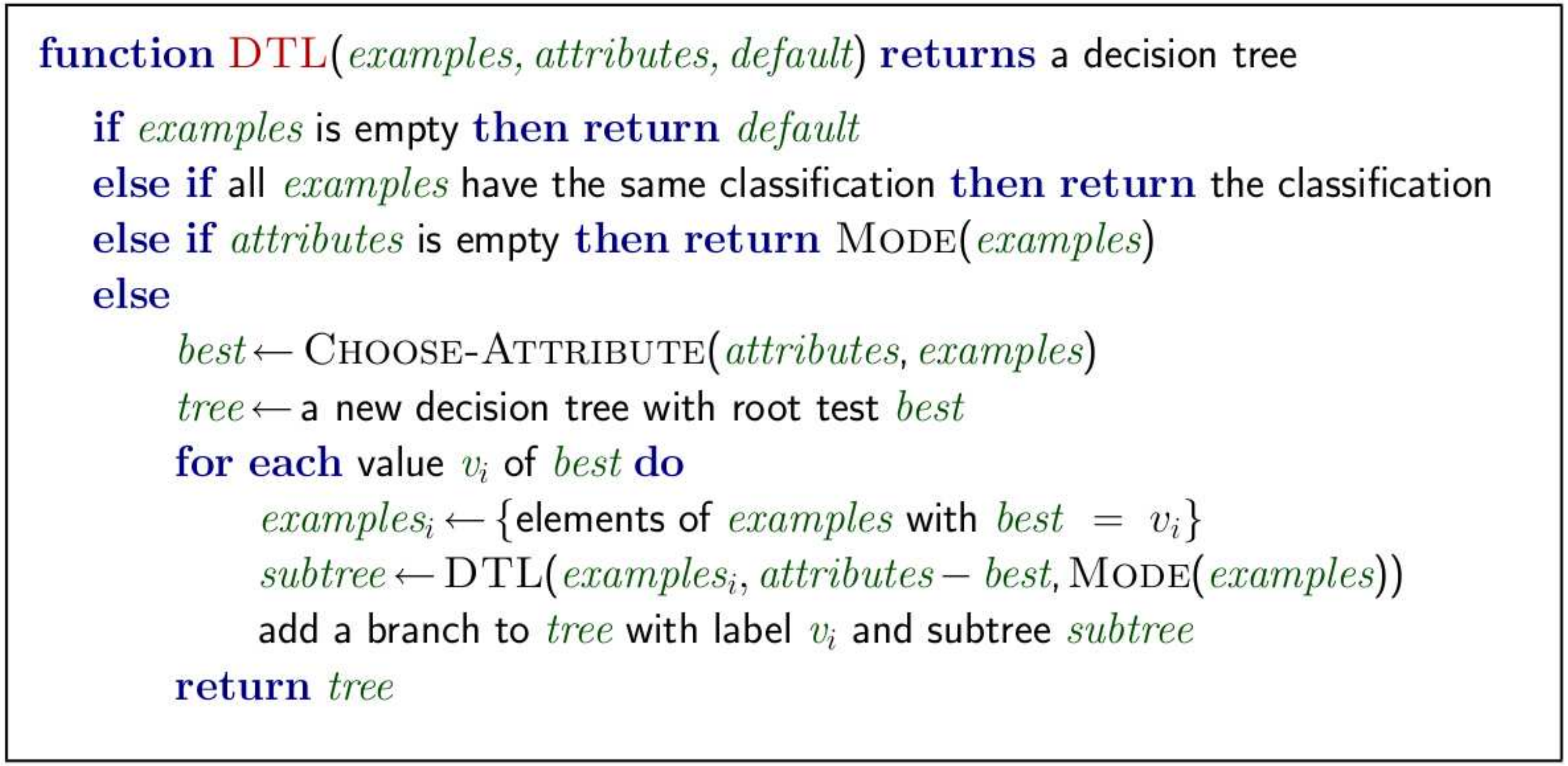
After the first attribute splits the samples, the remaining samples become decision tree problems themselves (or **subtrees**) but with less samples and one less attribute, e.g.,



This suggests a **recursive** approach to build decision trees.

Decision Tree Learning (DTL) Algorithm

Aim: Find the smallest tree consistent with the training samples.
Idea: Recursively choose "most significant" attribute as root of (sub)tree.

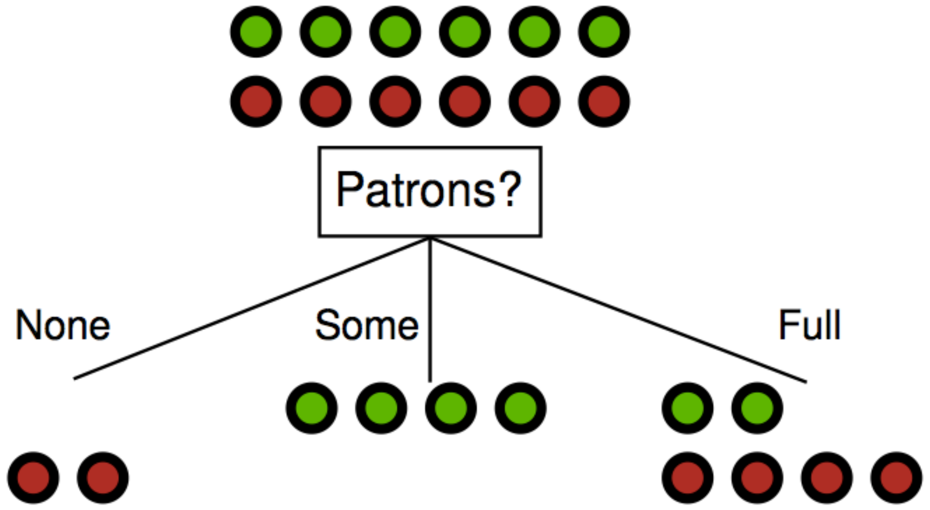


```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes,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

Decision Tree Learning (DTL) Algorithm (cont.)

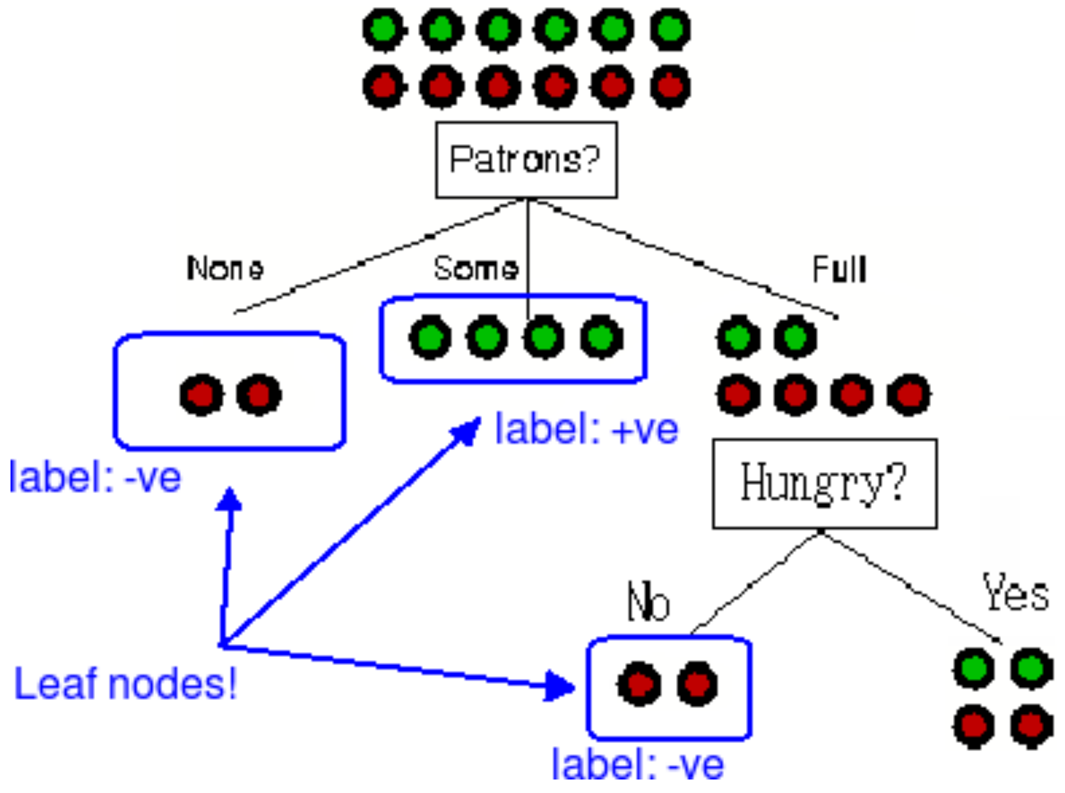
Four basic underlying ideas of the algorithm:

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



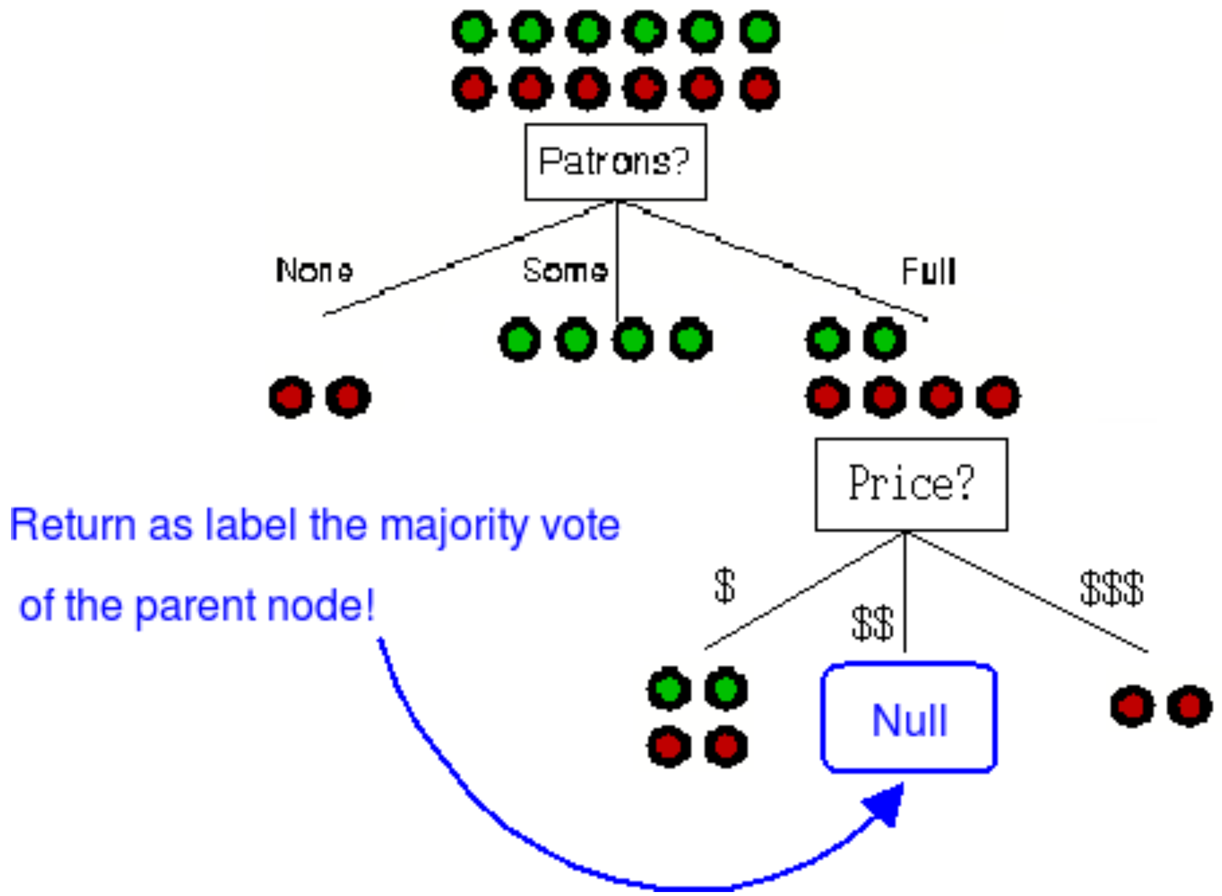
Decision Tree Learning (DTL) Algorithm (cont.)

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.,



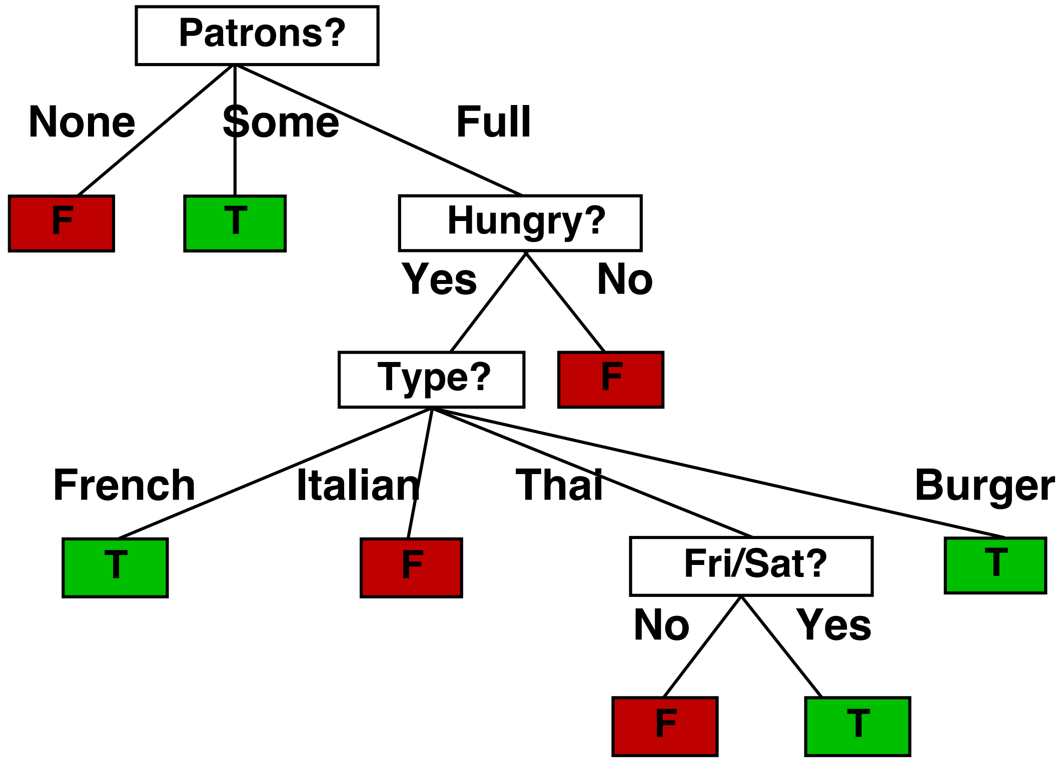
Decision Tree Learning (DTL) Algorithm (cont.)

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 (cont.)

- Result of applying DTL on the observations:



- Simpler than the tree shown in Page 2.

Choosing the Best Attribute

We need a measure of "good" and "bad" for attributes.

One way to do us is to compute the **information content** at a node, i.e. at node R

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

where $\{c_1, \ldots, c_L\}$ 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".

Choosing the Best Attribute (cont.)

Example:

At the root node of the restaurant problem, $c_1 = True$, $c_2 = False$ and there are 6 $True$ samples and 6 $False$ samples.

Therefore

$$\begin{aligned}
P(c_1) &= \frac{\text{no. of samples} = c_1}{\text{total no. of samples}} = \frac{6}{6+6} = 0.5 \\
P(c_2) &= \frac{\text{no. of samples} = c_2}{\text{total no. of samples}} = \frac{6}{6+6} = 0.5 \\
I(Root) &= -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1 \; bit
\end{aligned}$$

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).

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

Choosing the Best Attribute (cont.)

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

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 $c_1 = True$ with 2 samples and $c_2 = False$ with 4 samples, therefore

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

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

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

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 and the new information content, i.e.

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

where $\{E_{A=v_1}, \ldots, E_{A=v_M}\}$ 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

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

With similar calculations,

$$Gain(Type) = 0 \qquad \text{(Try this yourself!!!)}$$

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

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

Concluding Remarks

A different point of view is to regard $I(node)$ 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 $Gain(A)$ 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**.