Decision tree learning

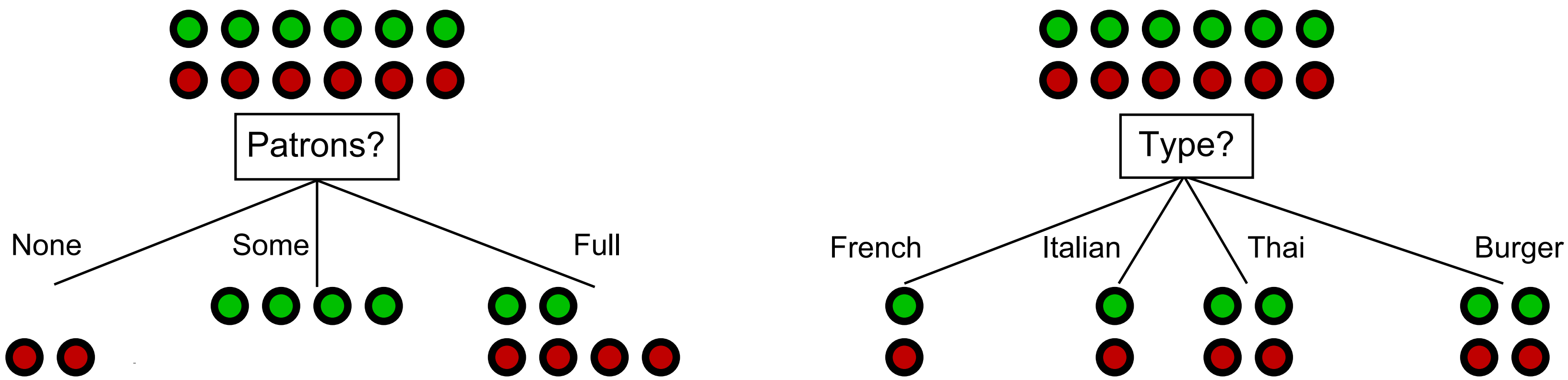
Aim: find a small tree consistent with the training examples

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
```

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Information Theory

- Consider communicating two messages (A and B) between two parties
- Bits are used to measure message size
- If $P(A) = 1$ and $P(B) = 0$, how many bits are needed?
- If $P(A) = .5$ and $P(B) = .5$, how many bits are needed?

Information Theory

◇ Consider communicating two messages (A and B) between two parties

◇ Bits are used to measure message size

◇ If $P(A) = 1$ and $P(B) = 0$, how many bits are needed?

◇ If $P(A) = .5$ and $P(B) = .5$, how many bits are needed?

◇ Information: $I(P(v_1), ...P(v_n)) = \Sigma_{i=1}^{n} -P(v_i) \log_2 P(v_i)$

◇ $I(1, 0) = 0$ bit

◇ $I(0.5, 0.5) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$ bit

Information Theory

◇ Consider communicating two messages (A and B) between two parties

◇ Bits are used to measure message size

◇ If $P(A) = 1$ and $P(B) = 0$, how many bits are needed?

◇ If $P(A) = .5$ and $P(B) = .5$, how many bits are needed?

◇ Information: $I(P(v_1), ...P(v_n)) = \Sigma_{i=1}^{n} -P(v_i) \log_2 P(v_i)$

◇ $I(1, 0) = 0$ bit

◇ $I(0.5, 0.5) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$ bit

◇ I measures the information content for communication (or uncertainty in what is already known)

◇ The more one knows, the less to be communicated, the smaller is I

◇ The less one knows, the more to be communicated, the larger is I

Using Information Theory

- $(P(pos), P(neg))$: probabilities of positive and negative
- Attribute $color$: black (1,0), white (0,1)
- Attribute $size$: large (.5,.5), small (.5,.5)

Using Information Theory

◇ $(P(pos), P(neg))$: probabilities of positive and negative

◇ Attribute $color$: black (1,0), white (0,1)

◇ Attribute $size$: large (.5,.5), small (.5,.5)

◇ Before selecting an attribute

- p = number of positive examples, n = number of negative examples
- Estimating probabilities: $P(pos) = \frac{p}{p+n}$, $P(neg) = \frac{n}{p+n}$
- $Before() = I(P(pos), P(neg))$

Selecting an Attribute

◊ Evaluating an attribute (e.g., color)

- p_i = number of positive examples for value i (e.g., black), n_i = number of negative ones
- Estimating probabilities for value i: $P_i(pos) = \frac{p_i}{p_i+n_i}$, $P_i(neg) = \frac{n_i}{p_i+n_i}$
- v values for attribute A (e.g., 2 for color)
- $Remainder(A) = After(A) = \Sigma_{i=1}^{v} \frac{p_i+n_i}{p+n} I(P_i(pos), P_i(neg))$ [expected information]

Selecting an Attribute

◇ Evaluating an attribute (e.g., color)

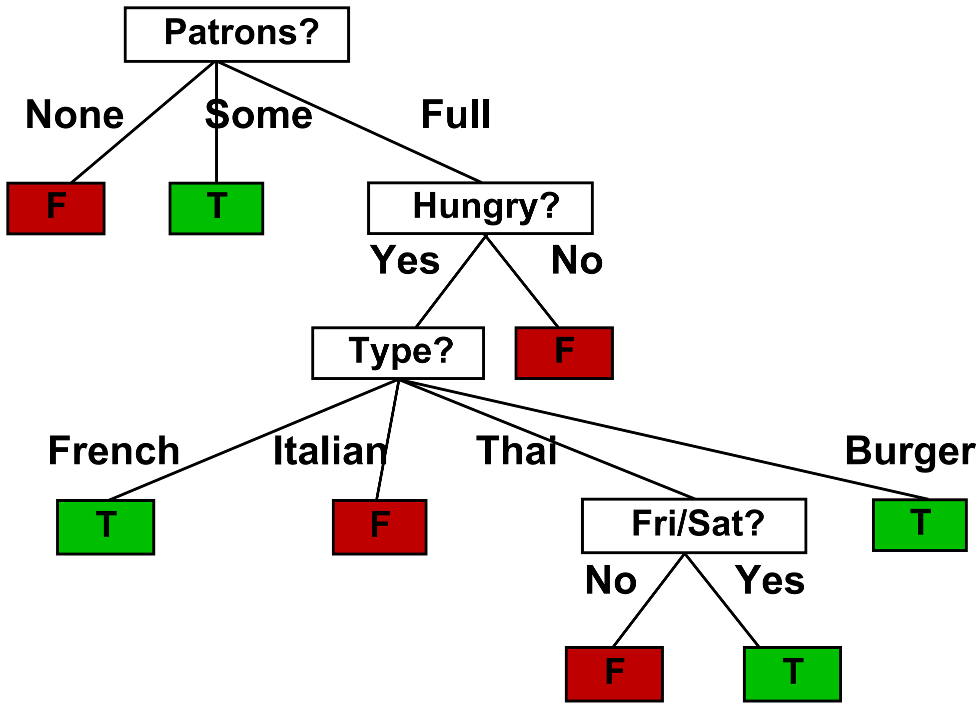
- p_i = number of positive examples for value i (e.g., black), n_i = number of negative ones
- Estimating probabilities for value i: $P_i(pos) = \frac{p_i}{p_i+n_i}$, $P_i(neg) = \frac{n_i}{p_i+n_i}$
- v values for attribute A (e.g., 2 for color)
- $Remainder(A) = After(A) = \Sigma_{i=1}^{v} \frac{p_i+n_i}{p+n} I(P_i(pos), P_i(neg))$ [expected information]

◇ "Information Gain" (reduction in uncertainty of what is known)

- $Gain(A) = Before() - After(A)$ [$Before()$ has more uncertainty]
- Choose attribute A with the largest $Gain(A)$

Example contd.

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Performance measurement

How do we know that $h \approx f$?

How about measuring the accuracy of h on the examples that were used to learn h?

Performance measurement

How do we know that $h \approx f$? (Hume's **Problem of Induction**)

1. Use theorems of computational/statistical learning theory
2. Try h on a new test set of examples
   - use **same distribution over example space** as training set
   - divide into two disjoint subsets: training and test sets
   - prediction accuracy: accuracy on the (unseen) test set

Performance measurement

Learning curve = % correct on test set as a function of training set size