Learning Agents

Chapter 18

Supervised Learning and Decision Trees

Jim Rehg
Georgia Tech

Decision Trees

Decision Trees

Decision trees are a type of classifier which use combinations of simple rules to make predictions

They are widely-used in practice due to their simplicity and interpretability

We illustrate the basic idea through an example…

Restaurant Example

- Classify: when I will vs. I won’t wait for a table at a restaurant
- Features include:
  - Is there an alternate in mind, Is it Friday when most restaurants are full, Does the restaurant have a nice bar for waiting, Am I really hungry, Is it raining, Type of restaurant, Price, Estimated wait time
  - Ex: on Friday I might be willing to wait since most places will be full, especially if it’s raining and annoying to go back outside
  - Ex: I’m willing to wait 20 mins for tex-mex, but only 10 for pizza

Attribute-based Representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*

Attribute-based Representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

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*

Example has a value for every attribute

Attribute-based Representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

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*

And the target f(x) classification value

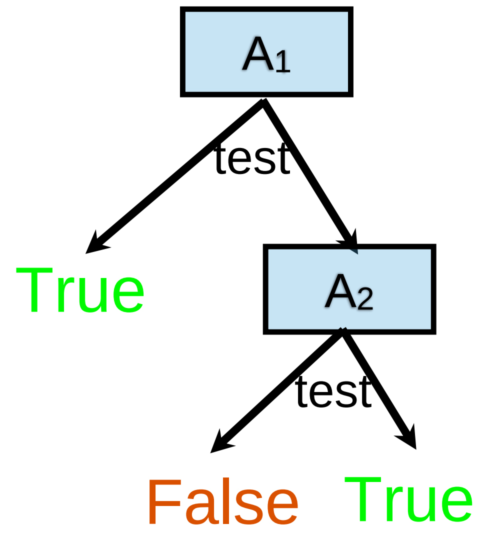
Attribute-based Representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

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

A set of examples like this is your Training Set

Decision Tree



Representation of hypothesis $h(x)$, which maps

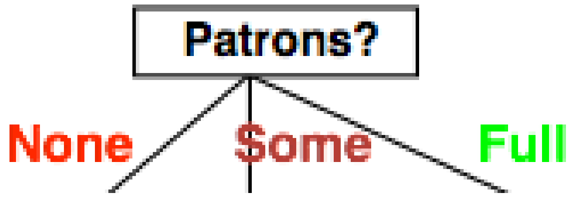
- input vector of attributes $x = \{A_1,\ldots,A_m\}$,
- where attribute A_i has d_i possible values,
- to a single "decision" output y

Implemented as a tree of tests applied to the attributes, with branches corresponding to attribute values

Output $h(x)$ is determined by the leaf nodes

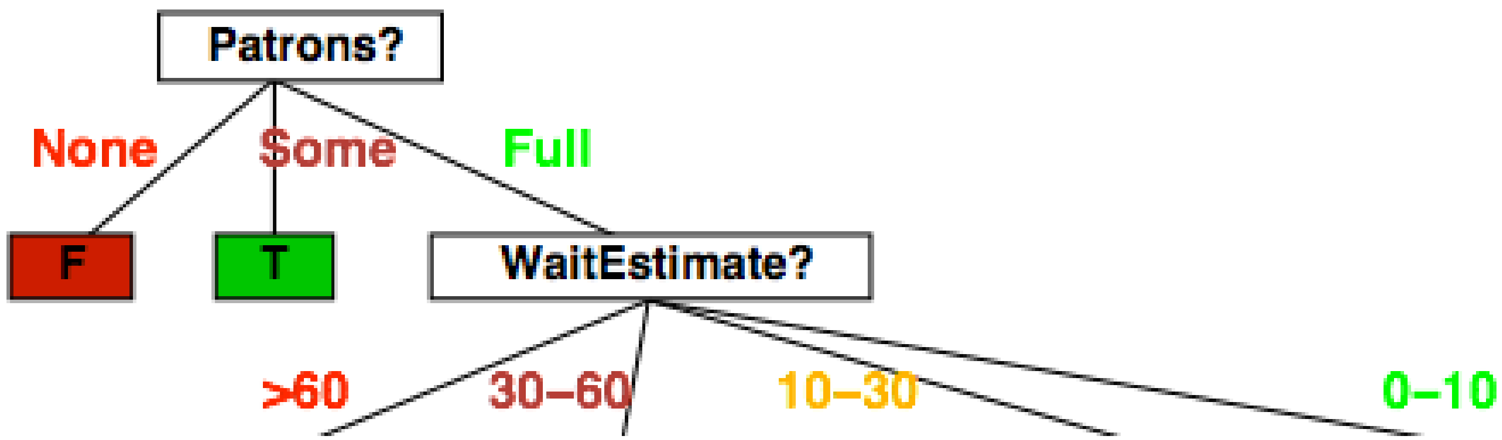
Decision Trees

One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



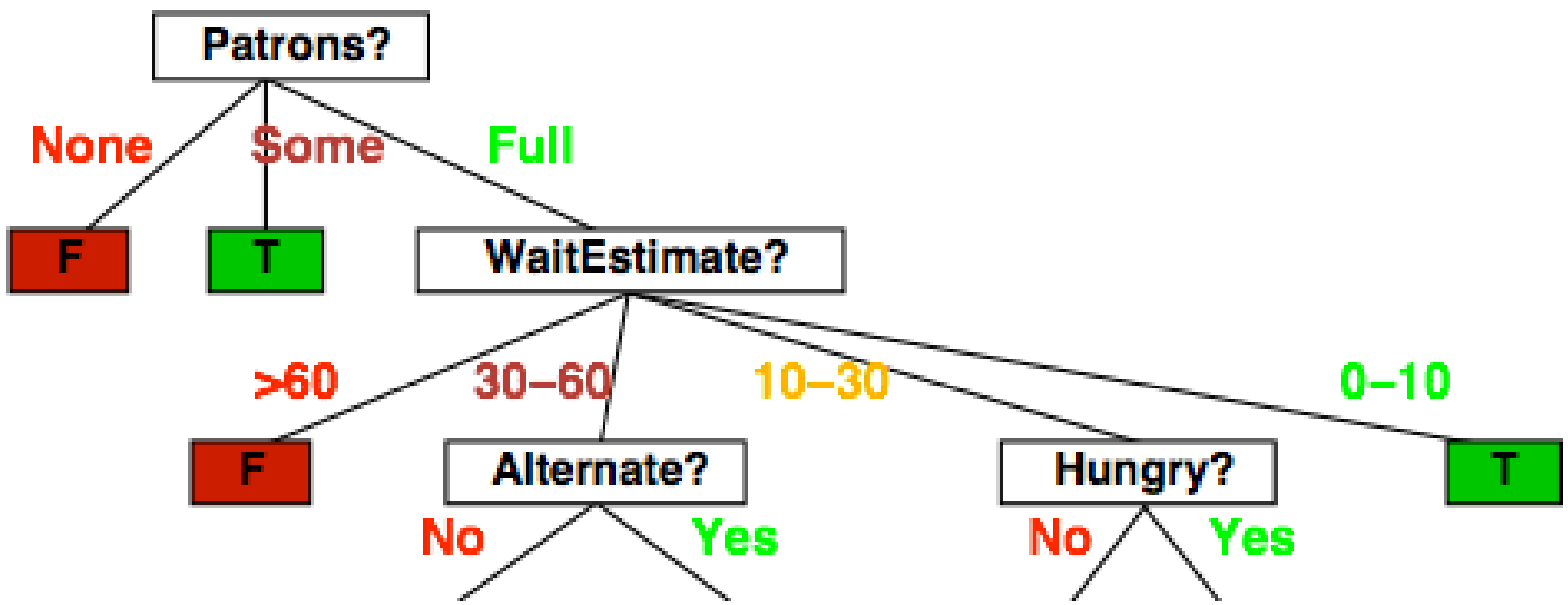
Decision Trees

One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



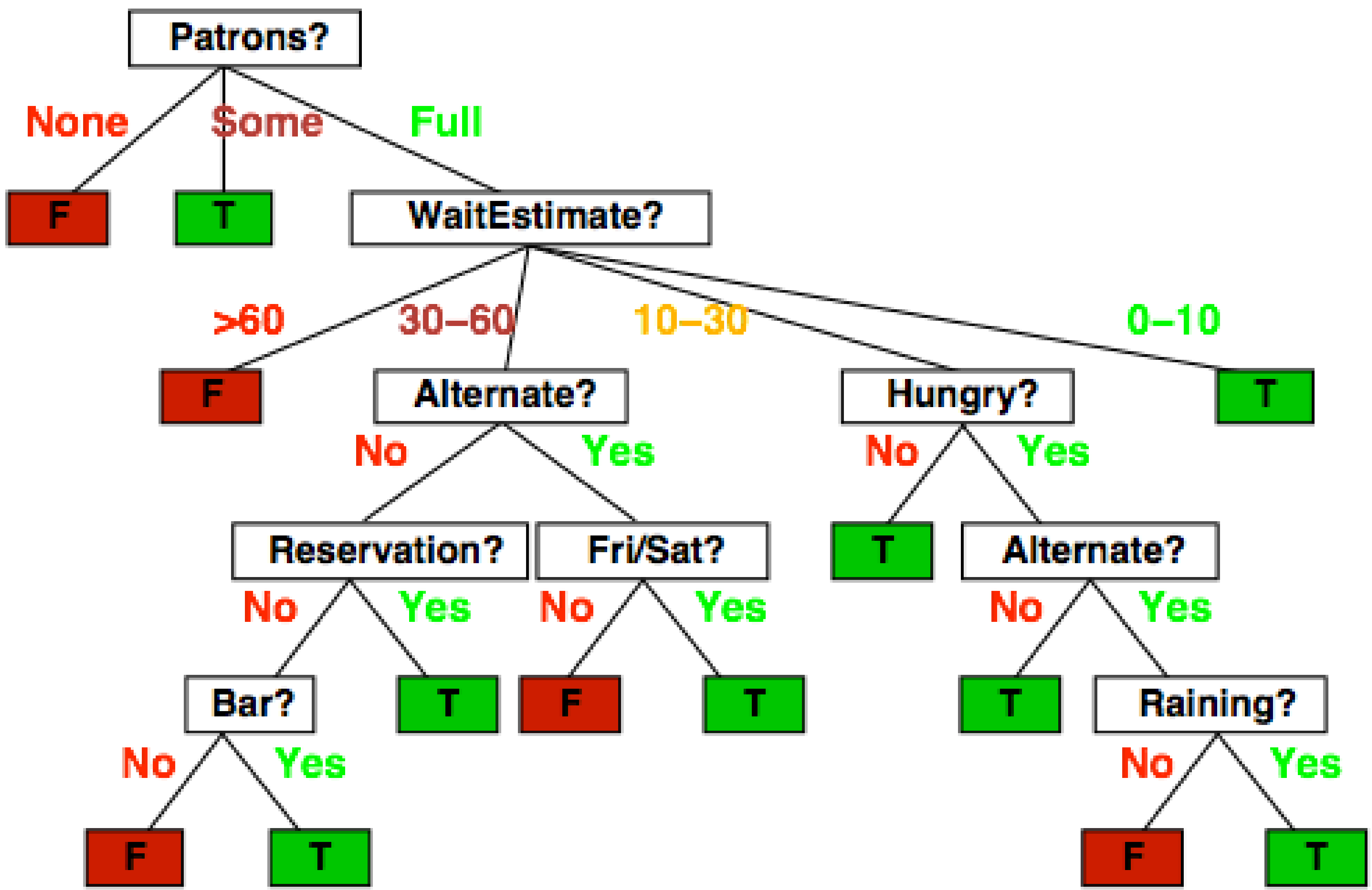
Decision Trees

One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



Decision Trees

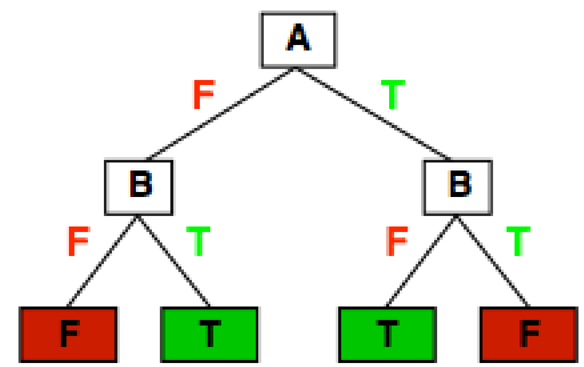
One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row $\longrightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Non-Boolean (discrete) attributes can always be made Boolean (by increasing # of attributes)

Thus space of decision trees is equivalent to space of Boolean functions

How many decision trees for N binary attributes?

Expressiveness

How many decision trees for N binary attributes?

N binary attributes
N-bit vector

Each Boolean function (i.e. decision tree) assigns 0/1 to each attribution combination (i.e. row)

2^N rows

A_1	A_2	…	A_N	h(x)
0	0	…	0	0
0	0	…	1	1
…	…	…	…	…
1	1	…	1	0

Bit vector of length 2^N

$\Rightarrow 2^{2^N}$ possible trees

For 20 attributes there are over $10^{300,000}$ trees!

Decision Tree Learning

Basic decision tree induction:

- Choose a leaf node
- Choose the best attribute to split on
- Split by adding branches to form new leaves
- Partition the node dataset into the leaves
- Repeat until convergence

Issues:

- *How to choose the best attribute to split on?*
- *How to decide when to stop?*

Decision Tree Learning

Basic decision tree induction:

- Choose a leaf node
- Choose the best attribute to split on
- Split by adding branches to form new leaves
- Partition the node dataset into the leaves
- Repeat until convergence

Issues:

- *How to choose the best attribute to split on?*
- *How to decide when to stop?*

Attribute-based Representations

Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

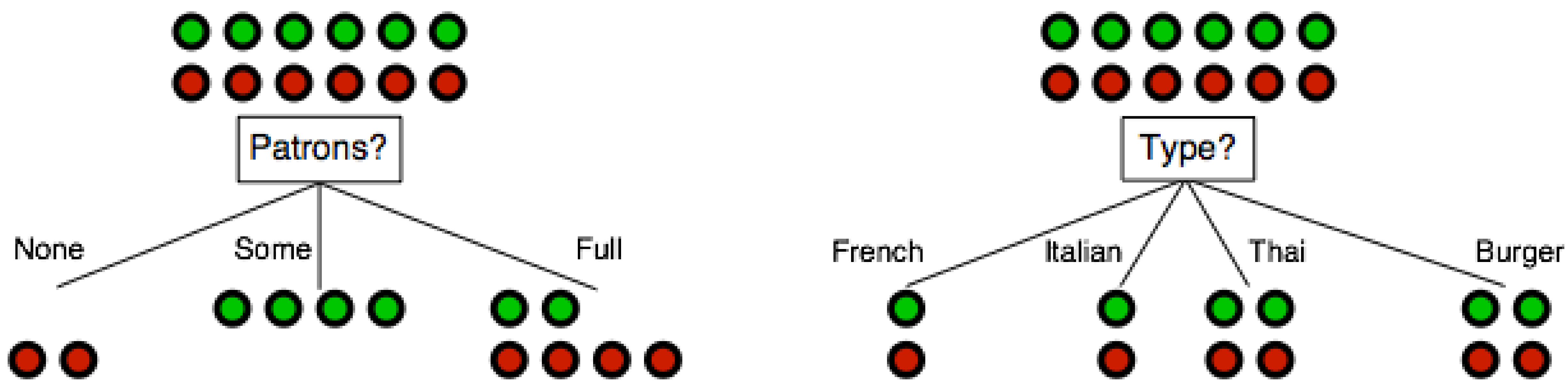
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	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

We are going to be passing sets of training examples through the Decision Tree tests

Denote each example by its target value (red/green)

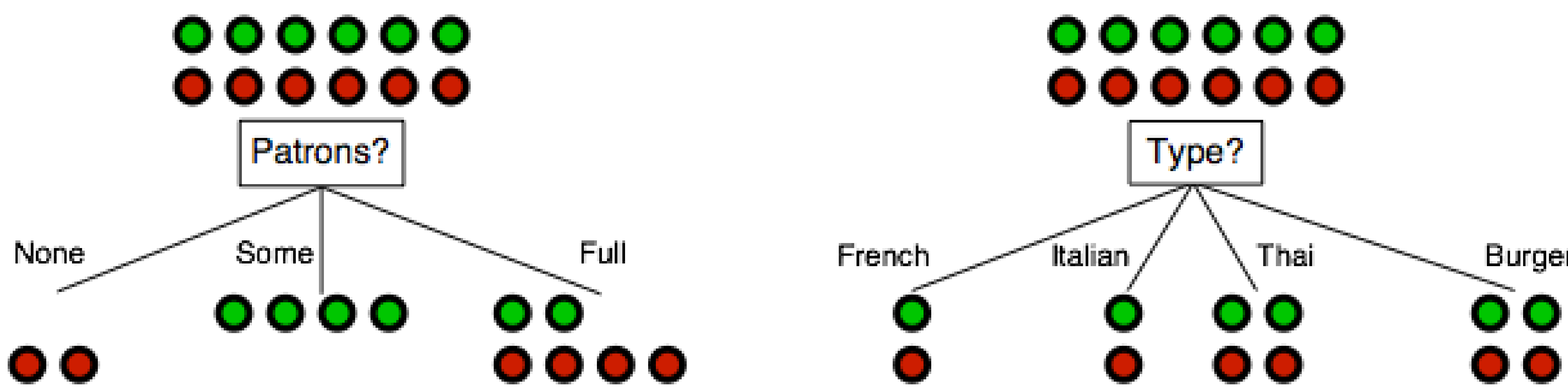
Choosing an Attribute

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



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

Note that leaves which are homogeneous (all data points have the same output) can’t be split further

Decision Tree Learning

Basic decision tree induction:

- Choose a leaf node
- Choose the best attribute to split on
- Split by adding branches to form new leaves
- Partition the node dataset into the leaves
- Repeat until convergence

Issues:

- *How to choose the best attribute to split on?*
- *How to decide when to stop?*

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
```

Stopping Criteria

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
    be
    tr
    fo
      add a branch to *tree* with label v_i and subtree *subtree*
    return *tree*

No examples left to split on:
Return the decision of the parent node

Stopping Criteria

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*
fo

re

Remaining examples are homogeneous:
Return the label of the examples

Stopping Criteria

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
        fo
        re
```

No attributes left to split the data:

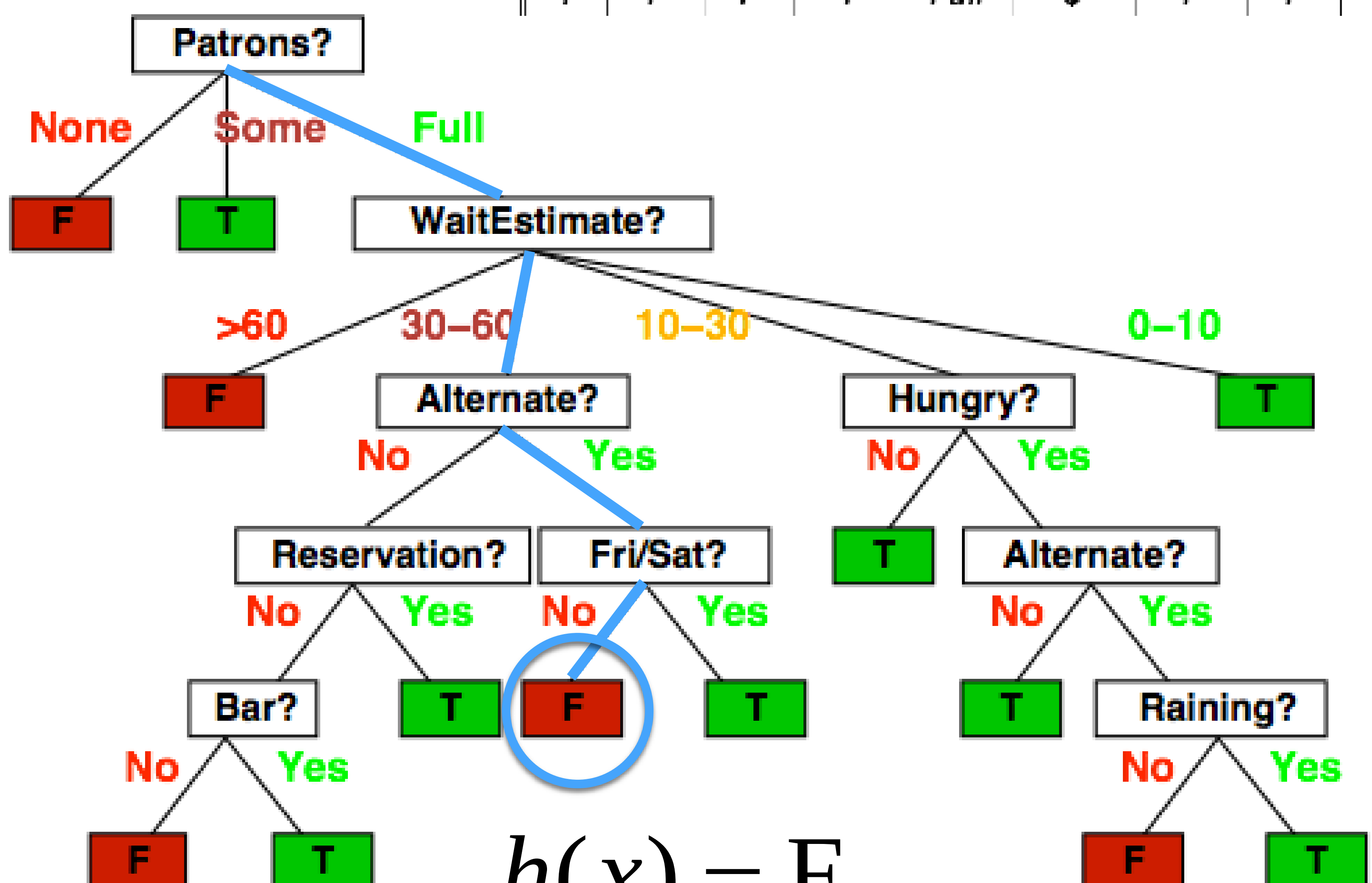
Return the most probable label

Output Predictions During Testing

How to predict y for an input pattern x?

Input x =

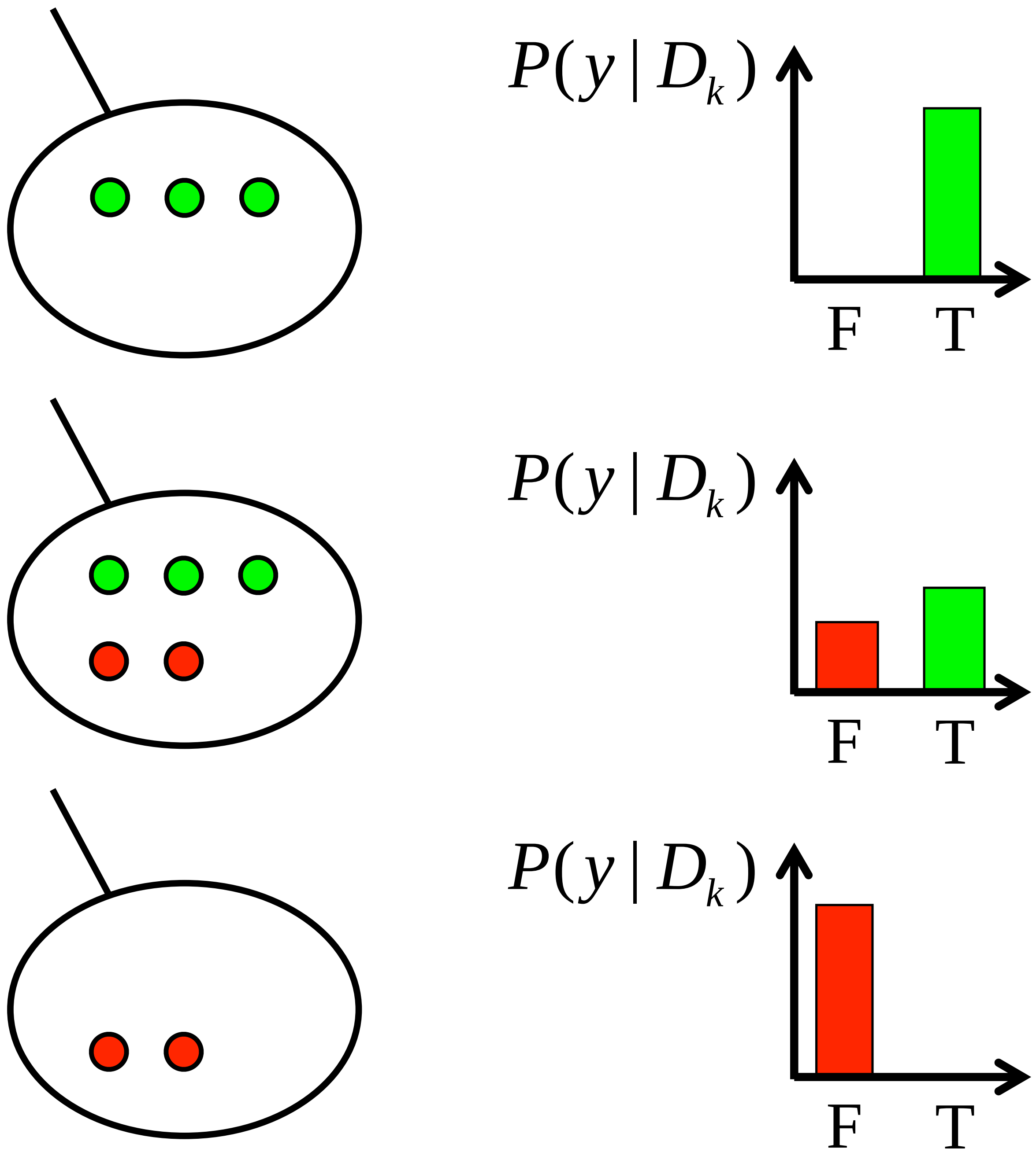
Attributes									
Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*
T	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*



$$h(x) = \mathrm{F}$$

Making Predictions at a Leaf

Example Leaves | *Empirical Distributions* | *Example Outputs*



T

T

F

Let D_k be dataset at leaf k

Prediction Formulas

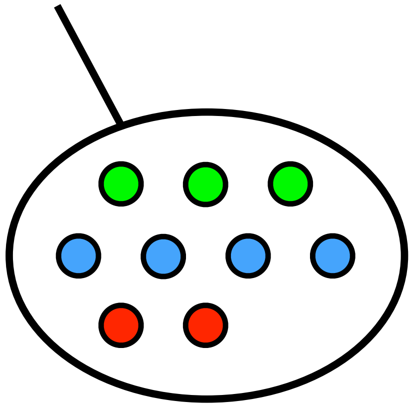
In general, for categorical y with M values:

$$P(y=m \mid D_k) = \frac{\#\,\text{examples } y=m \text{ in } D_k}{\text{total examples in } D_k} = \frac{C_k^m}{N_k} \qquad \text{for } 1 \le m \le M$$

For example:

$M = 3$

Leaf k



$C_k^1 = 3 \quad P(y=1) = 1/3$

$C_k^2 = 4 \quad P(y=2) = 4/9$

$C_K^3 = 2 \quad P(y=3) = 2/9$

In binary case ($M = 2$):

$$P(y=1 \mid D_k) = \frac{p_k}{p_k + n_k}$$

p_k = #positive examples in D_k

n_k = #negative examples in D_k

$p_k + n_k = N_k$

Quality of Predictions

We need a way to measure how likely a given leaf is to make useful predictions

Use this measure to compare distributions before and after a split, to select attributes

Entropy of the distribution is a useful measure:

$$H(P) = -\sum_{m=1}^{M} P(y=m) \log_2 P(y=m)$$

Entropy is a scalar (single number) measuring the uncertainty in the distribution

Binary Entropy

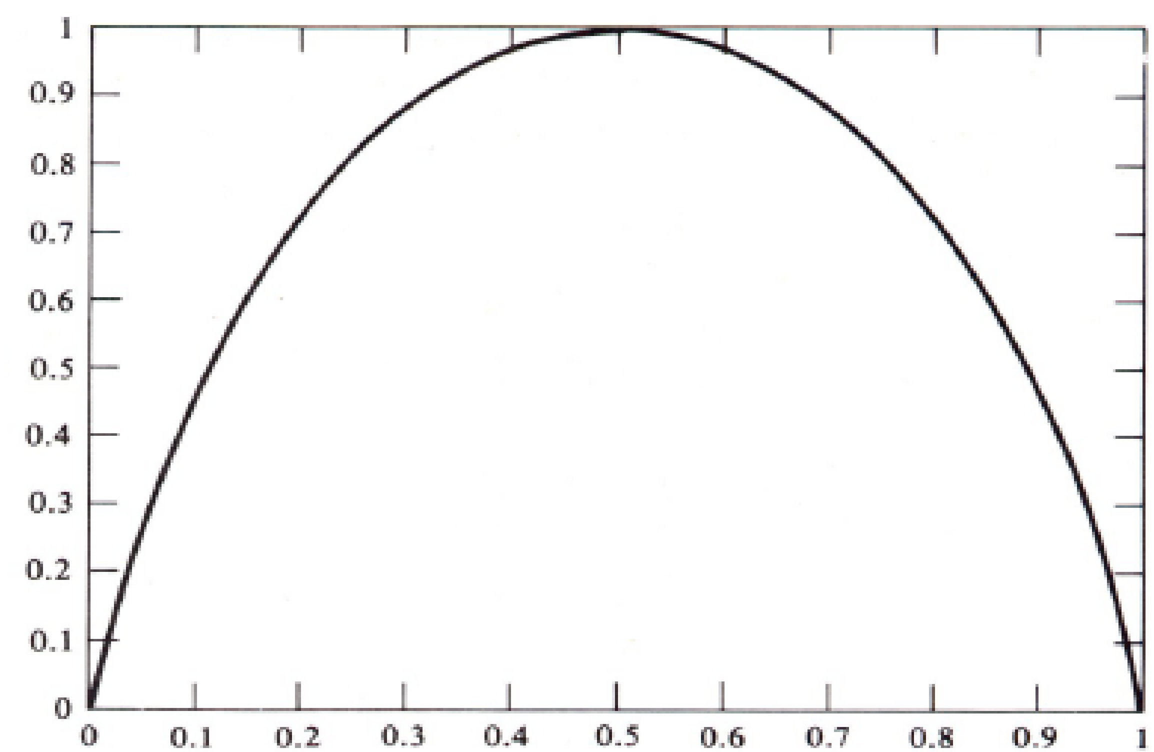
For Bernoulli distributions (binary variables) the entropy is given by:

$$H(P) = B(p_1) = -p_1 \log_2 p_1 - (1 - p_1)\log_2(1 - p_1)$$

where $p_1 = P(y=1)$

$0 \le B(p_1) \le 1$

$B(p_1)$



p_1

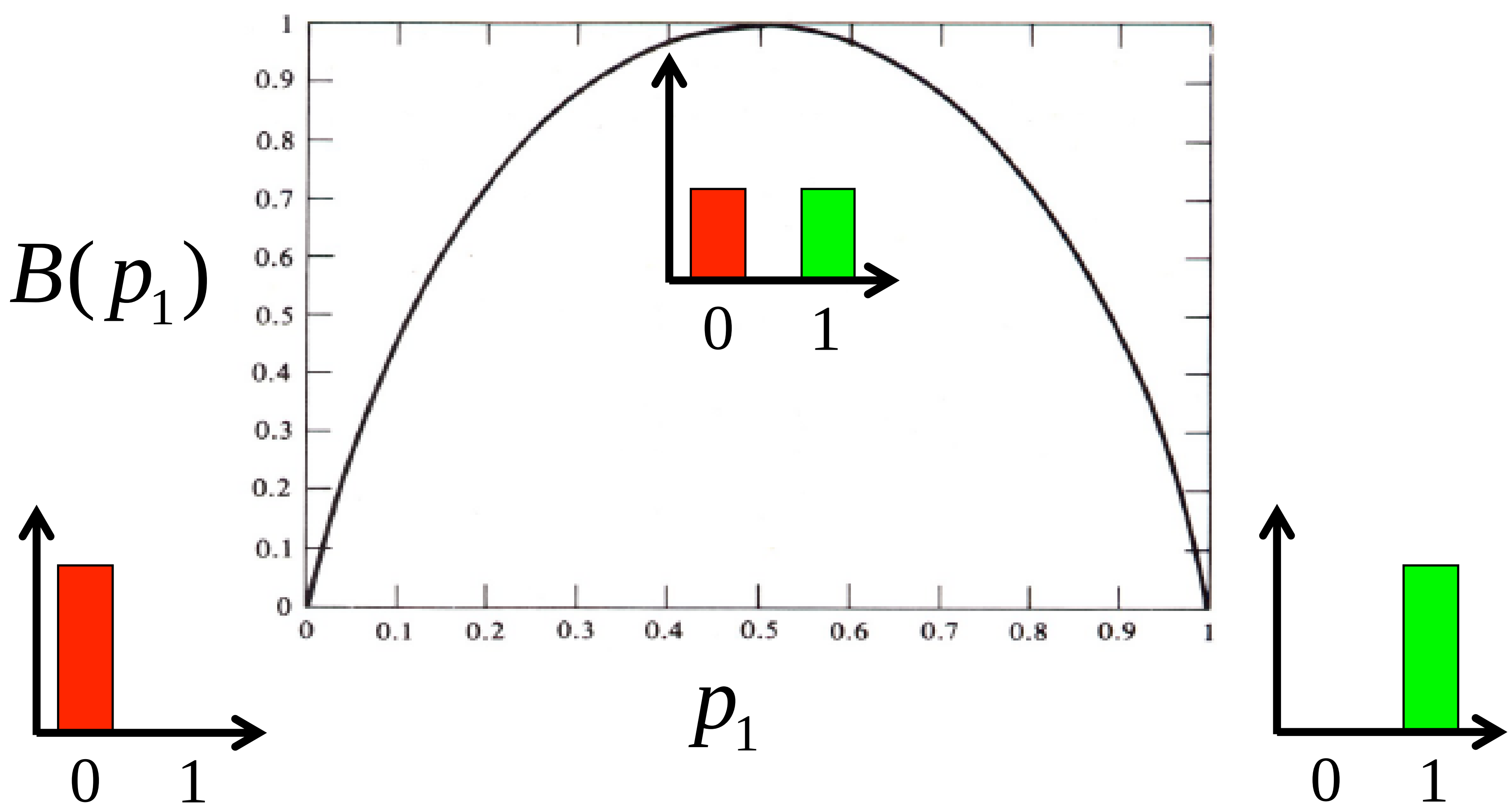
Binary Entropy

For Bernoulli distributions (binary variables) the entropy is given by:

$$H(P) = B(p_1) = -p_1 \log_2 p_1 - (1 - p_1)\log_2(1 - p_1)$$

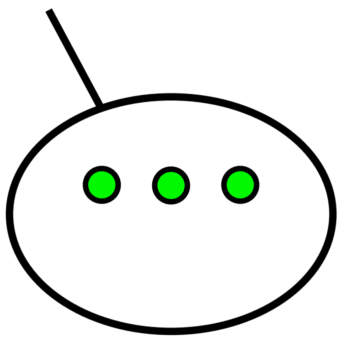
where $p_1 = P(y = 1)$

$0 \le B(p_1) \le 1$

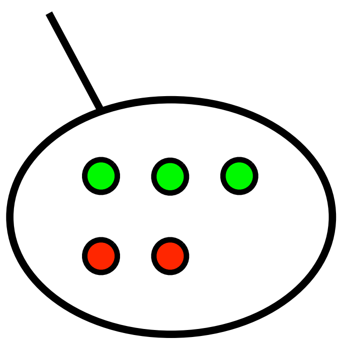


Using Entropy

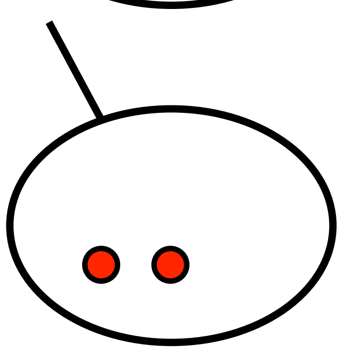
The lower the entropy, the more certainty in predicting an output label



$$B(3/3) = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = 0$$



$$B(3/5) = -3/5 \cdot \log_2 3/5 - 2/5 \cdot \log_2 2/5 = 0.971$$

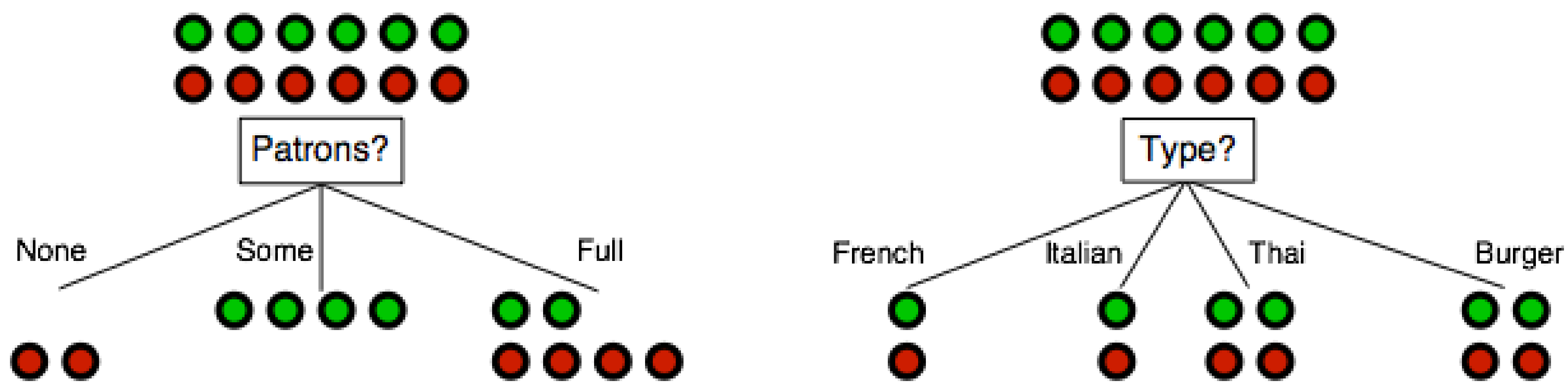


$$B(0/2) = -0 \cdot \log_2 0 - 1 \cdot \log_2 1 = 0$$

Strategy: *Select attributes to minimize entropy*

Change in Entropy

Select the attribute which produces the greatest decrease in entropy after the split



How can we measure the effect of observing an attribute A_i on the entropy $P(y)$?

Before the split: $H(P) = 1$
After the split: $H(P|\, A_i) = ?$

Conditional Entropy

Conditional Entropy $H(y|x)$

Measures entropy that remains in y after x is observed

$$H(y \mid x) = -\sum_x \sum_y P(x,y) \log_2 P(y \mid x)$$

$$= -\sum_x \sum_y P(y \mid x) P(x) \log_2 P(y \mid x)$$

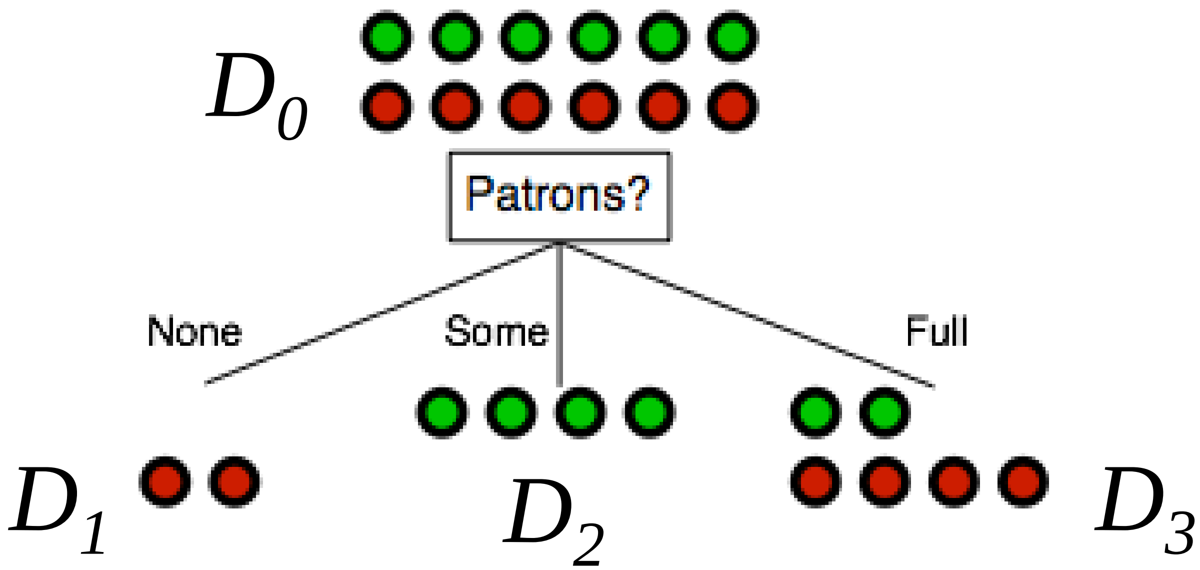
$$= \sum_x \boxed{P(x)} \left\{ \boxed{-\sum_y P(y \mid x) \log_2 P(y \mid x)} \right\}$$

The entropy for each value of x is weighted by its probability

This is the entropy of the distribution $P(y|x)$

Mutual Information

Split on attribute A_i (e.g. “patrons?”):

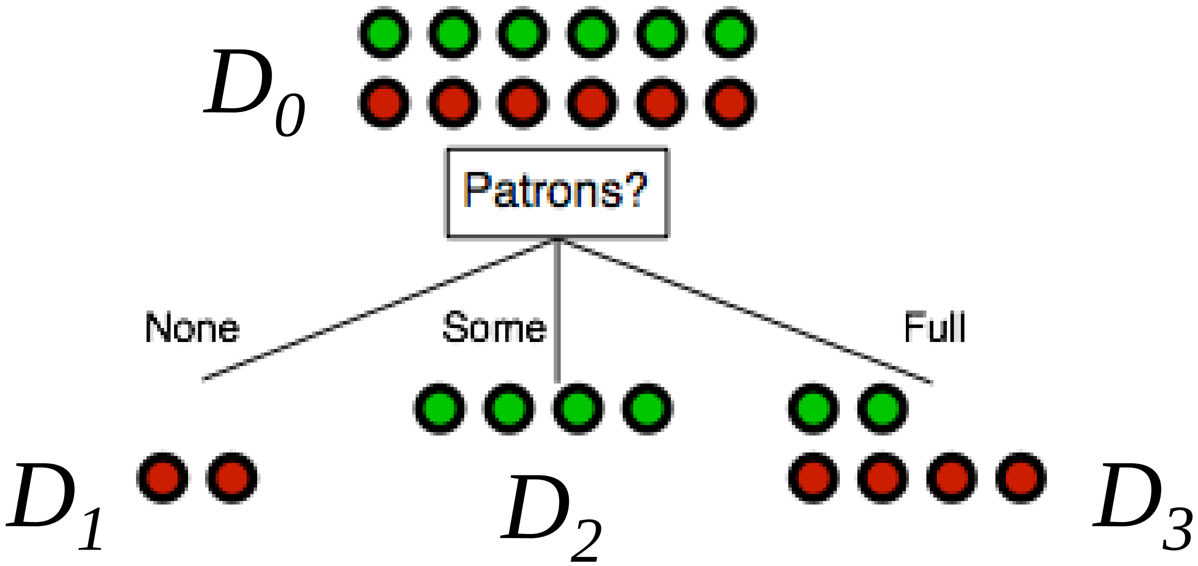


Before the split:

$$H(P(y \mid D_0)) = B(0.5) = 1$$

Mutual Information

Split on attribute A_i (e.g. “patrons?”):



Before the split:

$$H(P(y \mid D_0)) = B(0.5) = 1$$

After the split:

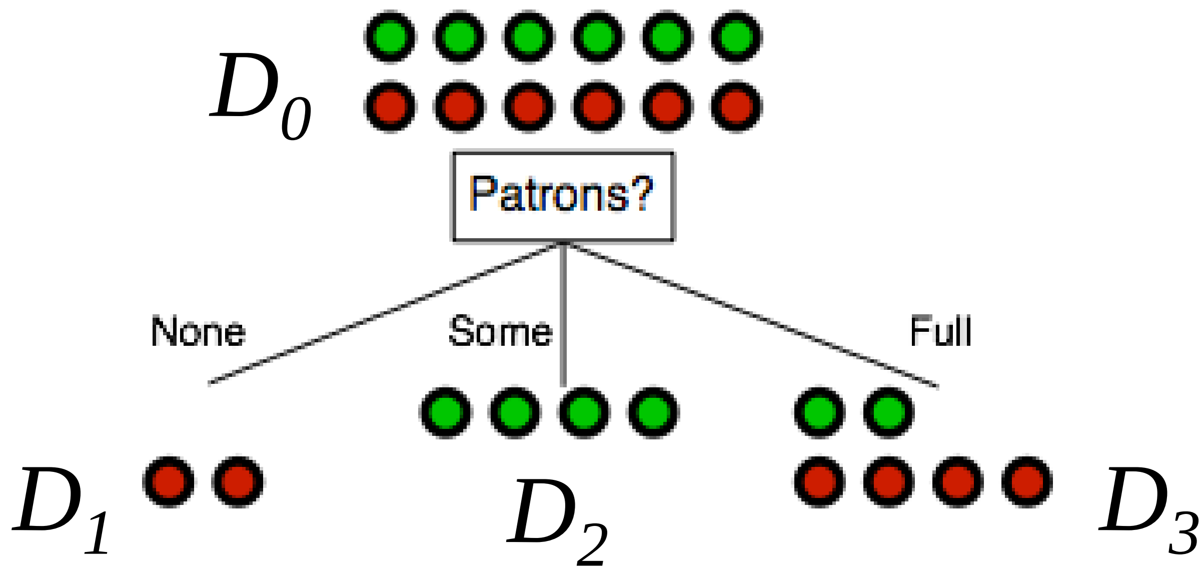
$$P(A_i = 1) = \tfrac{1}{6} \quad P(A_i = 2) = \tfrac{1}{3} \quad P(A_i = 3) = \tfrac{1}{2}$$

$$H(y \mid A_i) = \sum_{j=1}^{d_i} P(A_i = j) B(P(y = 1 \mid D_j))$$

$$= \tfrac{1}{6} B(0) + \tfrac{1}{3} B(1) + \tfrac{1}{2} B(\tfrac{2}{6}) = 0.459$$

Mutual Information

Split on attribute A_i (e.g. "patrons?"):



Before the split:

$$H(P(y \mid D_0)) = B(0.5) = 1$$

After the split:

$$I(y, A_i) = H(y) - H(y \mid A_i)$$
$$= 1 - 0.459 = 0.541$$

Note: $I(y, A_i) \geq 0$

The *mutual information* $I(y, A_i)$ measures the reduction in uncertainty about y that comes from measuring A_i