# Learning Decision Trees

## CS194-10 Fall 2011 Lecture 8

# Outline

- ♢ Decision tree models
- ♢ Tree construction
- ♢ Tree pruning
- ♢ Continuous input features

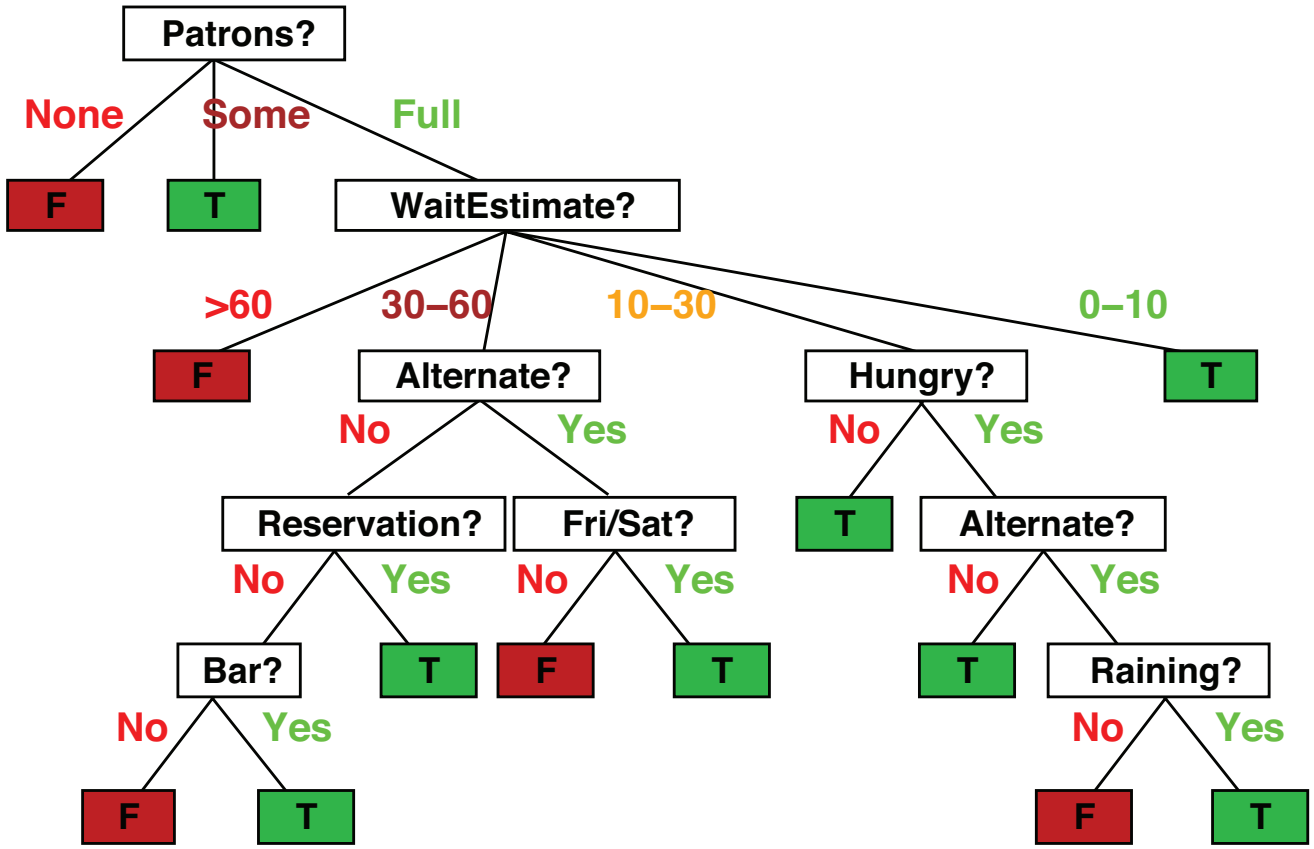
# Restaurant example

Discrete inputs (some have static discretizations), Boolean output

| Example | Input Attributes | | | | | | | | | | Goal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *WillWait* |
| $\mathbf{x}_1$ | *Yes* | *No* | *No* | *Yes* | *Some* | *$$$* | *No* | *Yes* | *French* | *0–10* | $y_1 =$ *Yes* |
| $\mathbf{x}_2$ | *Yes* | *No* | *No* | *Yes* | *Full* | *$* | *No* | *No* | *Thai* | *30–60* | $y_2 =$ *No* |
| $\mathbf{x}_3$ | *No* | *Yes* | *No* | *No* | *Some* | *$* | *No* | *No* | *Burger* | *0–10* | $y_3 =$ *Yes* |
| $\mathbf{x}_4$ | *Yes* | *No* | *Yes* | *Yes* | *Full* | *$* | *Yes* | *No* | *Thai* | *10–30* | $y_4 =$ *Yes* |
| $\mathbf{x}_5$ | *Yes* | *No* | *Yes* | *No* | *Full* | *$$$* | *No* | *Yes* | *French* | *>60* | $y_5 =$ *No* |
| $\mathbf{x}_6$ | *No* | *Yes* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Italian* | *0–10* | $y_6 =$ *Yes* |
| $\mathbf{x}_7$ | *No* | *Yes* | *No* | *No* | *None* | *$* | *Yes* | *No* | *Burger* | *0–10* | $y_7 =$ *No* |
| $\mathbf{x}_8$ | *No* | *No* | *No* | *Yes* | *Some* | *$$* | *Yes* | *Yes* | *Thai* | *0–10* | $y_8 =$ *Yes* |
| $\mathbf{x}_9$ | *No* | *Yes* | *Yes* | *No* | *Full* | *$* | *Yes* | *No* | *Burger* | *>60* | $y_9 =$ *No* |
| $\mathbf{x}_{10}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$$$* | *No* | *Yes* | *Italian* | *10–30* | $y_{10} =$ *No* |
| $\mathbf{x}_{11}$ | *No* | *No* | *No* | *No* | *None* | *$* | *No* | *No* | *Thai* | *0–10* | $y_{11} =$ *No* |
| $\mathbf{x}_{12}$ | *Yes* | *Yes* | *Yes* | *Yes* | *Full* | *$* | *No* | *No* | *Burger* | *30–60* | $y_{12} =$ *Yes* |

# Decision trees

Popular representation for hypotheses, even among humans!
E.g., here is the "true" tree for deciding whether to wait:



# Classification and regression

Each path from root to a leaf defines a region $R_m$ of input space

Let $\mathbf{X}_m$ be the training examples that fall into $R_m$

Classification tree:

– discrete output

– leaf value $\hat{y}_m$ typically set to the most common value in $\mathbf{X}_m$

Regression tree:

– continuous output

– leaf value $\hat{y}_m$ typically set to the mean value in $\mathbf{X}_m$ ($L_2$)

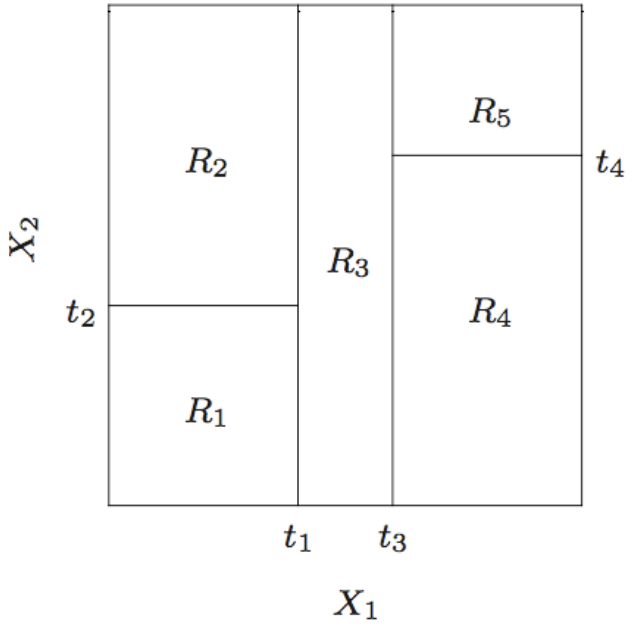
# Discrete and continuous inputs

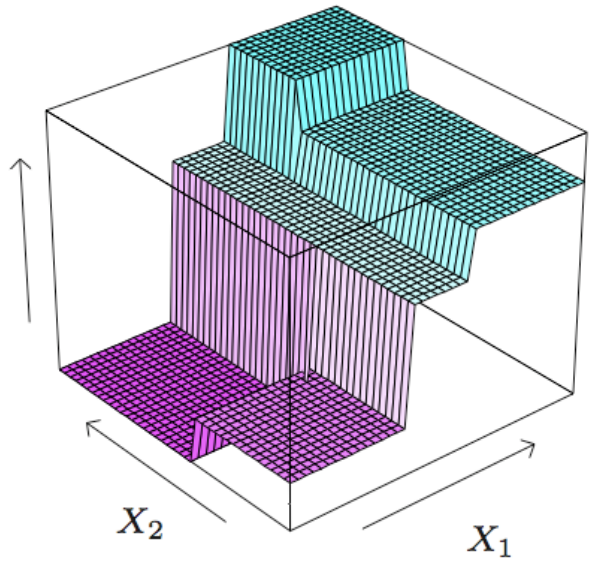
Simplest case: discrete inputs with small ranges (e.g., Boolean)

$\Rightarrow$ one branch for each value; attribute is "used up" ("complete split")

For continuous attribute, test is $X_j > c$ for some split point $c$

$\Rightarrow$ two branches, attribute may be split further in each subtree




Also split large discrete ranges into two or more subsets

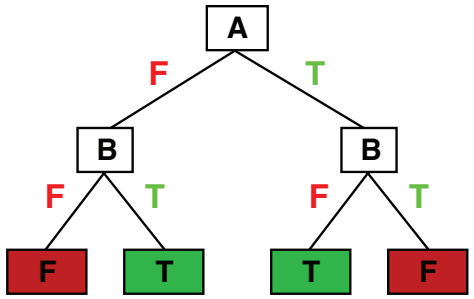
# Expressiveness

Discrete-input, discrete-output case:

– Decision trees can express any function of the input attributes.

– E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |




Continuous-input, continuous-output case:

– Can approximate any function arbitrarily closely

Trivially, there is a consistent decision tree for any training set
w/ one path to leaf for each example (unless $f$ nondeterministic in $x$)
but it probably won't generalize to new examples

Need some kind of regularization to ensure more **compact** decision trees

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows = $2^{2^n}$

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows = $2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows = $2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??

# Digression: Hypothesis spaces

How many distinct decision trees with $n$ Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with $2^n$ rows = $2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??

Each attribute can be in (positive), in (negative), or out
$\Rightarrow$ $3^n$ distinct conjunctive hypotheses

# Digression: Hypothesis spaces

More expressive hypothesis space

- increases chance that target function can be expressed
- increases number of hypotheses consistent w/ training set
  - $\Rightarrow$ many consistent hypotheses have large test error
  - $\Rightarrow$ may get worse predictions

$2^{2^n}$ hypotheses $\Rightarrow$ all but an exponentially small fraction will require $O(2^n)$ bits to describe in **any** representation (decision trees, SVMs, English, brains)

Conversely, any given hypothesis representation can provide compact descriptions for only an exponentially small fraction of functions

E.g., decision trees are bad for "additive threshold"-type functions such as "at least 6 of the 10 inputs must be true," but linear separators are good

# Decision tree learning

Aim: find a small tree consistent with the training examples

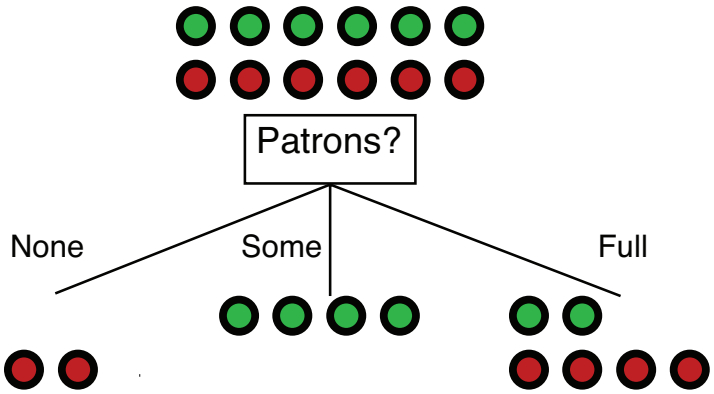
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

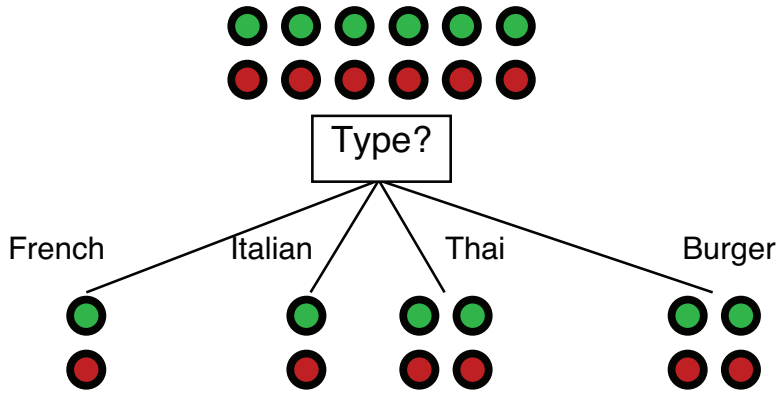
```
function DECISION-TREE-LEARNING(examples, attributes, parent_examples)
returns a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
      A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
      tree ← a new decision tree with root test A
      for each value v_k of A do
          exs ← {e : e ∈ examples and e.A = v_k}
          subtree ← DECISION-TREE-LEARNING(exs, attributes − A, examples)
          add a branch to tree with label (A = v_k) and subtree subtree
      return tree
```

# Choosing an attribute: Minimizing loss

Idea: if we can test only one more attribute, which one results in the smallest empirical loss on examples that come through this branch?



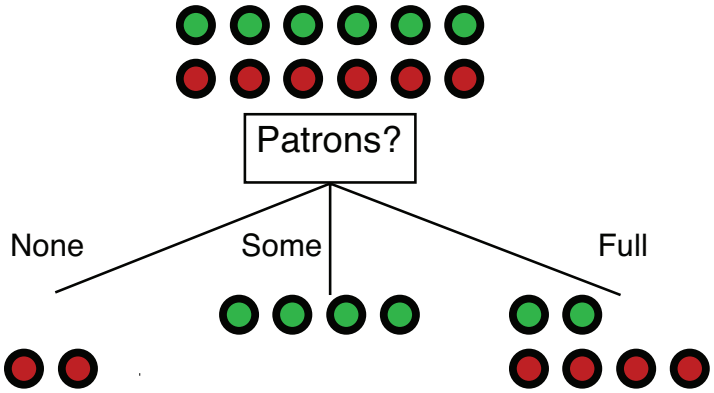



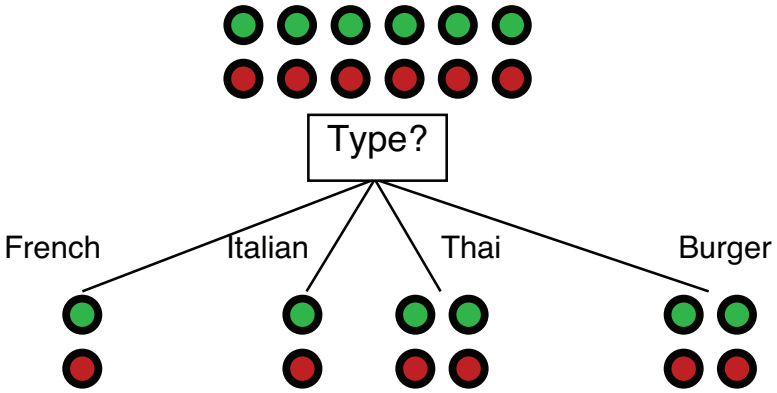

Assuming we pick the majority value in each new leaf, 0/1 loss is $0 + 0 + 2 = 2$ for $Patrons$ and $1 + 1 + 2 + 2 = 6$ for $Type$

For continuous output, assume least-squares fit (mean value) in each leaf, pick the attribute that minimizes total squared error across new leaves

# Choosing an attribute: Information gain

Idea: measure the contribution the attribute makes to increasing the "purity" of the $Y$-values in each subset of examples




*Patrons* is a better choice—gives information about the classification (also known as reducing entropy of distribution of output values)

# Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)

Convenient notation: $B(p) = H(\langle p, 1-p \rangle)$

# Information contd.

Suppose we have $p$ positive and $n$ negative examples at the root

$\Rightarrow$ $B(p/(p+n))$ bits needed to classify a new example

E.g., for 12 restaurant examples, $p=n=6$ so we need 1 bit

An attribute splits the examples $E$ into subsets $E_k$, each of which (we hope) needs less information to complete the classification

Let $E_k$ have $p_k$ positive and $n_k$ negative examples

$\Rightarrow$ $B(p_k/(p_k+n_k))$ bits needed to classify a new example

$\Rightarrow$ **expected** number of bits per example over all branches is

$$\sum_i \frac{p_i+n_i}{p+n} B(p_k/(p_k+n_k))$$

For $Patrons$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

# Empirical loss vs. information gain

Information gain tends to be less greedy:

– Suppose $X_j$ splits a 10/6 parent into 2/0 and 8/6 leaves

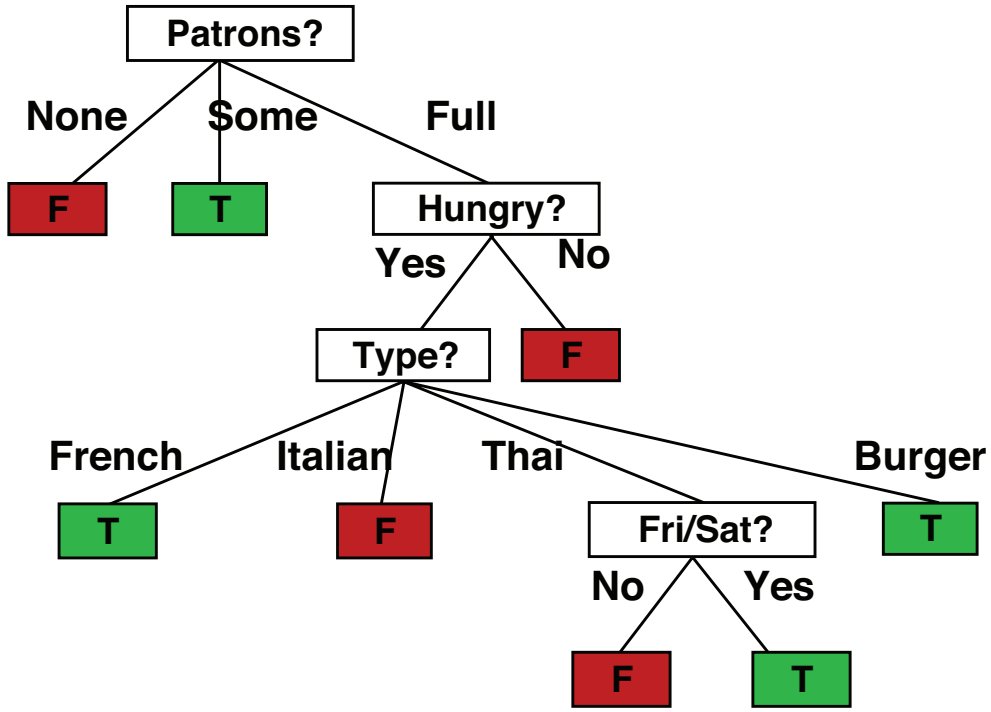
– 0/1 loss is 6 for parent, 6 for children: "no progress"

– 0/1 loss is still 6 with 4/0 and 6/6 leaves! – Information gain is 0.2044 bits: we have "peeled off" an easy category

Most experimental investigations indicate that information gain leads to better results (smaller trees, better predictive accuracy)

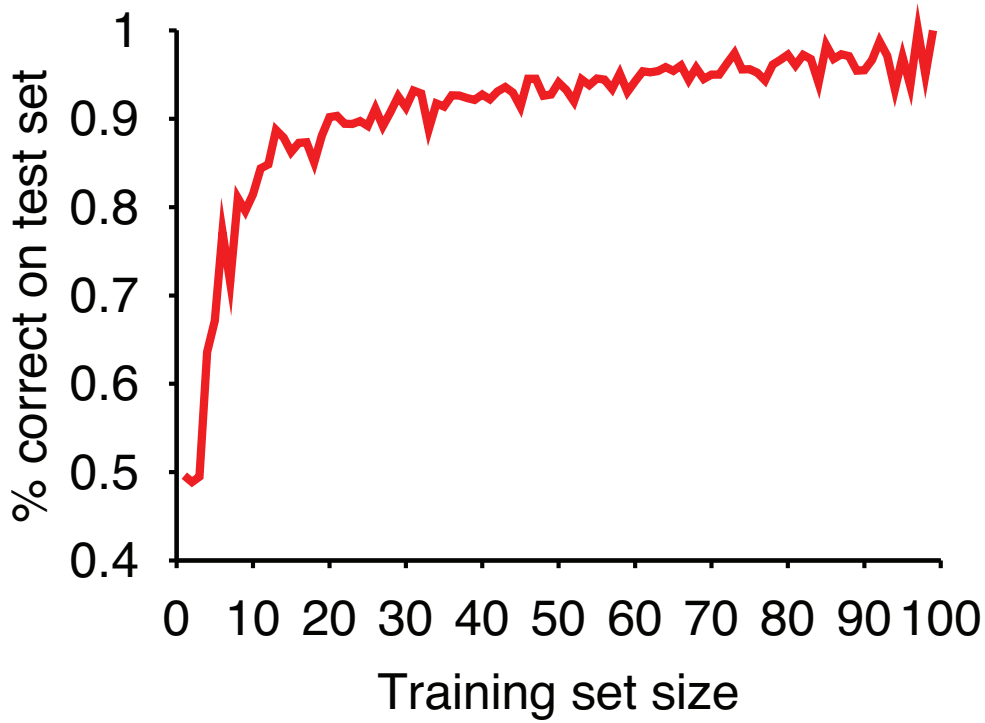
# Results on restaurant data

Decision tree learned from the 12 examples:




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

# Learning curve for restaurant data



# Pruning

Tree grows until it fits the data perfectly or runs out of attributes

In the non-realizable case, will severely overfit

Pruning methods trim tree back, removing "weak" tests

A test is weak if its ability to separate +ve and -ve is not
significantly different from that of a completely irrelevant attribute

E.g., $X_j$ splits a 4/8 parent into 2/2 and 2/6 leaves; gain is 0.04411 bits.
How likely is it that randomly assigning the examples to leaves of the same size leads to a gain of at least this amount?

Test the deviation from 0-gain split using $\chi^2$-squared statistic,
prune leaves of nodes that fail the test, repeat until done.
Regularization parameter is the $\chi^2$ threshold (10%, 5%, 1%, etc.)

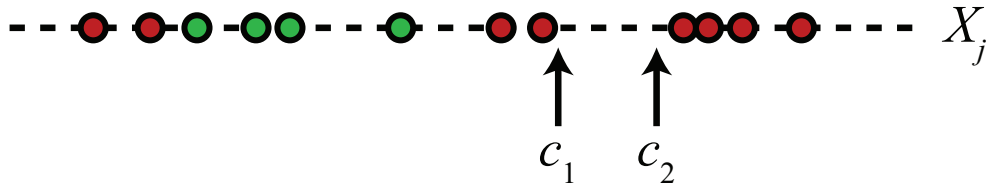
# Optimal splits for continuous attributes

Infinitely many possible split points $c$ to define node test $X_j > c$ ?

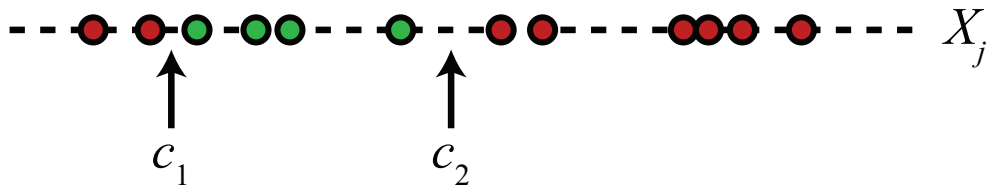
# Optimal splits for continuous attributes

Infinitely many possible split points $c$ to define node test $X_j > c$ ?

No! Moving split point along the empty space between two observed values has no effect on information gain or empirical loss; so just use midpoint

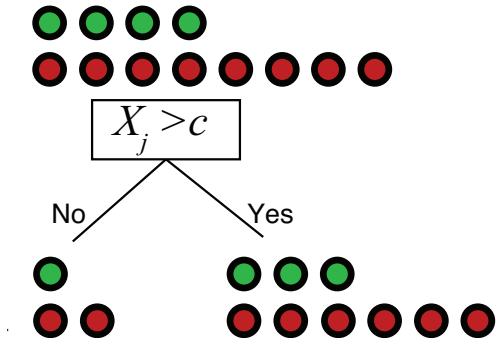



Moreover, only splits **between examples from different classes** can be optimal for information gain or empirical loss reduction



# Optimal splits contd.

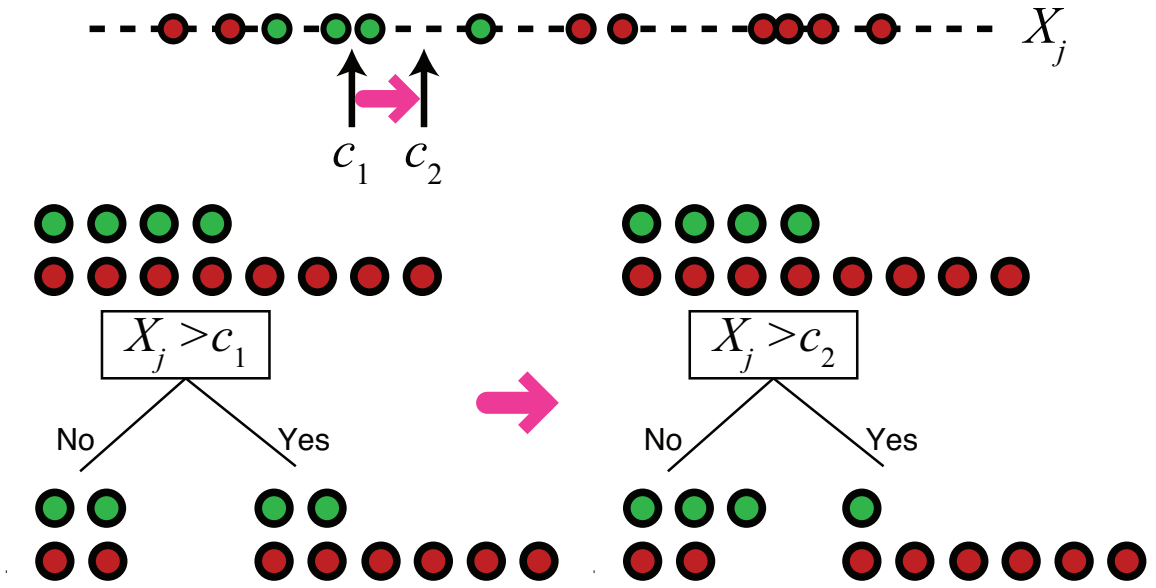
Lemma: Information gain is minimized (zero) when each child has the same output label distribution as the parent



# Optimal splits contd.

Lemma: if $c$ splits $X_j$ between two +ve examples, and
examples to the left of $c$ are **more** +ve than the parent distribution, then

- examples to the right are are **more** -ve than the parent distribution,
- moving $c$ right by one takes children **further** from parent distribution,
- and information gain necessarily increases




Gain for $c_1$: $B(\frac{4}{12}) - (\frac{4}{12}B(\frac{2}{4}) + \frac{8}{12}B(\frac{2}{8})) = 0.04411$ bits

Gain for $c_2$: $B(\frac{4}{12}) - (\frac{5}{12}B(\frac{3}{5}) + \frac{7}{12}B(\frac{1}{7})) = 0.16859$ bits

# Optimal splits for discrete attributes

A discrete attribute may have many possible values
- e.g., $ResidenceState$ has 50 values

Intuitively, this is likely to result in overfitting;
extreme case is an identifier such as SS#.

Solutions:
- don't use such attributes
- use them and hope for the best
- group values by hand (e.g., $NorthEast$, $South$, $MidWest$, etc.)
- find best split into two subsets–but there are $2^K$ subsets!

# Optimal splits for discrete attributes contd.

The following algorithm gives an optimal binary split:

Let $p_{jk}$ be proportion of +ve examples
in the set of examples that have $X_j = x_{jk}$

Order the values $x_{jk}$ by $p_{jk}$

Choose the best split point just as for continuous attributes

E.g., $p_{j,CA} = 7/10$, $p_{j,NY} = 4/10$, $p_{j,TX} = 5/20$, $p_{j,IL} = 3/20$;
ascending order is $IL$, $TX$, $NY$, $CA$; 3 possible split points.

# Additional tricks

Blended selection criterion: info gain near root, empirical loss near leaves

"Oblique splits": node test is a linear inequality $\mathbf{w}^T\mathbf{x} + b > 0$

Full regressions at leaves: $\hat{\mathbf{y}}(\mathbf{x}) = (\mathbf{X}_\ell^T\mathbf{X}_\ell)^{-1}\mathbf{X}_\ell^T\mathbf{Y}_\ell$

Cost-sensitive classification

Lopsided outputs: one class very rare (e.g., direct mail targeting)

Scaling up: disk-resident data

# Summary

"Of all the well-known learning methods, decision trees come closest to meeting the requirements for serving as an off-the-shelf procedure for data mining." [Hastie et al.]

Efficient learning algorithm

Handle both discrete and continuous inputs and outputs

Robust against any monotonic input transformation, also against outliers

Automatically ignore irrelevant features: no need for feature selection

Decision trees are usually interpretable