Will I run into traffic on my way home from work?
Why trees?

- interpretable/intuitive, popular in medical applications because they mimic the way people like to reason
- model discrete outcomes nicely
- powerful, nonlinear, can be as complex as you need them
- C4.5 and CART – both in “top 10” from 2008

(Quinlan, 1993)  (Breiman et al., 1984)

Some real examples (from Russell & Norvig, Mitchell)
- BP's GasOIL system for separating gas and oil on offshore platforms: C4.5 replaced a hand-designed rule system with 2500 rules. It saved BP millions. (1986)
- Learning to fly a Cessna on a flight simulator by watching human experts fly the simulator (1992)
Will I run into traffic on my way home from work?
Will I run into traffic on my way home from work?

Split and prune: Is this a good way to build a tree?

Well, at least it’s fast…
Will I run into traffic on my way home from work?

Split and prune: Is this a good way to build a tree?

Well, at least it’s fast…
Example (adapted from Russell & Norvig):

Will the customer wait for a table at a restaurant?

• OthOptions: Other options, True if there are restaurants nearby.
• Weekend: This is true if it is Friday, Saturday or Sunday.
• WaitArea: Does it have a bar or other nice waiting area to wait in?
• Plans: Does the customer have plans just after dinner?
• Price: This is either $, $$, $$$, or $$$$
• Precip: Is it raining or snowing?
• Restaur:
  • Mateo (fancy),
  • Juju (nice),
  • Blue Corn Mexican Café (casual),
  • Pompieri Pizza (very casual)
• Wait: Wait time estimate: 0-5 min, 6-15 min, 16-30 min, or 30+
• Crowded: Whether there are other customers (no, some, or full)
Dataset

<table>
<thead>
<tr>
<th></th>
<th>OthOptions</th>
<th>Weekend</th>
<th>WaitArea</th>
<th>Plans</th>
<th>Price</th>
<th>Precip</th>
<th>Restaur</th>
<th>Wait</th>
<th>Crowded</th>
<th>Stay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>$$$</td>
<td>No</td>
<td>Mateo</td>
<td>0-5</td>
<td>some</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>$</td>
<td>No</td>
<td>Juju</td>
<td>16-30</td>
<td>full</td>
<td>No</td>
</tr>
<tr>
<td>$x_3$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>$</td>
<td>No</td>
<td>Pizza</td>
<td>0-5</td>
<td>some</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>$</td>
<td>No</td>
<td>Juju</td>
<td>6-15</td>
<td>full</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>$$$</td>
<td>No</td>
<td>Mateo</td>
<td>30+</td>
<td>full</td>
<td>No</td>
</tr>
<tr>
<td>$x_6$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>$$</td>
<td>Yes</td>
<td>BlueCorn</td>
<td>0-5</td>
<td>some</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_7$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>$</td>
<td>Yes</td>
<td>Pizza</td>
<td>0-5</td>
<td>none</td>
<td>No</td>
</tr>
<tr>
<td>$x_8$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>$$</td>
<td>Yes</td>
<td>Juju</td>
<td>0-5</td>
<td>some</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_9$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>$</td>
<td>Yes</td>
<td>Pizza</td>
<td>30+</td>
<td>full</td>
<td>No</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>$$$</td>
<td>No</td>
<td>BlueCorn</td>
<td>6-15</td>
<td>full</td>
<td>No</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>$</td>
<td>No</td>
<td>Juju</td>
<td>0-5</td>
<td>none</td>
<td>No</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>$</td>
<td>No</td>
<td>Pizza</td>
<td>16-30</td>
<td>full</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Will the customer wait for a table at a restaurant?

Crowded?
- None: No
- Some: Mateo
  - Yes: BlueCorn
  - No: Pizza
     - No: Juju
     - Yes: Weekend?
       - No: Yes
       - Yes: Yes
Greedy tree induction:

• Start at the top of the tree.
• Grow it by “splitting” features one by one. To split, look at how “impure” the node is.
• Assign leaf nodes the majority vote in the leaf.
• At the end, go back and prune leaves to reduce overfitting.
Which of these two features should we split on?
Which gives the most information about whether the customer will wait?
Which of these two features should we split on?
Which gives the most information about whether the customer will wait?

Crowded?

None
+ x₁,x₃,x₄,x₆,x₈,x₁₂
- x₂,x₅,x₇,x₉,x₁₀,x₁₁

Some
+ x₁,x₃,x₆,x₈
- x₇,x₁₁

Full
+ x₄,x₁₂
- x₂,x₅,x₉,x₁₀

Plans?

Yes plans
+ x₄,x₁₂
- x₂,x₁₀

No plans
+ x₁
- x₅

Restaur?

+ x₁,x₃,x₄,x₆,x₈,x₁₂
- x₂,x₅,x₇,x₉,x₁₀,x₁₁

Mateo
+ x₁
- x₅

Juju
+ x₄,x₈
- x₂,x₁₁

Pizza
+ x₃,x₁₂
- x₇,x₉

BlueCorn
+ x₆
- x₁₀
Splitting Criteria for Decision Trees: Information Gain

The training probabilities in branch $j$ are:

$$\left[ \frac{\#pos_j}{\#pos_j + \#neg_j}, \frac{\#neg_j}{\#pos_j + \#neg_j} \right]$$

$.01,.99$ is good
$.50,.50$ is bad
A Splitting Criteria for Decision Trees: Information Gain

\[
\text{Gain}(S, A) = \text{expected reduction in entropy due to branching on attribute } A
\]

\[
= H \left( \left[ \frac{\#\text{pos}}{\#\text{pos} + \#\text{neg}}, \frac{\#\text{neg}}{\#\text{pos} + \#\text{neg}} \right] \right)
\]

\[
- \sum_{j=1}^{J} \frac{\#\text{pos}_j + \#\text{neg}_j}{\#\text{pos} + \#\text{neg}} \cdot H \left[ \frac{\#\text{pos}_j}{\#\text{pos}_j + \#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j + \#\text{neg}_j} \right].
\]
Which of these two features should we split on?
Which gives the most information about whether the customer will wait?

Crowded?
- None
  - + x_1, x_3, x_4, x_6, x_8, x_12
  - - x_2, x_5, x_7, x_9, x_{10}, x_{11}

- Some
  - + x_1, x_3, x_6, x_8
  - -

- Full
  - + x_4, x_{12}
  - - x_2, x_5, x_9, x_{10}

Restaur?
- Mateo
  - + x_1, x_3, x_4, x_6, x_8, x_{12}
  - - x_2, x_5, x_7, x_9, x_{10}, x_{11}

- Juju
  - + x_4, x_8
  - - x_2, x_{11}

- Pizza
  - + x_3, x_{12}
  - - x_7, x_9

BlueCorn
- + x_6
  - - x_{10}
A Splitting Criteria for Decision Trees: Information Gain

\[
\text{Gain}(S, A) = \text{expected reduction in entropy due to branching on attribute } A \\
= \text{original entropy} - \text{entropy after branching} \\
= H\left(\left[\frac{\#\text{pos}}{\#\text{pos} + \#\text{neg}}, \frac{\#\text{neg}}{\#\text{pos} + \#\text{neg}}\right]\right) \\
- \sum_{j=1}^{J} \frac{\#\text{pos}_j + \#\text{neg}_j}{\#\text{pos} + \#\text{neg}} H\left[\frac{\#\text{pos}_j}{\#\text{pos}_j + \#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j + \#\text{neg}_j}\right].
\]

(entropy of whole dataset) \hspace{1cm} (entropy after splitting)

\[
\text{Gain}(S, \text{Crowded}) = H\left(\left[\frac{1}{2}, \frac{1}{2}\right]\right) - \left[\frac{2}{12} H([0, 1]) + \frac{4}{12} H([1, 0]) + \frac{6}{12} H\left(\left[\frac{2}{6}, \frac{4}{6}\right]\right)\right] \\
\approx 0.541 \text{ bits.}
\]
Splitting Criteria for Decision Trees: Information Gain

\[
\text{Gain}(S, \text{Restaur}) = 1 - \left[ \frac{2}{12} H \left( \left[ \frac{1}{2}, \frac{1}{2} \right] \right) + \frac{2}{12} H \left( \left[ \frac{1}{2}, \frac{1}{2} \right] \right) + \frac{4}{12} H \left( \left[ \frac{2}{4}, \frac{2}{4} \right] \right) + \frac{4}{12} H \left( \left[ \frac{2}{4}, \frac{2}{4} \right] \right) \right] \approx 0 \text{ bits.}
\]

\[
\text{Gain}(S, \text{Crowded}) = H \left( \left[ \frac{1}{2}, \frac{1}{2} \right] \right) - \left[ \frac{2}{12} H([0, 1]) + \frac{4}{12} H([1, 0]) + \frac{6}{12} H \left( \left[ \frac{2}{6}, \frac{4}{6} \right] \right) \right] \approx 0.541 \text{ bits.}
\]
So far...

- I’ve discussed Information Gain, which is the splitting criteria for C4.5.
- I still need to discuss:
  - other possible splitting criteria
  - pruning criteria
  - CART
  - Some uses for greedy tree-splitting algorithms
  - What happens if you don’t want to be greedy?

(Quinlan, 1993)
Decision Trees Part 2

Cynthia Rudin
Machine Learning Class
Duke University
Where we left off:

- Information gain as the splitting criteria for C4.5 (Quinlan, 1993)
- Chooses a split to reduce the entropy as much as possible
A Splitting Criterion for Decision Trees: Information Gain

\[
\text{Gain}(S, A) = \text{expected reduction in entropy due to branching on attribute } A = \text{original entropy} - \text{entropy after branching} = H\left(\left[\frac{\#\text{pos}}{\#\text{pos} + \#\text{neg}}, \frac{\#\text{neg}}{\#\text{pos} + \#\text{neg}}\right]\right)
\]

\[-\sum_{j=1}^{J} \frac{\#\text{pos}_j + \#\text{neg}_j}{\#\text{pos} + \#\text{neg}} H\left[\frac{\#\text{pos}_j}{\#\text{pos}_j + \#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j + \#\text{neg}_j}\right].\]

Encourages too many branches?
Another splitting criterion: Information Gain Ratio (Quinlan, 1986)

\[
\text{Gain}(S, A) \leftarrow \text{want large}
\]

\[
\text{SplitInfo}(S, A) = - \sum_{j=1}^{J} \left( \frac{|S_j|}{|S|} \right) \log \left( \frac{|S_j|}{|S|} \right)
\]

|\(|S_j|\) is the amount of data in branch \(j\)|

\[
\text{Want large}
\]
More splitting criteria for binary splitting

Other replacements for $H([p,1-p])$

- **Gini Index** (used by CART, Breiman, 1984)
  
  Gini index = $2p(1-p) = \text{variance of Bernoulli}$

- **Misclassification error** $1-\max(p,1-p)$
  
  Classify according to majority vote: if $p \leq .5$ vote no, otherwise yes

<table>
<thead>
<tr>
<th>Yes, No</th>
<th>Predicted</th>
<th>Made Errors</th>
<th>$p$</th>
<th>$1-\max(p,1-p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 4</td>
<td>yes</td>
<td>4/10</td>
<td>.6</td>
<td>$1-\max(.6,.4)=1-.6=.4$</td>
</tr>
<tr>
<td>2, 8</td>
<td>no</td>
<td>2/10</td>
<td>.2</td>
<td>$1-\max(.2,.8)=1-.8=.2$</td>
</tr>
<tr>
<td>7, 3</td>
<td>vote yes</td>
<td>3/10</td>
<td>.7</td>
<td>$1-\max(.7,.3)=1-.7=.3$</td>
</tr>
</tbody>
</table>
More splitting criteria for binary splitting

```
>> p=[0:.01:1];
entr=(-p.*log2(p)-(1-p).*log2(1-p))*0.5;
gini=2.*p.*(1-p);
miscl=1-max(p,1-p);
plot(p,entr,'r')
hold on
plot(p,gini,'b')
plot(p,miscl,'g')
```
Keep splitting until:
• All observations in each leaf have the same class
• No more features to split (you split on all of them already)

This is likely to overfit
Decision Trees Part 3

Cynthia Rudin
Machine Learning Class
Duke University
Pruning

C4.5 recursively makes choices for pruning:

- Option 1: leaving the tree as is
- Option 2: collapse that part of the tree into a leaf. The leaf has the most frequent label in the data $S$ going to that part of the tree.
- Option 3: replace that part of the tree with one of its subtrees, corresponding to the most common branch in the split
Pruning

Which of the three options?

C4.5 computes upper bounds on the probability of error for each option. It uses standard upper confidence bounds on probabilities from the binomial distribution with $\alpha=0.25$.

- Prob of error for Option 1 $\leq$ UpperBound1
- Prob of error for Option 2 $\leq$ UpperBound2
- Prob of error for Option 3 $\leq$ UpperBound3
New dataset (from the Kranf site)

Consider split on Crowded

Option 1: Leave the tree as it is

<table>
<thead>
<tr>
<th>x_1</th>
<th>Some</th>
<th>$$</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_2</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_3</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_4</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_5</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_6</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_7</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_8</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_9</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{10}</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{11}</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{12}</td>
<td>Full</td>
<td>$$</td>
<td>No</td>
</tr>
<tr>
<td>x_{13}</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{14}</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{15}</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x_{16}</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{Crowded} & \\
\text{Some} & \quad 6 \text{ pos}, \ 0 \text{ neg} \\
& \quad \begin{cases} \text{Vote yes} & 0 \pm .206 \\ \text{Vote no} & 0 \pm .143 \end{cases} \\
\text{None} & \quad 9 \text{ pos}, \ 0 \text{ neg} \\
& \quad \begin{cases} \text{Vote yes} & 0 \pm .143 \\ \text{Vote no} & 0 \pm .75 \end{cases} \\
\text{Full} & \quad 0 \text{ pos}, \ 1 \text{ neg} \\
\end{align*}
\]

>> binocdf(0,6,.206)
ans = 0.2506

>> binocdf(0,9,.143)
ans = 0.2494

>> binocdf(0,1,.75)
ans = 0.2500
New dataset (from the Kranf site)

Consider split on Crowded

Option 1: Leave the tree as it is

<table>
<thead>
<tr>
<th></th>
<th>Crowded</th>
<th>Price</th>
<th>Stay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Some</td>
<td>$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_3$</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_6$</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_7$</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_8$</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_9$</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Full</td>
<td>$$</td>
<td>No</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{15}$</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_{16}$</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

UpperBound1 = average upper bound over leaves

\[
= \frac{1}{16} (6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75 )
\]
New dataset (from the Kranf site)

Consider split on Crowded
Option 2 & 3: Prune the tree to a leaf

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>Crowded</th>
<th>Price</th>
<th>Stay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_2 )</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_6 )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_7 )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_8 )</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_9 )</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{10} )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{11} )</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{12} )</td>
<td>Full</td>
<td>$$$$</td>
<td>No</td>
</tr>
<tr>
<td>( x_{13} )</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{14} )</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{15} )</td>
<td>Some</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>( x_{16} )</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### New dataset

Consider split on Crowded

Option 2 & 3: Prune the tree to a leaf

Crowded

15 pos, 1 neg

\[
\text{UpperBound1} = \text{average upper bound over leaves} = \frac{1}{16} (6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75) = 0.204
\]

\[
\text{UpperBound2} = \text{average upper bound over leaves} = \frac{1}{16} (16 \times 0.157) = 0.157
\]

\[
>> \text{binocdf}(1,16,0.157) \\
\text{ans} = 0.2589
\]

<table>
<thead>
<tr>
<th></th>
<th>Crowded</th>
<th>Price</th>
<th>Stay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₂</td>
<td>Some</td>
<td>$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₃</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₄</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₅</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₆</td>
<td>None</td>
<td>$$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₇</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₈</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₉</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₀</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₁</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₂</td>
<td>Full</td>
<td>$$</td>
<td>No</td>
</tr>
<tr>
<td>x₁₃</td>
<td>None</td>
<td>$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₄</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₅</td>
<td>Some</td>
<td>$$$</td>
<td>Yes</td>
</tr>
<tr>
<td>x₁₆</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Consider split on Crowded

Option 2 & 3: Prune the tree to a leaf

### UpperBound1

\[
\text{UpperBound1} = \text{average upper bound over leaves} \\
= \frac{1}{16} (6 \times 0.206 + 9 \times 0.143 + 1 \times 0.75) = 0.204
\]

### UpperBound2

\[
\text{UpperBound2} = \text{average upper bound over leaves} \\
= \frac{1}{16} (16 \times 0.157) = 0.157
\]

---

<table>
<thead>
<tr>
<th>Crowded</th>
<th>Price</th>
<th>Stay?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>Some</td>
<td>$$</td>
</tr>
<tr>
<td>x₂</td>
<td>Some</td>
<td>$$$$</td>
</tr>
<tr>
<td>x₃</td>
<td>None</td>
<td>$$</td>
</tr>
<tr>
<td>x₄</td>
<td>Some</td>
<td>$$</td>
</tr>
<tr>
<td>x₅</td>
<td>None</td>
<td>$$</td>
</tr>
<tr>
<td>x₆</td>
<td>None</td>
<td>$$$$</td>
</tr>
<tr>
<td>x₇</td>
<td>None</td>
<td>$$</td>
</tr>
<tr>
<td>x₈</td>
<td>None</td>
<td>$</td>
</tr>
<tr>
<td>x₉</td>
<td>Some</td>
<td>$$</td>
</tr>
<tr>
<td>x₁₀</td>
<td>None</td>
<td>$$</td>
</tr>
<tr>
<td>x₁₁</td>
<td>Some</td>
<td>$</td>
</tr>
<tr>
<td>x₁₂</td>
<td>Full</td>
<td>$$</td>
</tr>
<tr>
<td>x₁₃</td>
<td>None</td>
<td>$$</td>
</tr>
<tr>
<td>x₁₄</td>
<td>None</td>
<td>$</td>
</tr>
<tr>
<td>x₁₅</td>
<td>Some</td>
<td>$$$$</td>
</tr>
<tr>
<td>x₁₆</td>
<td>None</td>
<td>$</td>
</tr>
</tbody>
</table>
Other questions

Q. Where did $\alpha=.25$ come from?

Q. How do you know which subtrees to consider?

Q. Can I change it?

Q. Are the trees “optimal”? 
I figured something out! Do you want to see it?

That’s not useful. If you want a thesis you should study something else...

But... I have a thesis... 😊

CART – Classification and Regression Trees
(Breiman, Friedman, Olshen, Stone, 1984)

me
Leo Breiman
CART – *Classification and Regression Trees*
(Breiman, Friedman, Olshen, Stone, 1984)

- Does only binary splits, not multiway splits.
- Uses the Gini index for splitting.
- Uses **Minimum Cost Complexity** for pruning

Each subtree is assigned a cost. Choose the subtree with the lowest cost

\[
\text{Misclassification Error} \downarrow \quad \text{Regularization} \downarrow \quad \text{cost(subtree)} = \sum_{\text{leaves } j} \sum_{x_i \in \text{leaf } j} 1[y_i \neq \text{leaf's class}] + C [\# \text{leaves in subtree}]
\]

Each new leaf costs C.
That means each new leaf is worth the same as C misclassified points.
CART – Classification and \textit{Regression} Trees
(Breiman, Friedman, Olshen, Stone, 1984)

• For regression, assign $f(x)$ to be constant in each leaf.
• Choose value of $f(x)$ to minimize the squared loss:

$$R_{\text{train}}(f) = \sum_i (y_i - f(x_i))^2$$

\begin{align*}
= & \sum_{\text{leaves } j} \sum_{i \in \text{leaf } j} (y_i - f(x_i))^2 \quad \text{(Group terms by leaf)} \\
= & \sum_{\text{leaves } j} \sum_{i \in \text{leaf } j} (y_i - f_j)^2 =: \sum_{\text{leaves } j} R_{\text{train}}^j(f_j).
\end{align*}

(Constant $f$ in leaf) \hspace{2cm} Choose $f_j$ to minimize $R_{\text{train}}^j(f_j)$
0 = \frac{d}{df} \sum_{i \in \text{leaf}_j} (y_i - \tilde{f})^2 \bigg|_{\tilde{f}=f_j} \\
= -2 \sum_i (y_i - \tilde{f}) \bigg|_{\tilde{f}=f_j} = -2 \left( \left( \sum_i y_i \right) - |S_j| \tilde{f} \right) \bigg|_{\tilde{f}=f_j} \\
f_j = \frac{1}{|S_j|} \sum_{i \in \text{leaf}_j} y_i = \bar{y}_{S_j},

where \( \bar{y}_{S_j} \) is the sample average of the labels for leaf \( j \)'s examples.

Choose \( f_j \) to minimize \( R_j^{\text{train}}(f_j) \)
Decision tree for regression

1. if x2 < 3085.5 then node 2 else if x2 >= 3085.5 then node 3 else node 1
2. if x1 < 89 then node 4 else if x1 >= 89 then node 5 else 28.7931
3. if x1 < 115 then node 6 else if x1 >= 115 then node 7 else 15.5417
4. if x2 < 2162 then node 8 else if x2 >= 2162 then node 9 else 30.9375
5. if x2 < 2754.5 then node 10 else if x2 >= 2754.5 then node 11 else 24.0882
6. fit = 19.625
7. if x1 < 191.5 then node 12 else if x1 >= 191.5 then node 13 else 14.375
8. if x2 < 1951 then node 14 else if x2 >= 1951 then node 15 else 33.3056
9. if x2 < 3013.5 then node 16 else if x2 >= 3013.5 then node 17
10. fit = 25.5556
11. fit = 22.4375
12. if x2 < 3533.5 then node 18 else if x2 >= 3533.5 then node 19
13. fit = 12.3333
14. fit = 29.375
15. if x1 < 74.5 then node 20 else if x1 >= 74.5 then node 21
16. if x2 < 2581 then node 22 else if x2 >= 2581 then node 23
17. fit = 38
18. fit = 17.25
19. if x2 < 4122.5 then node 24 else if x2 >= 4122.5 then node 25
20. if x2 < 2127.5 then node 26 else if x2 >= 2127.5 then node 27
21. fit = 30.3333
22. if x2 < 2203.5 then node 28 else if x2 >= 2203.5 then node 29

```MATLAB```
>> load carsmall % Contains Horsepower, Weight, MPG
X = [Horsepower Weight];
Mdl = fitrtree(X,MPG);
Mdl = RegressionTree
   Properties, Methods
>> view(Mdl)
```
For real-valued features, CART chooses feature $j$ to split, split point $s$, values $C_1$ and $C_2$ for leaves all at the same time.

$$\min_{j, s} \left[ \min_{C_1} \sum_{x_i \in \{ \text{leaf} | x^{(j)} \leq s \}} (y_i - C_1)^2 + \min_{C_2} \sum_{x_i \in \{ \text{leaf} | x^{(j)} > s \}} (y_i - C_2)^2 \right]$$
For real-valued features, CART chooses feature $j$ to split, split point $s$, values $C_1$ and $C_2$ for leaves all at the same time.

\[
\min_{j, s} \min_{C_1} \sum_{x_i \in \{\text{leaf} | x^{(j)} \leq s\}} (y_i - C_1)^2 + \min_{C_2} \sum_{x_i \in \{\text{leaf} | x^{(j)} > s\}} (y_i - C_2)^2
\]

For each feature $j$ do a linesearch over $s$

\[
C_1 = \bar{y}_{\{\text{leaf} | x^{(j)} \leq s\}}
\]

\[
C_2 = \bar{y}_{\{\text{leaf} | x^{(j)} > s\}}
\]
For regression, CART also does cost-complexity pruning

\[ \text{cost} = \sum_{\text{leaves } j} \sum_{x_i \in S_j} (y_i - \bar{y}_{S_j})^2 + C[\# \text{ leaves in tree}] \]
Some Perspective

• We have learned several splitting and pruning procedures. We know how CART and C4.5 work.

• Why are we studying decision trees again?
  1) If you combine many decision trees, amazing results!
  2) Decision trees are interpretable (well, CART is, C4.5 not so much)

Greedy is not the only way to train an interpretable decision tree.
Advantages of Decision Tree Methods CART and C4.5 (and friends)

- Interpretable (at least CART)
- Handles nonlinearities
- Greedy, so very fast
- Can easily handle imbalanced data by reweighting the points.

Disadvantages of CART and C4.5

- Greedy, less accurate than other methods
- Heuristic algorithms – not elegant
- No proof of optimality
- No proof of nearness to optimality
- Tend to do poorly for imbalanced data, even after adjusting the parameters.
- CART tends to be more interpretable but less accurate than C4.5, but C4.5 produces completely uninterpretable models by default.
Modern Decision Tree Methods

• Aim for full minimization of the Cost Complexity objective

\[
\min_{\text{tree}} \hat{L}(\text{tree}, \{(x_i, y_i)\}_i) \text{ where }
\]

\[
\hat{L}(\text{tree}, \{(x_i, y_i)\}_i) = \frac{1}{n} \sum_{i=1}^{n} 1[\text{tree}(x_i) \neq y_i] + \lambda (\# \text{ leaves in tree})
\]

Misclassification Error  Sparsity

• Most recent method is called GOSDT – Generalized and Scalable Optimal Sparse Decision Trees (Lin et al, ICML 2020) – beyond scope of course
Modern Decision Tree Methods

- rain?
  - Y
  - N
    - construction?
      - Y
      - N
        - traffic
        - no traffic
    - rush hour?
      - Y
      - N
        - construction?
          - Y
          - N
            - no traffic
            - traffic
        - Friday?
          - Y
          - N
            - no traffic
            - traffic
Modern Decision Tree Methods

- Aim for full minimization of the Cost Complexity objective

\[
\min_{\text{tree}} \hat{L}(\text{tree}, \{(x_i, y_i)\}_i) \text{ where }
\]

\[
\hat{L}(\text{tree}, \{(x_i, y_i)\}_i) = \ell(\text{tree}, \{(x_i, y_i)\}_i) + \lambda(\# \text{ leaves in tree})
\]

Generalized Objectives

- Any loss function monotonically increasing in FP and FN
  - Balanced accuracy, weighted accuracy, F-1, precision, …
- Rank statistics
  - AUC and partial AUC under the ROC convex hull
- Can prove optimality or nearness to optimality for not-huge datasets

Sparsity
Modern Decision Tree Methods

• Not greedy splitting and pruning

• Sophisticated pairing of theoretical bounds for reducing the search space with techniques for careful storage+lookup of previously-solved subproblems.

(Dynamic programming + analytical bounds)
Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

Several theorems show that some partial trees can never be extended to form optimal trees.
Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

Several theorems show that some partial trees can never be extended to form optimal trees.
Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

Several theorems show that some partial trees can never be extended to form optimal trees.
\[ R(\text{tree}) = \frac{1}{n} \sum_{i=1}^{n} 1_{[y_i \neq \text{tree}(x_i)]} + C(\# \text{leaves in tree}) \]

**Theorem (Basic Size Bound).**
If we have previously encountered a tree with objective \( R^c \), then any optimal tree, \( \text{tree}^* \in \arg \min_{\text{tree}} R(\text{tree}) \), obeys

\[ \# \text{leaves in tree}^* \leq \lfloor R^c / C \rfloor. \]

\[
\begin{align*}
R(\text{tree}^*) & \leq R^c \quad \text{(definition of tree}^* \text{ as optimal)} \\
\frac{1}{n} \sum_{i=1}^{n} 1_{[y_i \neq \text{tree}^*(x_i)]} + C(\# \text{leaves in tree}^*) & \leq R^c \\
C(\# \text{leaves in tree}^*) & \leq R^c \\
\# \text{leaves in tree}^* & \leq \lfloor R^c / C \rfloor
\end{align*}
\]
Many other bounds

- Theorems prove certain splits are not possible because they won’t lead to accurate enough trees.

- We can sometimes prove that one partial tree can never be extended to form a better tree than one we have already seen.

- There are many isomorphisms of the same tree. We need only work with one of them.

- Trees that are very similar to each other have similar objective values.
Generalized and Scalable Optimal Sparse Decision Trees

Advantages of GOSDT
• Interpretable, simple models that generalize well
• Handles nonlinearities
• Handles wide variety of objectives, even custom objectives
• Handles imbalanced data
• Proof of optimality, or closeness to optimality
• No splitting and pruning heuristics

Disadvantages of GOSDT
• Can’t prove optimality for big datasets… yet