Decision Trees

Lecturer: Ji Liu

Thank Jerry Zhu for sharing his slides

[ Some slides from Andrew Moore http://www.cs.cmu.edu/~awm/tutorials and Chuck Dyer, with permission.]
The input

These names are the same: example, point, instance, item, input

Usually represented by a feature vector

- These names are the same: attribute, feature
- For decision trees, we will especially focus on discrete features (though continuous features are possible, see end of slides)
• The output
• These names are the same: label, target, goal
• It can be
  – Continuous, as in our population prediction ➔ Regression
  – Discrete, e.g., is this mushroom x edible or poisonous? ➔ Classification
Evaluating classifiers

- **During training**
  - Train a classifier from a training set \((x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\).

- **During testing**
  - For new test data \(x_{n+1}\ldots x_{n+m}\), your classifier generates predicted labels \(y'_{n+1}\ldots y'_{n+m}\).

- **Test set accuracy:**
  - You need to know the true test labels \(y_{n+1}\ldots y_{n+m}\).
  - Test set accuracy: \(acc = \frac{1}{m} \sum_{i=n+1}^{n+m} 1_{y_i = y'_i}\)
  - Test set error rate = \(1 - acc\)
Decision Trees

• One kind of classifier (supervised learning)

• Outline:
  – The tree
  – Algorithm
  – Mutual information of questions
  – Overfitting and Pruning
  – Extensions: real-valued features, tree→rules, pro/con
Akinator: Decision Tree

A Decision Tree

• A decision tree has 2 kinds of nodes
  1. Each leaf node has a class label, determined by majority vote of training examples reaching that leaf.
  2. Each internal node is a question on features. It branches out according to the answers.
### Automobile Miles-per-gallon prediction

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>modelyear</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4 low</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>75to78</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6 medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4 medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>europe</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8 high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6 medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4 low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8 high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6 high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8 high</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>8 high</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8 low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4 low</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>6 medium</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4 medium</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4 low</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8 high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4 low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>5 medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
<td></td>
</tr>
</tbody>
</table>
A very small decision tree

mpg values: bad good

Internal node question: “what is the number of cylinders”?

Leaves: classify by majority vote
A bigger decision tree

question: “what is the value of maker”?

question: “what is the value of horsepower”?

mpg values: bad good

Predict “good” is also reasonable by following its parent node instead of the root node.
1. Do not split when all examples have the same label

2. Can not split when we run out of questions
Decision tree algorithm

`buildtree(examples, questions, default)`
/* examples: a list of training examples
   questions: a set of candidate questions, e.g., “what’s the value of feature $x_i$?”
   default: default label prediction, e.g., over-all majority vote */

IF empty(examples) THEN return(default)
IF (examples have same label y) THEN return(y)
IF empty(questions) THEN return(majority vote in examples)

q = best_question(examples, questions)

Let there be n answers to q
  – Create and return an internal node with n children
  – The i$^\text{th}$ child is built by calling `buildtree({example|q=}$i$\text{th answer}$, questions\{q\}, default)`
The best question

• What do we want: pure leaf nodes, i.e. all examples having (almost) the same $y$.
• A good question $\rightarrow$ a split that results in pure child nodes
• How do we measure the degree of purity induced by a question? Here’s one possibility (Max-Gain in book):
  
  mutual information  
  (a.k.a. information gain)
  A quantity from information theory
Entropy (Impurity Measure)

- At the current node, there are \( n = n_1 + \ldots + n_k \) examples
  - \( n_1 \) examples have label \( y_1 \)
  - \( n_2 \) examples have label \( y_2 \)
  - \( \ldots \)
  - \( n_k \) examples have label \( y_k \)

- What’s the impurity of the node?
- Turn it into a game: if I put these examples in a bag, and grab one at random, what is the probability the example has label \( y_i \)?
Entropy (Impurity Measure)

- Probability estimated from samples:
  - with probability $p_1 = \frac{n_1}{n}$ the example has label $y_1$
  - with probability $p_2 = \frac{n_2}{n}$ the example has label $y_2$
  - ...
  - with probability $p_k = \frac{n_k}{n}$ the example has label $y_k$
- $p_1 + p_2 + ... + p_k = 1$
- The “outcome” of the draw is a random variable $y$ with probability $(p_1, p_2, ..., p_k)$
- What’s the impurity of the node ➔ what’s the uncertainty of $y$ in a random drawing?
Entropy (Impurity Measure)

\[ H(Y) = \sum_{i=1}^{k} -\Pr(Y = y_i) \log_2 \Pr(Y = y_i) \]

\[ = \sum_{i=1}^{k} -p_i \log_2 p_i. \]

- Interpretation: The number of yes/no questions (bits) needed on average to pin down the value of \( y \) in a random drawing.

\[ H(y) = \]

\[ H(y) = \]

\[ H(y) = \]
Entropy (Impurity Measure)

- **p(head)=0.5**  
  **p(tail)=0.5**  
  **H=1**

- **p(head)=0.51**  
  **p(tail)=0.49**  
  **H=0.9997**

- **p(head)=1**  
  **p(tail)=0**  
  **H=0 (Why?)**

Jerry's coin
Excellent Video for Entropy

https://www.youtube.com/watch?v=R4OlXb9aTvQ

• Entropy roughly measures the average number of yes/no questions we need to ask to figure out the class label of an object without any additional attribute information.
Conditional entropy

\[
H(Y|X = v) = \sum_{i=1}^{k} - \Pr(Y = y_i|X = v) \log_2 \Pr(Y = y_i|X = v)
\]

\[
H(Y|X) = \sum_{v: \text{values of } X} \Pr(X = v) H(Y|X = v)
\]

• Y: label. X: a question (e.g., a feature). v: an answer to the question
• Pr(Y|X=v): conditional probability
• H(Y|X) estimates the average number of y/n questions required after know the attribute information X
Information gain

- Information gain, or mutual information

\[ I(Y;X) = H(Y) - H(Y|X) \]

- Choose question (feature) \( X \) which maximizes \( I(Y;X) \).
Example

• Features: color, shape, size

• What’s the best question at root?
The training set

<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[ H(\text{class}) = \]
\[ H(\text{class} \mid \text{color}) = \]
<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[
H(\text{class}) = H(3/6, 3/6) = 1
\]

\[
H(\text{class} \mid \text{color}) = \frac{3}{6} \cdot H(2/3, 1/3) + \frac{1}{6} \cdot H(1, 0) + \frac{2}{6} \cdot H(0, 1)
\]

- 3 out of 6 are red
- 2 out of the red are +
- 1 out of 6 is blue
- 2 out of 6 are green
- 1 out of 6 is blue
- blue is +
- green is -
<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[ H(\text{class}) = H(3/6, 3/6) = 1 \]
\[ H(\text{class} | \text{color}) = 3/6 \times H(2/3, 1/3) + 1/6 \times H(1, 0) + 2/6 \times H(0, 1) \]
\[ I(\text{class}; \text{color}) = H(\text{class}) - H(\text{class} | \text{color}) = 0.54 \text{ bits} \]
<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[ H(\text{class}) = H(3/6,3/6) = 1 \]
\[ H(\text{class} | \text{shape}) = 4/6 \times H(1/2,1/2) + 2/6 \times H(1/2,1/2) \]
\[ I(\text{class}; \text{shape}) = H(\text{class}) - H(\text{class} | \text{shape}) = 0 \text{ bits} \]

Shape tells us nothing about the class!
<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[
H(\text{class}) = H(3/6, 3/6) = 1
\]

\[
H(\text{class} \mid \text{size}) = 4/6 \times H(3/4, 1/4) + 2/6 \times H(0, 1)
\]

\[
I(\text{class}; \text{size}) = H(\text{class}) - H(\text{class} \mid \text{size}) = 0.46 \text{ bits}
\]
<table>
<thead>
<tr>
<th>Example</th>
<th>Color</th>
<th>Shape</th>
<th>Size</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Red</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>Square</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Red</td>
<td>Circle</td>
<td>Big</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>Red</td>
<td>Circle</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Green</td>
<td>Square</td>
<td>Small</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Green</td>
<td>Square</td>
<td>Big</td>
<td>-</td>
</tr>
</tbody>
</table>

\[
I(\text{class}; \text{color}) = H(\text{class}) - H(\text{class} \mid \text{color}) = 0.54 \text{ bits}
\]

\[
I(\text{class}; \text{shape}) = H(\text{class}) - H(\text{class} \mid \text{shape}) = 0 \text{ bits}
\]

\[
I(\text{class}; \text{size}) = H(\text{class}) - H(\text{class} \mid \text{size}) = 0.46 \text{ bits}
\]

→ We select **color** as the question at root
Overfitting

• Overfitting happens if the prediction model is overcomplicated while the training data is few.

• Another perspective to say overfitting is the model fits the training data perfectly.

• https://www.youtube.com/watch?v=iILj9g8xObc
Example: Overfitting in SVM
Example: Overfitting in regression: Predicting US Population

- We have some training data \((n=11)\)
- What will the population be in 2020?

<table>
<thead>
<tr>
<th>Year</th>
<th>Million</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>75.995</td>
</tr>
<tr>
<td>1910</td>
<td>91.972</td>
</tr>
<tr>
<td>1920</td>
<td>105.71</td>
</tr>
<tr>
<td>1930</td>
<td>123.2</td>
</tr>
<tr>
<td>1940</td>
<td>131.67</td>
</tr>
<tr>
<td>1950</td>
<td>150.7</td>
</tr>
<tr>
<td>1960</td>
<td>179.32</td>
</tr>
<tr>
<td>1970</td>
<td>203.21</td>
</tr>
<tr>
<td>1980</td>
<td>226.51</td>
</tr>
<tr>
<td>1990</td>
<td>249.63</td>
</tr>
<tr>
<td>2000</td>
<td>281.42</td>
</tr>
</tbody>
</table>
Regression: Polynomial fit

• The degree $d$ (complexity of the model) is important

$$ f(x) = c_d x^d + c_{d-1} x^{d-1} + \cdots + c_1 x + c_0 $$

• Fit (=learn) coefficients $c_d, \ldots, c_0$ to minimize Mean Squared Error (MSE) on training data

$$ MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 $$
Overfitting

- As $d$ increases, MSE on training data improves, but prediction outside training data worsens.
Overfitting: Toy Example

• Predict if the outcome of throwing a die is “6” from its (color, size)
• Color = \{red, blue\}, Size=\{small, large\}
• Three training samples:
  – X1 = (red, large), y1 = not 6
  – X2 = (blue, small), y2 = not 6
  – X3 = (blue, large), y3 = 6
Overfitting: Example for Decision Tree

- Three training samples:
  - $X_1 = (\text{red, large}), y_1 = \text{not 6}$
  - $X_2 = (\text{blue, small}), y_2 = \text{not 6}$
  - $X_3 = (\text{blue, large}), y_3 = \text{6}$
Toy Example

• Assume “color” and “size” are independent attributes for any die
• Assume $P(\text{red})=P(\text{blue})=1/2$, $P(\text{large})=P(\text{small})=1/2$
• The prediction accuracy for this decision tree is $1-(1/2*1/6+1/4*5/6 + 1/4*1/6)=2/3$
Toy Example

- If the decision tree only has the root node, we predict all new instances as “Not 6”.
- The accuracy is $\frac{5}{6} > \frac{2}{3}$.
Overfit a decision tree

Five inputs, all bits, are generated in all 32 possible combinations

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Output $y = \text{copy of } e$
Except a random 25% of the records have $y$ set to the opposite of $e$
Overfit a decision tree

• The test set is constructed similarly
  – \( y = e \), but 25% the time we corrupt it by \( y = \neg e \)
  – The corruptions in training and test sets are independent

• The training and test sets are the same, except
  – Some \( y \)'s are corrupted in training, but not in test
  – Some \( y \)'s are corrupted in test, but not in training
Overfit a decision tree

- We build a full tree on the training set

Training set accuracy = 100%
25% of these training leaf node labels will be corrupted (≠e)
Overfit a decision tree

- And classify the test data with the tree

25% of the test examples are corrupted - independent of training data
Overfit a decision tree

On average:

• \( \frac{3}{4} \) training data uncorrupted
  – \( \frac{3}{4} \) of these are uncorrupted in test – correct labels
  – \( \frac{1}{4} \) of these are corrupted in test – wrong

• \( \frac{1}{4} \) training data corrupted
  – \( \frac{3}{4} \) of these are uncorrupted in test – wrong
  – \( \frac{1}{4} \) of these are also corrupted in test – correct labels

• Test accuracy = \( \frac{3}{4} \times \frac{3}{4} + \frac{1}{4} \times \frac{1}{4} = \frac{5}{8} = 62.5\% \)
Overfit a decision tree

- But if we knew a, b, c, d are irrelevant features and don’t use them in the tree...

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Pretend they don’t exist
Overfit a decision tree

- The tree would be

```
Root

e=0

e=1
```

In training data, about \( \frac{3}{4} \) y’s are 0 here. Majority vote predicts \( y=0 \)

In training data, about \( \frac{3}{4} \) y’s are 1 here. Majority vote predicts \( y=1 \)

In test data, \( \frac{1}{4} \) y’s are different from e.

Test accuracy = ?
Overfit a decision tree

- The tree would be

```
Root
   e=0
   e=1
```

In training data, about $\frac{3}{4}$ y’s are 0 here. Majority vote predicts $y=0$

In training data, about $\frac{3}{4}$ y’s are 1 here. Majority vote predicts $y=1$

In test data, $\frac{1}{4}$ y’s are different from e.

Test accuracy $= \frac{3}{4} = 75\%$ (better!)

Full tree test accuracy $= \frac{3}{4} \times \frac{3}{4} + \frac{1}{4} \times \frac{1}{4} = \frac{5}{8} = 62.5\%$
Overfit a decision tree

• In the full tree, we overfit by learning non-existent relations (noise)
Avoid overfitting: pruning

Pruning with a tuning set

1. Randomly split data into TRAIN and TUNE, say 70% and 30%
2. Build a full tree using only TRAIN
3. Prune the tree down on the TUNE set. On the next page you’ll see a greedy version.
Pruning

Prune(tree T, TUNE set)

1. Compute T’s accuracy on TUNE, call it A(T)

2. For every internal node N in T:
   a) New tree $T_N =$ copy of T, but prune (delete) the subtree under N.
   b) $N$ becomes a leaf node in $T_N$. The label is the majority vote of TRAIN examples reaching $N$.
   c) $A(T_N) =$ $T_N$’s accuracy on TUNE

3. Let $T^*$ be the tree (among the $T_N$’s and T) with the largest $A()$. Set $T \leftarrow T^*$  /* prune */

4. Repeat from step 1 until no more improvement available. Return T.
Real-valued features

• What if some (or all) of the features $x_1, x_2, \ldots, x_k$ are real-valued?
• Example: $x_1=$ height (in inches)
• Idea 1: branch on each possible numerical value.
Real-valued features

• What if some (or all) of the features $x_1, x_2, \ldots, x_k$ are real-valued?
• Example: $x_1 =$ height (in inches)
• Idea 1: branch on each possible numerical value. (fragments the training data and prone to overfitting)
• Idea 2: use questions in the form of $(x_1 > t?)$, where $t$ is a threshold. There are fast ways to try all (?) $t$.

\[
H(y \mid x_i > t?) = p(x_i > t)H(y \mid x_i > t) + p(x_i \leq t)H(y \mid x_i \leq t)
\]
\[
I(y \mid x_i > t?) = H(y) - H(y \mid x_i > t?)
\]
What does the feature space look like?

Axis-parallel cuts
Conclusions

• Decision trees are popular tools for data mining
  – Easy to understand
  – Easy to implement
  – Easy to use
  – Computationally cheap

• Overfitting might happen

• We used decision trees for classification
  (predicting a categorical output from categorical or real inputs)
What you should know

• Trees for classification
• Top-down tree construction algorithm
• Information gain
• Overfitting
• Pruning
• Real-valued features