So far ...

- Decision trees
- They will overfit
- How to split?
- When to stop?
What defines a good attribute?

Ideal split

Which one do you prefer?
Splitting: choosing a good attribute

Would we prefer to split on $X_1$ or $X_2$?

Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!
Measuring uncertainty

• Good split if we are more certain about classification after split
  – Deterministic good (all true or all false)
  – Uniform distribution bad
  – What about distributions in between?

<table>
<thead>
<tr>
<th>( P(Y=A) )</th>
<th>( P(Y=B) )</th>
<th>( P(Y=C) )</th>
<th>( P(Y=D) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1/4</td>
<td>1/8</td>
<td>1/8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( P(Y=A) )</th>
<th>( P(Y=B) )</th>
<th>( P(Y=C) )</th>
<th>( P(Y=D) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>
Entropy

Entropy $H(Y)$ of a random variable $Y$

$$H(Y) = - \sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation:

$H(Y)$ is the expected number of bits needed to encode a randomly drawn value of $Y$ (under most efficient code)
Entropy Example

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

\[ P(Y=t) = \frac{5}{6} \]
\[ P(Y=f) = \frac{1}{6} \]

\[ H(Y) = - \frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6} \]
\[ = 0.65 \]
Conditional Entropy

Conditional Entropy $H(Y|X)$ of a random variable $Y$ conditioned on a random variable $X$

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

Example:

\[
P(X_1 = t) = \frac{4}{6} \\
P(X_1 = f) = \frac{2}{6}
\]

\[
H(Y|X_1) = - \frac{4}{6} (1 \log_2 1 + 0 \log_2 0) - \frac{2}{6} (1/2 \log_2 1/2 + 1/2 \log_2 1/2)
\]

= $\frac{2}{6}$
Information gain

Decrease in entropy (uncertainty) after splitting

\[ IG(X) = H(Y) - H(Y | X) \]

- \( IG(X) \) is non-negative (>=0)
- Prove by showing \( H(Y|X) \leq H(X) \), with Jensen’s inequality

In our running example:

\[ IG(X_1) = H(Y) - H(Y|X_1) \]
\[ = 0.65 - 0.33 \]

\( IG(X_1) > 0 \rightarrow \) we prefer the split!
Learning decision trees

• Start from empty decision tree
• Split on next best attribute (feature)
  – Use, for example, information gain to select attribute:

\[
\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y \mid X_i)
\]

• Recurse
Suppose we want to predict MPG

Look at all the information gains...
A Decision Stump

mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad

cylinders = 4
4 17
Predict good

cylinders = 5
1 0
Predict bad

cylinders = 6
8 0
Predict bad

cylinders = 8
9 1
Predict bad

First split looks good! But, when do we stop?
Don’t split a node if all matching records have the same output value.
Don’t split a node if none of the attributes can create multiple non-empty children.
Base Case Two: No attributes can distinguish
Base Cases: An idea

• **Base Case One**: If all records in current data subset have the same output then don’t recurse

• **Base Case Two**: If all records have exactly the same set of input attributes then don’t recurse

**Proposed Base Case 3:**
If all attributes have zero information gain then don’t recurse

• *Is this a good idea?*
The problem with Base Case 3

\[ y = a \text{ XOR } b \]

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

The information gains:

The resulting decision tree:

\[
\begin{align*}
\text{y values:} & & 0 & 1 \\
\text{root} & & 2 & 2 \\
\text{Predict 0}
\end{align*}
\]
If we omit Base Case 3:

\[ y = a \text{ XOR } b \]

\[
\begin{array}{ccc}
    a & b & y \\
    0 & 0 & 0 \\
    0 & 1 & 1 \\
    1 & 0 & 1 \\
    1 & 1 & 0 \\
\end{array}
\]

Is it OK to omit Base Case 3?
Summary: Building Decision Trees

BuildTree($\text{DataSet,Output}$)

• If all output values are the same in $\text{DataSet}$, return a leaf node that says “predict this unique output”

• If all input values are the same, return a leaf node that says “predict the majority output”

• Else find attribute $X$ with highest Info Gain

• Suppose $X$ has $n_X$ distinct values (i.e. $X$ has arity $n_X$).
  – Create a non-leaf node with $n_X$ children.
  – The $i$’th child should be built by calling
    
    BuildTree($DS_i,\text{Output}$)

    Where $DS_i$ contains the records in $\text{DataSet}$ where $X = i$th value of $X$. 
The test set error is much worse than the training set error…

…why?
Decision trees will overfit!!!

• **Standard decision trees have no learning bias**
  – Training set error is always zero!
    • (If there is no label noise)
  – Lots of variance
  – Must introduce some bias towards simpler trees

• **Many strategies for picking simpler trees**
  – Fixed depth
  – Fixed number of leaves
  – Or something smarter...
Decision trees will overfit!!!
One Definition of Overfitting

• Assume:
  – Data generated from distribution $D(X,Y)$
  – A hypothesis space $H$

• Define errors for hypothesis $h \in H$
  – Training error: $\text{error}_{\text{train}}(h)$
  – Data (true) error: $\text{error}_D(h)$

• We say $h$ overfits the training data if there exists an $h' \in H$ such that:
  \[
  \text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')
  \]
  and
  \[
  \text{error}_D(h) > \text{error}_D(h')
  \]
Occam’s Razor

• Why Favor Short Hypotheses?

• Arguments for:
  – Fewer short hypotheses than long ones
    → A short hyp. less likely to fit data by coincidence
    → Longer hyp. that fit data may might be coincidence

• Arguments against:
  – Argument above really uses the fact that hypothesis space is small!!!
  – What is so special about small sets based on the size of each hypothesis?
Consider this split
How to Build Small Trees

Two reasonable approaches:

• **Optimize on the held-out (development) set**
  – If growing the tree larger hurts performance, then stop growing!!!
  – Requires a larger amount of data...

• **Use statistical significance testing**
  – Test if the improvement for any split it likely due to noise
  – If so, don’t do the split!
A Chi Square Test

mpg values: bad good

| maker  | america | 0 | 10 | H( mpg | maker = america ) = 0 |
|--------|---------|---|----|----------------------------|
| asia   | 2       | 5 |    | H( mpg | maker = asia ) = 0.863121 |
| europe | 2       | 2 |    | H( mpg | maker = europe ) = 1      |

H(mpg) = 0.702467 H(mpg|maker) = 0.478183
IG(mpg|maker) = 0.224284

- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we’d have seen data of at least this apparent level of association anyway?

By using a particular kind of chi-square test, the answer is 13.5%

We will not cover Chi Square tests in class. See page 93 of the original ID3 paper [Quinlan, 86].
Using Chi-squared to avoid overfitting

• Build the full decision tree as before
• But when you can grow it no more, start to prune:
  – Beginning at the bottom of the tree, delete splits in which \( p_{\text{chance}} > \text{MaxPchance} \)
  – Continue working your way up until there are no more prunable nodes

\textit{MaxPchance} is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise
Pruning example

- With MaxPchance = 0.05, you will see the following MPG decision tree:

When compared to the unpruned tree
- improved test set accuracy
- worse training accuracy

<table>
<thead>
<tr>
<th>mpg values:</th>
<th>bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinders</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>Predict bad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predict good</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Num Errors</th>
<th>Set Size</th>
<th>Percent Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>Test Set</td>
<td>56</td>
<td>352</td>
</tr>
</tbody>
</table>
MaxPchance

• Technical note: MaxPchance is a regularization parameter that helps us bias towards simpler models

Expected Test set Error

We’ll learn to choose the value of magic parameters like this one later!
Real-Valued inputs

What should we do if some of the inputs are real-valued?

<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</td>
<td>97</td>
<td>75</td>
<td>2265</td>
<td>18.2</td>
<td>77</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>199</td>
<td>90</td>
<td>2648</td>
<td>15</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>121</td>
<td>110</td>
<td>2600</td>
<td>12.8</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>350</td>
<td>175</td>
<td>4100</td>
<td>13</td>
<td>73</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>198</td>
<td>95</td>
<td>3102</td>
<td>16.5</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>108</td>
<td>94</td>
<td>2379</td>
<td>16.5</td>
<td>73</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>113</td>
<td>95</td>
<td>2228</td>
<td>14</td>
<td>71</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>302</td>
<td>139</td>
<td>3570</td>
<td>12.8</td>
<td>78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>120</td>
<td>79</td>
<td>2625</td>
<td>18.6</td>
<td>82</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>455</td>
<td>225</td>
<td>4425</td>
<td>10</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>107</td>
<td>86</td>
<td>2464</td>
<td>15.5</td>
<td>76</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>131</td>
<td>103</td>
<td>2830</td>
<td>15.9</td>
<td>78</td>
<td>europe</td>
</tr>
</tbody>
</table>

*Infinite number of possible split values!!!*

*Finite dataset, only finite number of relevant splits!*
“One branch for each numeric value” idea:

Hopeless: with such high branching factor will shatter the dataset and overfit
Threshold splits

- **Binary tree**: split on attribute $X$ at value $t$
  - One branch: $X < t$
  - Other branch: $X \geq t$

- **Requires small change**
  - Allow repeated splits on same variable
  - How does this compare to “branch on each value” approach?
The set of possible thresholds

• Binary tree, split on attribute X
  – One branch: $X < t$
  – Other branch: $X \geq t$

• Search through possible values of $t$
  – Seems hard!!!

• But only finite number of $t$’s are important
  – Sort data according to $X$ into \{\(x_1, \ldots, x_m\)\}
  – Consider split points of the form \(x_i + (x_{i+1} - x_i)/2\)
Picking the best threshold

• Suppose $X$ is real valued with threshold $t$

• Want $IG(Y|X:t)$: the information gain for $Y$ when testing if $X$ is greater than or less than $t$

• Define:
  • $H(Y|X:t) =$
    
    $$H(Y|X < t) \ P(X < t) + H(Y|X >= t) \ P(X >= t)$$
  • $IG(Y|X:t) = H(Y) - H(Y|X:t)$
  • $IG^*(Y|X) = \max_t IG(Y|X:t)$

• Use: $IG^*(Y|X)$ for continuous variables
<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinders</td>
<td>&lt; 5</td>
<td></td>
<td>0.48268</td>
</tr>
<tr>
<td></td>
<td>&gt;= 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displacement</td>
<td>&lt; 198</td>
<td></td>
<td>0.428205</td>
</tr>
<tr>
<td></td>
<td>&gt;= 198</td>
<td></td>
<td></td>
</tr>
<tr>
<td>horsepower</td>
<td>&lt; 94</td>
<td></td>
<td>0.48268</td>
</tr>
<tr>
<td></td>
<td>&gt;= 94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>&lt; 2789</td>
<td></td>
<td>0.379471</td>
</tr>
<tr>
<td></td>
<td>&gt;= 2789</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acceleration</td>
<td>&lt; 18.2</td>
<td></td>
<td>0.159982</td>
</tr>
<tr>
<td></td>
<td>&gt;= 18.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modelyear</td>
<td>&lt; 81</td>
<td></td>
<td>0.319193</td>
</tr>
<tr>
<td></td>
<td>&gt;= 81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maker</td>
<td>america</td>
<td></td>
<td>0.0437265</td>
</tr>
<tr>
<td></td>
<td>asia</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>europe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example with MPG
Example tree for our continuous dataset
What you need to know about decision trees

• Decision trees are one of the most popular ML tools
  – Easy to understand, implement, and use
  – Computationally cheap (to solve heuristically)
• Information gain to select attributes (ID3, C4.5,...)
• Presented for classification, can be used for regression and density estimation too
• Decision trees will overfit!!!
  – Must use tricks to find “simple trees”, e.g.,
    • Fixed depth/Early stopping
    • Pruning
    • Hypothesis testing
Acknowledgements

• Some of the material in the decision trees presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:
  – [http://www.cs.cmu.edu/~awm/tutorials](http://www.cs.cmu.edu/~awm/tutorials)