Decision Trees: many possible refs., e.g., Mitchell, Chapter 3
Boosting: (Linked from class website) Schapire ’01

Decision Trees
Boosting

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

February 6th, 2006
Announcements

- Recitations stay on Thursdays
  - 5-6:30pm in Wean 5409
  - This week: Decision Trees and Boosting

- Pittsburgh won the Super Bowl !!
A dataset is **linearly separable** iff \( \exists \) a separating hyperplane:

- \( \exists \mathbf{w} \), such that:
  - \( w_0 + \sum_i w_i x_i > 0 \); if \( \mathbf{x} = \{x_1, \ldots, x_n\} \) is a positive example
  - \( w_0 + \sum_i w_i x_i < 0 \); if \( \mathbf{x} = \{x_1, \ldots, x_n\} \) is a negative example
Not linearly separable data

- Some datasets are **not linearly separable**!
Addressing non-linearly separable data – Option 1, non-linear features

Choose non-linear features, e.g.,

- Typical linear features: \( w_0 + \sum_i w_i x_i \)
- Example of non-linear features:
  - Degree 2 polynomials, \( w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j \)

Classifier \( h_w(x) \) still linear in parameters \( w \)

- Usually easy to learn (closed-form or convex/concave optimization)
- Data is linearly separable in higher dimensional spaces
- More discussion later this semester
Addressing non-linearly separable data – Option 2, non-linear classifier

- Choose a classifier $h_w(x)$ that is non-linear in parameters $w$, e.g.,
  - Decision trees, neural networks, nearest neighbor,…
- More general than linear classifiers
- But, can often be harder to learn (non-convex/concave optimization required)
- But, but, often very useful
- (BTW. Later this semester, we’ll see that these options are not that different)
A small dataset: Miles Per Gallon

Suppose we want to predict MPG

<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>low</td>
<td>low</td>
<td>high</td>
<td>75to78</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</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</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</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</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</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>70to74</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</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</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</td>
<td>high</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
<td></td>
</tr>
</tbody>
</table>

From the UCI repository (thanks to Ross Quinlan)
A Decision Stump

mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad

<table>
<thead>
<tr>
<th>cylinders = 4</th>
<th>cylinders = 5</th>
<th>cylinders = 6</th>
<th>cylinders = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 17</td>
<td>1 0</td>
<td>8 0</td>
<td>9 1</td>
</tr>
<tr>
<td>Predict good</td>
<td>Predict bad</td>
<td>Predict bad</td>
<td>Predict bad</td>
</tr>
</tbody>
</table>

bad gasmikje
Recursion Step

Take the Original Dataset..

And partition it according to the value of the attribute we split on.
Recursion Step

```
mpg values:   bad    good

root
22    18
p chance = 0.001

cylinders = 3   cylinders = 4   cylinders = 5   cylinders = 6   cylinders = 8
0   0   4   17   1   0   8   0   9   1
Predict bad   Predict good   Predict bad   Predict bad   Predict bad

Build tree from These records..
Build tree from These records..
Build tree from These records..
Build tree from These records..

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8
```
Second level of tree

Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)
mpg values: bad  good

The final tree

©2006 Carlos Guestrin
Classification of a new example

Classifying a test example – traverse tree and report leaf label

\(<C_y=4, \text{Europe}, \text{acc}= \text{low}> = \text{predict bad}\)
Are all decision trees equal?

- Many trees can represent the same concept.
- But, not all trees will have the same size!
  - e.g., $\phi = A \land B \lor \neg A \land C$ ((A and B) or (not A and C))
Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recurse
Choosing a good attribute

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

X₁ \(\text{t} \lor \text{f}\)

\(\frac{4\times T, 0\times F}{3\times F, 1\times T}\)

X₂ \(\text{t} \lor \text{f}\)

\(\frac{3\times T, 1\times F}{2\times T, 2\times F}\)
Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad

\[
\begin{array}{cccc}
P(Y=A) &=& 1/2 & \quad P(Y=B) = 1/4 & \quad P(Y=C) = 1/8 & \quad P(Y=D) = 1/8 \\
\end{array}
\]

- better
  - Less uncertainty

\[
\begin{array}{cccc}
P(Y=A) &=& 1/4 & \quad P(Y=B) = 1/4 & \quad P(Y=C) = 1/4 & \quad P(Y=D) = 1/4 \\
\end{array}
\]

- worse
  - More uncertainty
Entropy

Entropy $H(X)$ 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).
Andrew Moore’s Entropy in a nutshell

Low Entropy

High Entropy
Andrew Moore’s Entropy in a nutshell

Low Entropy

..the values (locations of soup) sampled entirely from within the soup bowl

High Entropy

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room

©2006 Carlos Guestrin
Information gain

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before you split: $H(Y)$
  - Entropy after split on X: $H(Y|X)$
    - Weight by probability of following each branch, i.e., normalized number of records
      \[
      H(Y \mid X) = - \sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)
      \]

- Information gain is difference
  \[
  IG(X) = H(Y) - H(Y \mid X)
  \]
  \[
  \text{BTW. } H(Y) - H(Y \mid X) \geq 0 \text{ (information never lost!)}
  \]

\[
\begin{array}{c|c|c|c}
X_1 & X_2 & Y \\
\hline
T & T & T \\
T & F & T \\
T & T & T \\
T & F & T \\
F & T & T \\
F & F & F \\
\end{array}
\]
Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on \( \arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y \mid X_i) \)
- Recurse
### Information Gain Example

<table>
<thead>
<tr>
<th>wealth values:</th>
<th>poor</th>
<th>rich</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>Female</td>
<td>14423</td>
</tr>
<tr>
<td>Male</td>
<td>22732</td>
<td>9918</td>
</tr>
</tbody>
</table>

- \( H(\text{wealth} | \text{gender} = \text{Female}) = 0.497654 \)
- \( H(\text{wealth} | \text{gender} = \text{Male}) = 0.885847 \)

- \( H(\text{wealth}) = 0.793844 \)
- \( H(\text{wealth}|\text{gender}) = 0.757154 \)
- \( IG(\text{wealth}|\text{gender}) = 0.0366896 \)
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
Don’t split a node if all matching records have the same output value.
Base Case
Two

Don’t split a node if none of the attributes can create multiple non-empty children
Base Case Two: No attributes can distinguish

Information gains using the training set (2 records)

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinders</td>
<td>3</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>displacement</td>
<td>low</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>horsepower</td>
<td>low</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>weight</td>
<td>low</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>acceleration</td>
<td>low</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>medium</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>modelyear</td>
<td>70to74</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>75to78</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>79to83</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>maker</td>
<td>america</td>
<td></td>
<td>0</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>
Base Cases

- 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
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

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

\( y = a \text{ XOR } b \)

The information gains:

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The resulting decision tree:

\( y \text{ values: } 0 \quad 1 \)

root

2 2

Predict 0
If we omit Base Case 3:

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

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

The resulting decision tree:
Basic Decision Tree Building
Summarized

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 and return a non-leaf node with $n_X$ children.
  - The $i$'th child should be built by calling $\text{BuildTree}(\text{DS}_i, \text{Output})$
    Where $\text{DS}_i$ built consists of all those records in $\text{DataSet}$ for which $X = i$th distinct value of $X$. 

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!

Idea One: Branch on each possible real value
“One branch for each numeric value” idea:

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

- Binary tree, split on attribute $X$
  - One branch: $X < t$
  - Other branch: $X \geq t$
Choosing threshold split

- 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 $x_i + (x_{i+1} - x_i)/2$
A better idea: thresholded splits

Suppose X is real valued

Define $IG(Y|X:t)$ as $H(Y) - H(Y|X:t)$

Define $H(Y|X:t) =$

\[ H(Y|X < t) \cdot P(X < t) + H(Y|X \geq t) \cdot P(X \geq t) \]

$IG(Y|X:t)$ is the information gain for predicting Y if all you know is whether X is greater than or less than t

Then define $IG^*(Y|X) = \max_t IG(Y|X:t)$

For each real-valued attribute, use $IG^*(Y|X)$ for assessing its suitability as a split
Information gains using the training set (40 records)

<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 using reals
MPG Test set error

Num Errors  Set Size  Percent Wrong
Training Set  1  40  2.50
Test Set      74  352  21.02

why?
overfitting!

horsepower = low  
0  4
Predict good
pchance = 0.894

horsepower = medium
2  1
Predict bad
pchance = 0.894

horsepower = high
0  0
Predict bad

acceleration = low
1  0
Predict bad

acceleration = medium
1  1
Predict bad
(unexpandable)

acceleration = high
0  0
Predict bad

modelyear = 70to74
0  0
Predict bad

modelyear = 75to78
1  0
Predict good

modelyear = 79to83
0  0
Predict bad
The test set error is much worse than the training set error...

...why?
Decision trees & Learning Bias

Any "separable" data

G no examples with
same attrib. values
have diff. labels
can classified exactly with
a d. tree

⇒ (almost) zero bias!
Decision trees will overfit

- Standard decision trees are have no learning biased
  - Training set error is always zero!
  - Lots of variance
  - Will definitely overfit!!!
  - Must bias towards simpler trees

- Many strategies for picking simpler trees:
  - Fixed depth
  - Fixed number of leaves
  - Or something smarter…
Consider this split
A chi-square test

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?

<table>
<thead>
<tr>
<th>mpg values:</th>
<th>bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>maker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>america</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>asia</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>europe</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>H( mpg</td>
<td>maker = america ) = 0</td>
<td></td>
</tr>
<tr>
<td>H( mpg</td>
<td>maker = asia ) = 0.863121</td>
<td></td>
</tr>
<tr>
<td>H( mpg</td>
<td>maker = europe ) = 1</td>
<td></td>
</tr>
<tr>
<td>H(mpg) = 0.702467</td>
<td>H(mpg</td>
<td>maker) = 0.478183</td>
</tr>
</tbody>
</table>
| \[ \text{IG}(\text{mpg}|\text{maker}) = 0.224284 \]
A chi-square test

<table>
<thead>
<tr>
<th>maker</th>
<th>america</th>
<th>0</th>
<th>10</th>
<th>mpg values: bad</th>
<th>good</th>
</tr>
</thead>
<tbody>
<tr>
<td>asia</td>
<td>2</td>
<td>5</td>
<td></td>
<td>H( mpg</td>
<td>maker = america ) = 0</td>
</tr>
<tr>
<td>europe</td>
<td>2</td>
<td>2</td>
<td></td>
<td>H( mpg</td>
<td>maker = asia ) = 0.863121</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H( mpg</td>
<td>maker = europe ) = 1</td>
</tr>
</tbody>
</table>

| H(mpg)       | 0.702467 |
| H(mpg|maker)    | 0.478183 |
| I(G(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%

(Such simple hypothesis tests are very easy to compute, unfortunately, not enough time to cover in the lecture)
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 you way up until there are no more prunable nodes

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

With MaxPchance = 0.1, you will see the following MPG decision tree:

- mpg values: bad good
- root
- 22 18
- pchance = 0.001

- cylinders = 3
  - Predict bad
  - 0 0
- cylinders = 4
  - Predict good
  - 4 17
- cylinders = 5
  - Predict bad
  - 1 0
- cylinders = 6
  - Predict bad
  - 8 0
- cylinders = 8
  - Predict bad
  - 9 1

Note the improved test set accuracy compared with the unpruned tree.

<table>
<thead>
<tr>
<th></th>
<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>
<td>12.50 higher</td>
</tr>
<tr>
<td>Test Set</td>
<td>56</td>
<td>352</td>
<td>15.91 lower</td>
</tr>
</tbody>
</table>

©2006 Carlos Guestrin
MaxPchance

- Technical note MaxPchance is a regularization parameter that helps us bias towards simpler models.

We’ll learn to choose the value of these magic parameters soon!
What you need to know about decision trees

- Decision trees are one of the most popular data mining tools
  - Easy to understand
  - Easy to implement
  - Easy to 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!!!
  - Zero bias classifier → Lots of variance
  - Must use tricks to find “simple trees”, e.g.,
    - Fixed depth/Early stopping
    - Pruning
    - Hypothesis testing
Fighting the bias-variance tradeoff

- Simple (a.k.a. weak) learners are good
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - Low variance, don’t usually overfit

- Simple (a.k.a. weak) learners are bad
  - High bias, can’t solve hard learning problems

- Can we make weak learners always good???
  - No!!
  - But often yes…
Boosting  [Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (rewighted) training data, then let learned classifiers vote

- On each iteration $t$:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – $h_t$
  - A strength for this hypothesis – $\alpha_t$

- Final classifier:

- Practically useful
- Theoretically interesting
Learning from weighted data

- Sometimes not all data points are equal
  - Some data points are more equal than others

- Consider a weighted dataset
  - $D(i)$ – weight of $i$th training example $(x^i, y^i)$

- Now, in all calculations, whenever used, $i$th training example counts as $D(i)$ “examples”
  - e.g., MLE for Naïve Bayes, redefine $Count(Y=y)$ to be weighted count
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in Y = \{-1, +1\}\)
Initialize \(D_1(i) = 1/m\).
For \(t = 1, \ldots, T\):

- Train base learner using distribution \(D_t\).
- Get base classifier \(h_t : X \rightarrow \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:
  \[
  D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
  \]
  where \(Z_t\) is a normalization factor
  \[
  Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
  \]
- Output the final classifier:
  \[
  H(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right).
  \]

Figure 1: The boosting algorithm AdaBoost.
Given: \((x_1, y_1), \ldots, (x_m, y_m)\) where \(x_i \in X, y_i \in \{\pm 1\}\)

Initialize \(D_1(i) = 1/m\).

For \(t = 1, \ldots, T\):

- Train base learner using distribution \(D_t\).
- Get base classifier \(h_t : X \to \mathbb{R}\).
- Choose \(\alpha_t \in \mathbb{R}\).
- Update:

\[
\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)
\]

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

\[
\epsilon_t = P_{i \sim D_i}[x_i \neq y_i]
\]

\[
\epsilon_t = \frac{1}{\sum_{i=1}^n D_t(i)} \sum_{i=1}^m D_t(i) \delta(h(x_i) \neq y_i)
\]
What $\alpha_t$ to choose for hypothesis $h_t$?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i))$$

Where $f(x) = \sum_t \alpha_t h_t(x); H(x) = \text{sign}(f(x))$
What $\alpha_t$ to choose for hypothesis $h_t$?

Training error of final classifier is bounded by:

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_{t} Z_t$$

Where $f(x) = \sum_{t} \alpha_t h_t(x); H(x) = \text{sign}(f(x))$
What $\alpha_t$ to choose for hypothesis $h_t$?

Training error of final classifier is bounded by:

$$
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_i \exp(-y_if(x_i)) = \prod_t Z_t
$$

Where $f(x) = \sum_t \alpha_th_t(x); H(x) = \text{sign}(f(x))$

If we minimize $\prod_t Z_t$, we minimize our training error.

We can tighten this bound by choosing $\alpha_t$ and $h_t$ on each iteration to minimize $Z_t$.

$$
Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha ty_ih_t(x_i))
$$
What $\alpha_t$ to choose for hypothesis $h_t$?

We can minimize this bound by choosing $\alpha_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire ’97]:

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$$
Strong, weak classifiers

- If each classifier is (at least slightly) better than random
  \[ \varepsilon_t < 0.5 \]

- AdaBoost will achieve zero *training error* (exponentially fast):

\[
\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \prod_{t} Z_t \leq \exp \left( -2 \sum_{t=1}^{T} (1/2 - \varepsilon_t)^2 \right)
\]

- Is it hard to achieve better than random training error?
Comparison of C4.5, Boosting C4.5, Boosting decision stumps (depth 1 trees), 27 benchmark datasets
AdaBoost and AdaBoost.MH on Train (left) and Test (right) data from Irvine repository. [Schapire and Singer, ML 1999]
Boosting and Logistic Regression

Logistic regression assumes:

\[ P(Y = 1|X) = \frac{1}{1 + \exp(f(x))} \]

And tries to maximize data likelihood:

\[ P(data|H) = \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i f(x_i))} \]

Equivalent to minimizing log loss

\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i))) \]
Boosting and Logistic Regression

Logistic regression equivalent to minimizing log loss

\[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]

Boosting minimizes similar loss function!!

\[ \frac{1}{m} \sum_i \exp(-y_if(x_i)) = \prod_t Z_t \]

Both smooth approximations of 0/1 loss!
Logistic regression and Boosting

Logistic regression:
- Minimize loss fn
  \[ \sum_{i=1}^{m} \ln(1 + \exp(-y_if(x_i))) \]
- Define
  \[ f(x) = \sum_{j} w_j x_j \]
  where \( x_j \) predefined

Boosting:
- Minimize loss fn
  \[ \sum_{i=1}^{m} \exp(-y_if(x_i)) \]
- Define
  \[ f(x) = \sum_{t} \alpha_t h_t(x) \]
  where \( h(x_i) \) defined dynamically to fit data
- Weights \( \alpha_j \) learned incrementally
What you need to know about Boosting

- Combine weak classifiers to obtain very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error

- AdaBoost algorithm

- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)

- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier
Acknowledgements

- Much of the decision trees material in the 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)

- Much of the boosting material in the presentation is courtesy of Tom Mitchell