Linear separability

A dataset is **linearly separable** iff \( \exists \) a separating hyperplane:

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

- Some datasets are **not linearly separable!**

\[
\text{XOR}
\]

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 \( w \)
  - As easy to learn
  - Data is linearly separable in higher dimensional spaces
  - More discussion later this semester

\[
\begin{align*}
 f_w(x) &= w_0 + w_1 x_1 + w_2 x_2 \\
 f_w(x) &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 \\
 f_w(x) &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 \geq 0 \\
 f_w(x) &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 < 0
\end{align*}
\]
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

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

Recursion Step

Take the Original Dataset...

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

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8
Build tree from These records..
Build tree from These records..
Build tree from These records..
Build tree from These records..

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)
Classification of a new example

- Classifying a test example – traverse tree and report leaf label

  - Cylinders: 4
  - Maker: Asian
  - Horsepower: Medium
  - Age: Medium
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

Measuring uncertainty

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

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

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

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

Information gain

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before you split
    \[ H(Y) = -\sum_{k} p(Y=k) \log p(Y=k) \]
  - Entropy after split
    \[ H(Y|X) = -\sum_{j} \sum_{k} p(Y=k|X=x_j) p(X=x_j) \log p(Y=k|X=x_j) \]
  - Weight by probability of following each branch, i.e., normalized number of records
    \[ \text{Information gain is difference} \quad IG(X) = H(Y) - H(Y|X) \]

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</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>
</tbody>
</table>

©2005-2007 Carlos Guestrin
Learning decision trees

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

![Decision Tree Diagram]

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: Predict bad
- cylinders = 4: Predict good
- cylinders = 5: Predict bad
- cylinders = 6: Predict bad
- cylinders = 8: Predict bad

Don't split a node if all matching records have the same output value

Base Case
One
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
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

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

\[y = a \ XOR \ b\]

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>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The resulting decision tree:

y values: 0 1

root

2 2

Predict 0

If we omit Base Case 3:

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

\[y = a \ XOR \ b\]

The resulting decision tree:

y values: 0 1

root

2 2

p\text{chance} = 0.414

Predict 0, Predict 1
Basic Decision Tree Building Summarized

BuildTree(DataSet, Output)
- If all output values are the same in 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 BuildTree($DS_i$, Output)
    - Where $DS_i$ built consists of all those records in DataSet for which $X = i$th distinct value of $X$.

Announcements

- Pittsburgh won the Super Bowl !!
The test set error is much worse than the training set error.

...why?
Decision trees & Learning Bias

- Standard decision trees are have no learning biased
  - Training set error is always zero!
    - (If there is no label noise)
  - 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…
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?
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? By using a particular kind of chi-square test, the answer is 7.2%.

(Such simple hypothesis tests are very easy to compute, unfortunately, not enough time to cover in the lecture, but in your homework, you’ll have fun! :))

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:

![Decision Tree Diagram](image)

Note the improved test set accuracy compared with the unpruned tree

<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

![Graph](image)

- We’ll learn to choose the value of these magic parameters soon!
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>model year</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>198</td>
<td>90</td>
<td>2648</td>
<td>16</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>123</td>
<td>110</td>
<td>2880</td>
<td>12.8</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>360</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>6</td>
<td>302</td>
<td>139</td>
<td>3070</td>
<td>12.8</td>
<td>78</td>
<td>america</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
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 $H(Y|X:t) = H(Y|X < t) P(X < t) + H(Y|X \geq t) P(X \geq t)$
  - $IG(Y|X:t)$ 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

Example with MPG

<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>0.48286</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 5</td>
<td>0.426205</td>
<td></td>
</tr>
<tr>
<td>displacement</td>
<td>&lt; 108</td>
<td>0.48288</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 108</td>
<td>0.379471</td>
<td></td>
</tr>
<tr>
<td>horsepower</td>
<td>&lt; 94</td>
<td>0.48288</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 94</td>
<td>0.15982</td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>&lt; 2700</td>
<td>0.319163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 2700</td>
<td>0.0437266</td>
<td></td>
</tr>
<tr>
<td>acceleration</td>
<td>&lt; 19.2</td>
<td>0.319163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 19.2</td>
<td>0.0437266</td>
<td></td>
</tr>
<tr>
<td>modelyear</td>
<td>&lt; 81</td>
<td>0.319163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\geq 81</td>
<td>0.0437266</td>
<td></td>
</tr>
<tr>
<td>maker</td>
<td>america</td>
<td>0.319163</td>
<td></td>
</tr>
<tr>
<td></td>
<td>asia</td>
<td>0.0437266</td>
<td></td>
</tr>
<tr>
<td></td>
<td>europe</td>
<td>0.0437266</td>
<td></td>
</tr>
</tbody>
</table>
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
Acknowledgements

- Some of the 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)