Linear separability

- A dataset is **linearly separable** iff there exists a **separating hyperplane**:
  - Exists \( w \), such that:
    - \( w_0 + \sum_i w_i x_i > 0 \); if \( x = (x_1, \ldots, x_k) \) is a positive example
    - \( w_0 + \sum_i w_i x_i < 0 \); if \( x = (x_1, \ldots, x_k) \) 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, $w_0 + \sum_i w_i x_i + \sum_{i,j} w_{ij} x_i x_j$
- 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 quarter
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, boosting, nearest neighbor, neural networks…
- 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 quarter, 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>high</td>
<td>785.78</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>785.83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>low</td>
<td>785.83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>785.83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>704.74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>low</td>
<td>785.83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>785.83</td>
<td>europe</td>
</tr>
</tbody>
</table>

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

Examples in which cylinders = 4

Examples in which cylinders = 5

Examples in which cylinders = 6

Examples in which cylinders = 8
Recursion Step

Build tree from These examples.. Build tree from These examples.. Build tree from These examples.. Build tree from These examples..

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)
Classification of a new example

- Classifying a test example – traverse tree and report leaf label
Are all decision trees equal?

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

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>$X_1$</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>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Measuring uncertainty

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

$P(Y=A) = 1/2$  $P(Y=B) = 1/4$  $P(Y=C) = 1/8$  $P(Y=D) = 1/8$

$P(Y=A) = 1/4$  $P(Y=B) = 1/4$  $P(Y=C) = 1/4$  $P(Y=D) = 1/4$
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

High Entropy

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

..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
  - Entropy after split
    - Weight by probability of following each branch, i.e., normalized number of records

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

- Information gain is difference

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

Suppose we want to predict MPG

Look at all the information gains…
A Decision Stump

mpg values: bad good

root
22 18
pchange = 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

Base Case
One

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

- 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

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

\[Y = A \text{ XOR } B\]

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>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

The resulting bad decision tree:

y values: 0 1

root

2 2

Predict 0

If we omit Base Case 3:

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

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

The resulting decision tree:

y values: 0 1

root

2 2

Predict 0  Predict 1  Predict 1  Predict 0
Basic Decision Tree Building
Summarized

BuildTree(\textit{DataSet, Output})

- If all output values are the same in \textit{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 \(\text{BuildTree(\textit{DS}_i, \textit{Output})}\)
      Where \(DS_i\) built consists of all those records in \textit{DataSet} for which \(X = i\)th distinct value of \(X\).

MPG Test set error

<table>
<thead>
<tr>
<th>Num Errors</th>
<th>Set Size</th>
<th>Percent Wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Test Set</td>
<td>74</td>
<td>352</td>
</tr>
</tbody>
</table>
The test set error is much worse than the training set error...

...why?

**Decision trees & Learning Bias**
Decision trees will overfit

- Standard decision trees are have no learning bias
  - 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…

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?

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 see readings...)
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

\( \text{MaxPchance} \) 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:

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</td>
</tr>
<tr>
<td>Test Set</td>
<td>56</td>
<td>352</td>
<td>15.91</td>
</tr>
</tbody>
</table>

©Carlos Guestrin 2005-2013
MaxPchance

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

Expected True Error

- Decreasing MaxPchance
- Increasing

High Bias

High Variance

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>75</td>
<td>2265</td>
<td>18.2</td>
<td>71</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>106</td>
<td>90</td>
<td>2648</td>
<td>15</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>121</td>
<td>110</td>
<td>2600</td>
<td>12.8</td>
<td>71</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>383</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>106</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>
</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 ≥ 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 \(H(Y|X:t) = H(Y|X < t) P(X < t) + H(Y|X >= t) P(X >= t)\)
  - \(IG(Y|X:t)\) 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
- Note, may split on an attribute multiple times, with different thresholds
Example with MPG

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; 106</td>
<td>[ ]</td>
<td>0.428265</td>
</tr>
<tr>
<td></td>
<td>&gt;= 106</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; 2769</td>
<td>[ ]</td>
<td>0.379471</td>
</tr>
<tr>
<td></td>
<td>&gt;= 2769</td>
<td>[ ]</td>
<td></td>
</tr>
<tr>
<td>acceleration</td>
<td>&lt; 18.2</td>
<td>[ ]</td>
<td>0.169862</td>
</tr>
<tr>
<td></td>
<td>&gt;= 18.2</td>
<td>[ ]</td>
<td></td>
</tr>
<tr>
<td>model year</td>
<td>&lt; 81</td>
<td>[ ]</td>
<td>0.318193</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 tree using reals
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 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)