Linear separability

- A dataset is **linearly separable** iff there exists a **separating hyperplane**:
  - Exists \( \mathbf{w} \), 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

©Carlos Guestrin 2005-2013
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

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

Recursion Step
Recursion Step

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)
The final tree

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., \( \phi = A \land B \lor \neg A \land C \) ((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

```
  X1  X2  Y
  T   T  T
  T   F  T
  F   T  T
  F   F  T
```

"Certainty" is good!

Measuring uncertainty

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

\[
P(Y=A) = 1/2 \quad P(Y=B) = 1/4 \quad P(Y=C) = 1/8 \quad P(Y=D) = 1/8
\]

\[
P(Y=A) = 1/4 \quad P(Y=B) = 1/4 \quad P(Y=C) = 1/4 \quad P(Y=D) = 1/4
\]
Entropy

Entropy $H(Y)$ of a random variable $Y = \{y_1, y_2, \ldots, y_k\}$

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

*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) unpredictable...
- almost uniformly sampled throughout our dining room

High Entropy

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

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
    - Information gain is difference

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

\[
IG(Y_i) = 0.65 - \frac{1}{3} = 0.32
\]
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

When do you stop?
1) entropy is 0
2) cannot split
3) when \( IG \) is 0

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
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

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

The resulting bad decision tree:

y values: 0 1
root

If we omit Base Case 3:

y = a XOR b

The resulting decision tree:
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 $X$.

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

\[ h(k) \text{ bias under low high low high} \]