Today’s lecture

- Information Gain for measuring association between inputs and outputs
- Learning a decision tree classifier from data

Deciding whether a pattern is interesting

- We will use information theory
- A very large topic, originally used for compressing signals
- But more recently used for data mining...

(The topic of Information Gain will now be discussed, but you will find it in a separate Andrew Handout)

Bits

You are watching a set of independent random samples of X
You see that X has four possible values

\[
\begin{align*}
P(X=A) &= \frac{1}{4} \\
P(X=B) &= \frac{1}{4} \\
P(X=C) &= \frac{1}{4} \\
P(X=D) &= \frac{1}{4}
\end{align*}
\]

So you might see: BAACBADCDADDDA...
You transmit data over a binary serial link. You can encode each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

01000100100111010011111100...
Fewer Bits

Someone tells you that the probabilities are not equal

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

It’s possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

(Possible coding)

\[
\begin{array}{c|c}
A & 00 \\
B & 01 \\
C & 10 \\
D & 111 \\
\end{array}
\]

Fewer Bits

Suppose there are three equally likely values...

\[
P(X=B) = 1/3 \quad P(X=C) = 1/3 \quad P(X=D) = 1/3
\]

Here’s a naïve coding, costing 2 bits per symbol

\[
\begin{array}{c|c}
A & 00 \\
B & 01 \\
C & 10 \\
\end{array}
\]

Can you think of a coding that would need only 1.6 bits per symbol on average?

In theory, it can in fact be done with 1.58496 bits per symbol.

General Case

Suppose \(X\) can have one of \(m\) values... \(V_1, V_2, \ldots, V_m\)

\[
\begin{array}{c|c|c|c}
P(X=V_1) = p_1 & P(X=V_2) = p_2 & \ldots & P(X=V_m) = p_m \\
\end{array}
\]

What’s the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from \(X\)'s distribution? It’s

\[
H(X) = -\sum_{j=1}^{m} p_j \log_2 p_j
\]

- \"High Entropy\" means \(X\) is from a uniform (boring) distribution
- \"Low Entropy\" means \(X\) is from varied (peaks and valleys) distribution

A histogram of the frequency distribution of values of \(X\) would have many lows and one or two highs

A histogram of the frequency distribution of values of \(X\) would be flat

...and so the values sampled from it would be all over the place

...and so the values sampled from it would be more predictable
Entropy in a nut-shell

Low Entropy High Entropy

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

Specific Conditional Entropy \( H(Y|X=v) \)

Suppose I'm trying to predict output \( Y \) and I have input \( X \)

\[ X = \text{College Major} \]
\[ Y = \text{Likes "Gladiator"} \]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Let's assume this reflects the true probabilities

E.g. From this data we estimate

- \( P(\text{LikeG} = \text{Yes}) = 0.5 \)
- \( P(\text{Major} = \text{Math} & \text{LikeG} = \text{No}) = 0.25 \)
- \( P(\text{Major} = \text{Math}) = 0.5 \)
- \( P(\text{LikeG} = \text{Yes} | \text{Major} = \text{History}) = 0 \)

Note:

- \( H(X) = 1.5 \)
- \( H(Y) = 1 \)

Specific Conditional Entropy \( H(Y|X=v) \)

Definition of Specific Conditional Entropy:

\[ H(Y|X=v) = \text{The entropy of } Y \text{ among only those records in which } X \text{ has value } v \]

X = College Major
Y = Likes "Gladiator"

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example:

- \( H(Y|X=\text{Math}) = 1 \)
- \( H(Y|X=\text{History}) = 0 \)
- \( H(Y|X=\text{CS}) = 0 \)

Conditional Entropy \( H(Y|X) \)

Definition of Conditional Entropy:

\[ H(Y|X) = \text{The average specific conditional entropy of } Y \]

= if you choose a record at random what will be the conditional entropy of \( Y \), conditioned on that row's value of \( X \)

= Expected number of bits to transmit \( Y \) if both sides will know the value of \( X \)

\[ = \sum_j P(X=v_j) \cdot H(Y | X = v_j) \]
### Conditional Entropy

**Definition of Conditional Entropy:**

\[ H(Y|X) = \sum_{v_j} \text{Prob}(X=v_j) \cdot H(Y | X = v_j) \]

**Example:**

| \(v_j\) | \(\text{Prob}(X=v_j)\) | \(H(Y | X = v_j)\) |
|---|---|---|
| Math | 0.5 | 1 |
| History | 0.25 | 0 |
| CS | 0.25 | 0 |

\[ H(Y, X) = 0.5 \times 1 + 0.25 \times 0 + 0.25 \times 0 = 0.5 \]

### Information Gain

**Definition of Information Gain:**

\[ IG(Y | X) = \text{I must transmit } Y \] How many bits on average would it save me if both ends of the line knew \(X\)?

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

**Example:**

- \(H(Y) = 1\)
- \(H(Y | X) = 0.5\)
- Thus \(IG(Y | X) = 1 - 0.5 = 0.5\)

### Learning Decision Trees

- A Decision Tree is a tree-structured plan of a set of attributes to test in order to predict the output.
- To decide which attribute should be tested first, simply find the one with the highest information gain.
- Then recurse...

### A small dataset: Miles Per Gallon

<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>low</td>
<td>high</td>
<td>75 to 78</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>70 to 74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75 to 78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70 to 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>70 to 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>70 to 74</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75 to 78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70 to 74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>8</td>
<td>high</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>79 to 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>75 to 78</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>79 to 83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>75 to 78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>79 to 83</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79 to 83</td>
<td>america</td>
<td></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

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8

Recurse again on the seven records in which there are 4 cylinders and the maker was based in Asia
(Similar recursion in the other cases)

Recursion Step

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
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

The final tree

Second level of tree

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Records in which cylinders = 4
Predic good
Predic good
Predic good
Predic good

Base Case

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 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

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

The information gains:

The resulting decision tree:

\[ y \text{ values: } \begin{cases} 0 & 1 \\ \text{root} & 2 \\ 2 & 2 \\ \text{Predict } 0 \end{cases} \]

If we omit Base Case 3:

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

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 nX distinct values (i.e. X has arity nX).
• The i'th child should be built by calling BuildTree(DSi, Output)
  Where DS_i built consists of all those records in DataSet for which X = i' th distinct value of X.

Training Set Error
• For each record, follow the decision tree to see what it would predict
  For what number of records does the decision tree’s prediction disagree with the true value in the database?
• This quantity is called the training set error. The smaller the better.

Stop and reflect: Why are we doing this learning anyway?
• It is not usually in order to predict the training data’s output on data we have already seen.
Stop and reflect: Why are we doing this learning anyway?

- It is not usually in order to predict the training data’s output on data we have already seen.
- It is more commonly in order to predict the output value for future data we have not yet seen.

Warning: A common data mining misperception is that the above two bullets are the only possible reasons for learning. There are at least a dozen others.

Test Set Error

- Suppose we are forward thinking.
- We hide some data away when we learn the decision tree.
- But once learned, we see how well the tree predicts that data.
- This is a good simulation of what happens when we try to predict future data.
- And it is called Test Set Error.

The test set error is much worse than the training set error...

...why?

An artificial example

- We’ll create a training dataset
  - Five inputs, all bits, are generated in all 32 possible combinations
  - Output y = copy of e, Except a random 25% of the records have y set to the opposite of e
In our artificial example
• Suppose someone generates a test set according to the same method.
• The test set is identical, except that some of the y’s will be different.
• Some y’s that were corrupted in the training set will be uncorrupted in the testing set.
• Some y’s that were uncorrupted in the training set will be corrupted in the test set.

Training set error for our artificial tree
All the leaf nodes contain exactly one record and so...
• We would have a training set error of zero

Testing the tree with the test set
<table>
<thead>
<tr>
<th>1/4 of the test set records are corrupted</th>
<th>3/4 are fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/16 of the test set will be correctly predicted for the wrong reasons</td>
<td></td>
</tr>
<tr>
<td>3/16 of the test set will be wrongly predicted because the test record is corrupted</td>
<td></td>
</tr>
<tr>
<td>3/16 of the test predictions will be wrong because the tree node is corrupted</td>
<td></td>
</tr>
<tr>
<td>9/16 of the test predictions will be fine</td>
<td></td>
</tr>
</tbody>
</table>

In total, we expect to be wrong on 3/8 of the test set predictions

What’s this example shown us?
• This explains the discrepancy between training and test set error
• But more importantly... ...it indicates there’s something we should do about it if we want to predict well on future data.

Suppose we had less data
• Let’s not look at the irrelevant bits

Output y = copy of e, except a random 25% of the records have y set to the opposite of e

What decision tree would we learn now?
Without access to the irrelevant bits...

- Root
- \( e = 0 \) \( e = 1 \)

These nodes will be unexpandable

Overfitting

- Definition: If your machine learning algorithm fits noise (i.e. pays attention to parts of the data that are irrelevant) it is overfitting.
- Fact (theoretical and empirical): If your machine learning algorithm is overfitting then it may perform less well on test set data.

Avoiding overfitting

- Usually we do not know in advance which are the irrelevant variables
- ...and it may depend on the context
  - For example, if \( y = a \text{ AND } b \) then \( b \) is an irrelevant variable only in the portion of the tree in which \( a = 0 \)

  But we can use simple statistics to warn us that we might be overfitting.

In total, we expect to be wrong on only 1/4 of the test set predictions

In about 12 of the 16 records in this node the output will be 0

So this will almost certainly predict 0

In about 12 of the 16 records in this node the output will be 1

So this will almost certainly predict 1
A chi-squared 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-squared test, the answer is 13.5%.

What is a Chi-Square test?

- Google "chi square" for excellent explanations
- Takes into account "surprise" that a feature generates: \( \sum \frac{(\text{unsplit-number} - \text{split-number})^2}{\text{unsplit-number}} \)
- Gives probability that rate you saw was generated by "luck of the draw"
- Does "likes-Matrix" predict "CS grad"?

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.

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:

<table>
<thead>
<tr>
<th>Class</th>
<th>CS</th>
<th>Non-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likes</td>
<td>15972</td>
<td>145643</td>
</tr>
<tr>
<td>Hates</td>
<td>3</td>
<td>37</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>CS</th>
<th>Non-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likes</td>
<td>21543</td>
<td>145643</td>
</tr>
<tr>
<td>Hates</td>
<td>3</td>
<td>173</td>
</tr>
</tbody>
</table>

Note: The improved test set accuracy compared with the unpruned tree.
MaxPchance

- Technical note (dealt with in other lectures): MaxPchance is a regularization parameter.

The simplest tree

- Note that this pruning is heuristically trying to find
  *The simplest tree structure for which all within-leaf-node disagreements can be explained by chance*
- This is not the same as saying "the simplest classification scheme for which..."
- Decision trees are biased to prefer classifiers that can be expressed as trees.

Expressiveness of Decision Trees

- Assume all inputs are Boolean and all outputs are Boolean.
- What is the class of Boolean functions that are possible to represent by decision trees?
- Answer: All Boolean functions.
  
  **Simple proof:**
  1. Take any Boolean function
  2. Convert it into a truth table
  3. Construct a decision tree in which each row of the truth table corresponds to one path through the decision tree.

Real-Valued inputs

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

<table>
<thead>
<tr>
<th></th>
<th>mpg</th>
<th>cylinders</th>
<th>displacemen</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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>108</td>
<td>94</td>
<td>2379</td>
<td>16.5</td>
<td>71</td>
<td>asia</td>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
</tbody>
</table>

  **Idea One: Branch on each possible real value**

"One branch for each numeric value" idea:

- 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

Hopeless: with such high branching factor will shatter the dataset and over fit

Note pchance is 0.222 in the above...if MaxPchance was 0.05 that would end up pruning away to a single root node.
Computational Issues

- You can compute $\text{IG}^*(Y|X)$ in time $R \log R + 2Rn_Y$

- Where
  
  $R$ is the number of records in the node under consideration
  $n_Y$ is the arity (number of distinct values of) $Y$

How?

Sort records according to increasing values of $X$. Then create a $2 \times n_Y$ contingency table corresponding to computation of $\text{IG}(Y|X:x_{\text{min}})$. Then iterate through the records, testing for each threshold between adjacent values of $X$, incrementally updating the contingency table as you go. For a minor additional speedup, only test between values of $Y$ that differ.

Example with MPG

Unpruned tree using reals

Pruned tree using reals

Binary categorical splits

- One of Andrew’s favorite tricks
- Allow splits of the following form

Example:

Predicting age from census
Conclusions

- Decision trees are the single most popular data mining tool
- Easy to understand
- Easy to implement
- Easy to use
- Computationally cheap
- It’s possible to get in trouble with overfitting
- They do classification: predict a categorical output from categorical and/or real inputs

What you should know

- What’s information gain, and why we use it
- The recursive algorithm for building an unpruned decision tree
- What are training and test set errors
- Why test set errors can be bigger than training set
- Why pruning can reduce test set error
- How to exploit real-valued inputs

What we haven’t discussed

- It’s easy to have real-valued outputs too—these are called Regression Trees*
- Bayesian Decision Trees can take a different approach to preventing overfitting
- Computational complexity (straightforward and cheap) *
- Alternatives to Information Gain for splitting nodes
- How to choose MaxPChance automatically *
- The details of Chi-Squared testing *
- Boosting—a simple way to improve accuracy *

* = discussed in other Andrew lectures

For more information

- Two nice books
  - C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning) by J. Ross Quinlan
- Dozens of nice papers, including
- Dozens of software implementations available on the web for free and commercially for prices ranging between $50 - $300,000
Discussion

• Instead of using information gain, why not choose the splitting attribute to be the one with the highest prediction accuracy?
• Instead of greedily, heuristically, building the tree, why not do a combinatorial search for the optimal tree?
• If you build a decision tree to predict wealth, and marital status, age and gender are chosen as attributes near the top of the tree, is it reasonable to conclude that those three inputs are the major causes of wealth?
• ..would it be reasonable to assume that attributes not mentioned in the tree are not causes of wealth?
• ..would it be reasonable to assume that attributes not mentioned in the tree are not correlated with wealth?
• What about multi-attribute splits?