Decision Trees and Information Gain

Slides are modified based on Prof. Andrew Moore’s tutorial at http://www.cs.cmu.edu/~awm/tutorials

Outline

- Entropy and Information Gain
- Learning an unpruned decision tree recursively
- Training Set Error
- Test Set Error
- Overfitting
- Information Gain of a real valued input
Entropy

• The term is used in a lot of fields

• In information theory,
  Entropy $H(X)$ of a random variable $X$ is...

  ...the expected number of bits needed to encode a randomly drawn value of $X$ (under most efficient code)

Bits

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

<table>
<thead>
<tr>
<th>$P(X=A)$</th>
<th>$P(X=B)$</th>
<th>$P(X=C)$</th>
<th>$P(X=D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

So you might see: BAACBADCDADDDA...

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

01000010010011101100111111100...

Average number of bits = 2
**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 fewer bits on average per symbol.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
</tr>
<tr>
<td>D</td>
<td>111</td>
</tr>
</tbody>
</table>

Avg bits = \(1\times1/2 + 2\times1/4 + 3\times1/8 + 3\times1/8 = 1.75\)

(This is just one of several ways)

---

**General Case**

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

\[
P(X=V_1) = p_1 \quad P(X=V_2) = p_2 \quad \ldots \quad P(X=V_m) = p_m
\]

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

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

\(H(X)\) = The entropy of \(X\)

- “High Entropy” means \(X\) is from a uniform (boring) distribution
- “Low Entropy” means \(X\) is from varied (peaks and valleys) distribution
Recap: Entropy

Low Entropy means sth. is almost certainly true
High Entropy means high uncertainty

If $X$ is a binary variable,
when is $H(X) = 1$?
when is $H(X) = 0$?

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

Suppose I’m trying to predict output $Y$ and I have input $X$

$X =$ 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>

Let’s assume this reflects the true probabilities

E.G. From this data we estimate

* $P(\text{LikeG} = \text{Yes}) = 0.5$
* $P(\text{Major} = \text{Math} \land \text{LikeG} = \text{No}) = 0.25$
* $P(\text{Major} = \text{Math}) = 0.5$
* $P(\text{LikeG} = \text{Yes} \mid \text{Major} = \text{History}) = 0$

Note:

* $H(X) = 1.5$
* $H(Y) = 1$
Specific Conditional Entropy $H(Y|X=v)$

**Definition of Specific Conditional Entropy:**

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

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

<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>
Conditional Entropy

X = College Major
Y = Likes “Gladiator”

**Definition of Conditional Entropy:**

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

\[ = \sum \text{Prob}(X=v_j) \times H(Y \mid X = v_j) \]

**Example:**

<table>
<thead>
<tr>
<th>(v_j)</th>
<th>Prob(X=v_j)</th>
<th>H(Y\mid X = v_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>History</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>CS</td>
<td>0.25</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Information Gain

X = College Major
Y = Likes “Gladiator”

**Definition of Information Gain:**

\[ IG(Y \mid X) = \text{I must transmit } Y. \]

How many bits on average would it save me if both ends of the line knew \(X\)?

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

**Example:**

- \(H(Y) = 1\)
- \(H(Y \mid X) = 0.5\)
- Thus \(IG(Y \mid X) = 1 - 0.5 = 0.5\)
### Information Gain Example

| gender | Female | 14423 | 1769 | H(wealth | gender = Female) = 0.497654 |
|--------|--------|-------|------|----------------------------------|
| Male   | 22732  | 9918  |      | H(wealth | gender = Male) = 0.885847       |

$H(\text{wealth}) = 0.793844$  $H(\text{wealth} | \text{gender}) = 0.757154$

$IG(\text{wealth} | \text{gender}) = 0.0365896$

---

### Another example

| agegroup | 10s | 2507 | 3 | H(wealth | agegroup = 10s) = 0.0133271 |
|----------|-----|------|---|--------------------------------|
| 20s      | 11202 | 743 |   | H(wealth | agegroup = 20s) = 0.334006  |
| 30s      | 9468 | 3461 |   | H(wealth | agegroup = 30s) = 0.838134  |
| 40s      | 6738 | 3965 |   | H(wealth | agegroup = 40s) = 0.951961  |
| 50s      | 4110 | 2507 |   | H(wealth | agegroup = 50s) = 0.957376  |
| 60s      | 2245 | 899 |   | H(wealth | agegroup = 60s) = 0.834049  |
| 70s      | 668 | 147 |   | H(wealth | agegroup = 70s) = 0.680862  |
| 80s      | 115 | 16 |   | H(wealth | agegroup = 80s) = 0.535474  |
| 90s      | 42 | 13 |   | H(wealth | agegroup = 90s) = 0.788941  |

$H(\text{wealth}) = 0.793844$  $H(\text{wealth} | \text{agegroup}) = 0.709463$

$IG(\text{wealth} | \text{agegroup}) = 0.0343813$
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>high</td>
<td>79 to 83</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75 to 78</td>
<td>asia</td>
<td>Europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75 to 78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75 to 78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>75 to 78</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75 to 78</td>
<td>america</td>
<td></td>
</tr>
</tbody>
</table>

From the UCI repository (thanks to Ross Quinlan)

Suppose we want to predict MPG.

Look at all the information gains...
A Decision Stump

Recursion Step

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

Build tree from These records..
Records in which cylinders = 4
Build tree from These records..
Records in which cylinders = 5
Build tree from These records..
Records in which cylinders = 6
Build tree from These records..
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)
The final tree

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
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

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($D_{S_i}, Output$)
    Where $D_{S_i}$ built consists of all those records in DataSet for which X = $i$th distinct value of X.
Outline

Information Gain
Learning an unpruned decision tree recursively

Training Set Error
Test Set Error
Overfitting
Information Gain of a real valued input

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 \textit{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.

Outline

- Information Gain
- Learning an unpruned decision tree recursively
- Training Set Error
- Test Set Error
- Overfitting
- Information Gain of a real valued input
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?

**Outline**

- Information Gain
- Learning an unpruned decision tree recursively
- Training Set Error
- Test Set Error
- Overfitting
- Information Gain of a real valued input
An artificial example

- We’ll create a training dataset

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

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.
Building a tree with the artificial training set

• Suppose we build a full tree (we always split until base case 2)

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

These bits are hidden

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

32 records

What decision tree would we learn now?

Without access to the irrelevant bits...

These nodes will be unexpandable
Without access to the irrelevant bits...

---

### Diagram

- **Root**
  - **e=0**
  - **e=1**

These nodes will be unexpandable

- 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

---

### Table

<table>
<thead>
<tr>
<th>Condition</th>
<th>Fine Corruptions</th>
<th>Corrupted Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/4 of the test set records are corrupted</td>
<td>n/a</td>
<td>1/4 of the test set will be wrongly predicted because the test record is corrupted</td>
</tr>
<tr>
<td>3/4 are fine</td>
<td>n/a</td>
<td>3/4 of the test predictions will be fine</td>
</tr>
</tbody>
</table>

---

In total, we expect to be wrong on only 1/4 of the test set predictions
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 a validation set to warn us that we might be overfitting.
Outline

Information Gain
Learning an unpruned decision tree recursively
Training Set Error
Test Set Error
Overfitting

Information Gain of a real valued input

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>2280</td>
<td>18.2</td>
<td>77</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>196</td>
<td>96</td>
<td>2648</td>
<td>10</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>121</td>
<td>110</td>
<td>2600</td>
<td>12.9</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>250</td>
<td>175</td>
<td>4100</td>
<td>13.9</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>196</td>
<td>95</td>
<td>3104</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>2271</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>2238</td>
<td>14</td>
<td>71</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>202</td>
<td>130</td>
<td>3570</td>
<td>12.8</td>
<td>78</td>
<td>america</td>
</tr>
<tr>
<td>bad</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>456</td>
<td>225</td>
<td>4435</td>
<td>10</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>107</td>
<td>85</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>

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 over fit

After pruning, it would likely to end up with a single root node.

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
Conclusions

- Decision trees have lots of advantages
  - 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 entropy and information gain, and how to interpret them
- 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
- How to exploit real-valued inputs
Discussion

• 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?

Questions?