A learning problem: predict fuel efficiency

- 40 Records
- Discrete data (for now)
- Predict MPG

From the UCI repository (thanks to Ross Quinlan)

Overview of Learning

<table>
<thead>
<tr>
<th>Type of Supervision</th>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Function</td>
<td>Classification</td>
<td></td>
<td>Clustering</td>
</tr>
<tr>
<td>Continuous Function</td>
<td>Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
<td></td>
</tr>
</tbody>
</table>

Hypotheses: decision trees \( f : X \rightarrow Y \)

- Each internal node tests an attribute \( x_i \).
- Each branch assigns an attribute value \( x_i = v \).
- Each leaf assigns a class \( y \).
- To classify input \( x \)?
  traverse the tree from root to leaf, output the labeled \( y \).

Hypothesis space

- How many possible hypotheses?
- What functions can be represented?
- How many will be consistent with a given dataset?
- How will we choose the best one?

Are all decision trees equal?

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

- Which tree do we prefer?
Learning decision trees is hard!!!

- Finding 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

Improving Our Tree

Recursive Step

Build tree from these records...
Build tree from these records...
Build tree from these records...
Build tree from these records...

A full tree

Two Questions

Greedy Algorithm:
- Start from empty decision tree
- Split on the best attribute (feature)
- Recurse

1. Which attribute gives the best split?
2. When to stop recursion?

Which attribute gives the best split?

A1: The one with the highest information gain
   Defined in terms of entropy

A2: Actually many alternatives, eg, accuracy
   Seeks to reduce the misclassification rate
Entropy

Entropy $H(Y)$ 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).

Entropy Example

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

$P(Y=t) = \frac{5}{6}$

$P(Y=f) = \frac{1}{6}$

$$H(Y) = - \frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6}$$

$= 0.65$

Conditional Entropy

Conditional Entropy $H(Y|X)$ of a random variable $Y$ conditioned on a random variable $X$:

$$H(Y|X) = - \sum_{j=1}^{l} P(X=x_j) \sum_{i=1}^{k} P(Y=y_i | X=x_j) \log_2 P(Y=y_i | X=x_j)$$

Example:

<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>F</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>

$P(X_1=t) = \frac{4}{6}$

$P(X_1=f) = \frac{2}{6}$

$H(Y|X_1) = - \frac{4}{6} (1 \log_2 1 + 0 \log_2 0)$

$= \frac{2}{6}$

$= 0.33$

Information Gain

**Advantage of attribute** — decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y | X)$$

In our running example:

$$IG(X_1) = H(Y) - H(Y|X_1)$$

$= 0.65 - 0.33$

$= 0.32$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

Alternate Splitting Criteria

- **Misclassification Impurity**
  Minimum probability that a training pattern will be misclassified
  
  $$M(Y) = 1 - \max_i P(Y=y_i)$$

- **Misclassification Gain**
  
  $$IG_m(X) = \left[1 - \max_i P(Y=y_i)\right] - \left[\max_j \left(\max_i P(Y=y_i | x=x_j)\right)\right]$$

Learning Decision Trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use information gain (or...?) to select attribute:
    
    $$\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y | X_i)$$

- Recurse
Suppose we want to predict MPG

Now, Look at all the information gains...

When to Terminate?

Base Case Two: No attributes can distinguish

Base Case Two

Don’t split a node if none of the attributes can create multiple (non-empty) children

Tree After One Iteration

Base Case One

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

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

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

The resulting decision tree:

<table>
<thead>
<tr>
<th>y values</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Predict 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The information gains:

Building Decision Trees

BuildTree(DataSet, Output)

If all output values are the same in dataSet,

Then return a leaf node that says “predict this unique output”

If all input values are the same,

Then 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 \( i^\text{th} \) child is built by calling BuildTree(DS\(_i\), Output)

Where DS\(_i\) consists of all those records in DataSet for which \( X = i^\text{th} \) distinct value of \( X \).

General View of a Classifier

Hypothesis: Decision Boundary for labeling function

<table>
<thead>
<tr>
<th>Label: +</th>
<th>Label: -</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y )</td>
<td></td>
</tr>
<tr>
<td>( 0.0 )</td>
<td></td>
</tr>
<tr>
<td>( 1.0 )</td>
<td></td>
</tr>
<tr>
<td>( 2.0 )</td>
<td></td>
</tr>
<tr>
<td>( 3.0 )</td>
<td></td>
</tr>
</tbody>
</table>

Decision Tree Decision Boundaries

Decision trees divide the feature space into non-parallel rectangles, and label each rectangle with one of the \( K \) classes.
Ok, so how does it perform?

The test set error is much worse than the training set error… why?

Decision trees will overfit

• Our decision trees have no learning bias
  – Training set error is always zero!
    • (if there is no label noise)
  – Lots of variance
  – Will definitely overfit!!!
    – Must introduce some bias towards simpler trees

• Why might one pick simpler trees?

Occam’s Razor

• Why Favor Short Hypotheses?
  • Arguments for:
    – Fewer short hypotheses than long ones
      → A short hyp. less likely to fit data by coincidence
    – Longer hyp. that fit data may might be coincidence
  • Arguments against:
    – Argument above on really uses the fact that hypothesis space is small!!
    – What is so special about small sets based on the complexity of each hypothesis?

How to Build Small Trees

Several reasonable approaches:

• Stop growing tree before overfit
  – Bound depth or \# leaves
  – Base Case 3
    – Doesn’t work well in practice

• Grow full tree; then prune
  – Optimize on a held-out (development set)
    • If growing the tree hurts performance, then cut back
  – Con: Requires a larger amount of data...
  – Use statistical significance testing
    • Test if the improvement for any split is likely due to noise
    • If so, then prune the split!
  – Convert to logical rules
    • Then simplify rules
Reduced Error Pruning

Split data into *training & validation* sets (10-33%)

Train on training set (overfitting)

Do until further pruning is harmful:

1) Evaluate effect on validation set of pruning *each* possible node (and tree below it)
2) Greedily remove the node that *most improves* accuracy of validation set

Effect of Reduced-Error Pruning

Alternatively

- Chi-squared pruning
  - Grow tree fully
  - Consider leaves in turn
    - Is parent split worth it?
- Compared to Base-Case 3?

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 13.5%

Such hypothesis tests are relatively easy to compute, but involved

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.
Regularization

- Note for Future: MaxPChance is a regularization parameter that helps us bias towards simpler models.

![Graph showing the relationship between MaxPChance and tree size]

We’ll learn to choose the value of magic parameters like this one later!

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)
- Improved by
  - Carlos Guestrin &
  - Luke Zettlemoyer