**Decision Trees**

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

**Another Example Problem**

**A Decision Tree**

<table>
<thead>
<tr>
<th>Type</th>
<th>Doors</th>
<th>Tires</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Minivan</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUV</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

**Decision Trees**

- Decision tree representation
  - Each internal node tests an attribute
  - Each branch corresponds to an attribute value
  - Each leaf node assigns a classification

**When to Consider Decision Trees**

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

**Examples**

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

**Top-Down Induction of Decision Trees**

**Main loop:**
1. A = the “best” decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A, create descendant of node
4. Divide training examples among child nodes
5. If training examples perfectly classified, STOP
   - Else iterate over new leaf nodes

**Which attribute is best?**

- $A_1$ [29+,35-]
- $A_2$ [20+,30-]
Entropy

$S$ = sample of training examples
$p_+$ = proportion of positive examples in $S$
$p_-$ = proportion of negative examples in $S$
Entropy measures the impurity of $S$

$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$

Information Gain

$Gain(S, A) = \text{expected reduction in entropy due to sorting on } A$

$Gain(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{\text{Entropy}(S_v)}{|S_v|}$

Car Examples

<table>
<thead>
<tr>
<th>Color</th>
<th>Type</th>
<th>Doors</th>
<th>Tires</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>SUV</td>
<td>2</td>
<td>Whitewall</td>
<td>+</td>
</tr>
<tr>
<td>Blue</td>
<td>Minivan</td>
<td>4</td>
<td>Whitewall</td>
<td>-</td>
</tr>
<tr>
<td>Green</td>
<td>Car</td>
<td>4</td>
<td>Whitewall</td>
<td>-</td>
</tr>
<tr>
<td>Red</td>
<td>Minivan</td>
<td>4</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Green</td>
<td>Car</td>
<td>2</td>
<td>Blackwall</td>
<td>+</td>
</tr>
<tr>
<td>Green</td>
<td>SUV</td>
<td>4</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Blue</td>
<td>SUV</td>
<td>2</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Blue</td>
<td>Car</td>
<td>2</td>
<td>Whitewall</td>
<td>-</td>
</tr>
<tr>
<td>Red</td>
<td>SUV</td>
<td>2</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Green</td>
<td>Minivan</td>
<td>4</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Green</td>
<td>Minivan</td>
<td>2</td>
<td>Whitewall</td>
<td>-</td>
</tr>
</tbody>
</table>

Selecting Root Attribute

$Gain(S, \text{Color}) = 0.029$

Best attribute: \text{Type} (Gain = 0.200)
Selecting Next Attribute

\[ \text{Gain}(\text{Type}, \text{Color}) = 0.971 - \left( \frac{1}{5} \right) 0.0 - \left( \frac{2}{5} \right) 1.0 = 0.171 \]
\[ \text{Gain}(\text{Type}, \text{Doors}) = 0.971 - \left( \frac{3}{5} \right) 0.0 - \left( \frac{2}{5} \right) 0.0 = 0.971 \]
\[ \text{Gain}(\text{Type}, \text{Tires}) = 0.971 - \left( \frac{2}{5} \right) 1.0 - \left( \frac{3}{5} \right) 0.918 = 0.020 \]

Resulting Tree

\[ \text{Gain}(\text{SSUV}, \text{Color}) = 0.918 - \left( \frac{2}{6} \right) 1.0 - \left( \frac{1}{6} \right) 0.0 - \left( \frac{3}{6} \right) 0.918 = 0.126 \]
\[ \text{Gain}(\text{SSUV}, \text{Doors}) = 0.918 - \left( \frac{4}{6} \right) 1.011 - \left( \frac{2}{6} \right) 1.0 = 0.044 \]
\[ \text{Gain}(\text{SSUV}, \text{Tires}) = 0.918 - \left( \frac{2}{6} \right) 0.0 - \left( \frac{4}{6} \right) 0.0 = 0.918 \]

Hypothesis Space Search by ID3

- Hypothesis space is complete!
- Target function is in there (but will we find it?)
- Outputs a single hypothesis (which one?)
- Cannot play 20 questions
- No back tracing
- Local minima possible
- Statistically-based search choices
- Robust to noisy data
- Inductive bias: approximately “prefer shortest tree”

Inductive Bias in ID3

Note \( H \) is the power set of instances \( X \)

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a preference for some hypotheses, rather than a restriction of hypothesis space \( H \)
- Occam’s razor: prefer the shortest hypothesis that fits the data

Occam’s Razor

Why prefer short hypotheses?

Argument in favor:
- Fewer short hypotheses than long hypotheses
- Short hyp. that fits data unlikely to be coincidence
- Long hyp. that fits data more likely to be coincidence

Argument opposed:
- There are many ways to define small sets of hypotheses
- e.g., all trees with a prime number of nodes that use attributes beginning with “Z”
- What is so special about small sets based on size of hypothesis?
Overfitting in Decision Trees
Consider adding a noisy training example: 
<Green,SUV,2,Blackwall> +
What happens to decision tree below?

Overfitting
Consider error of hypothesis $h$ over
- training data: $error_{train}(h)$
- entire distribution $D$ of data: $error_D(h)$
Hypothesis $h \in H$ overfits the training data if there is an alternative hypothesis $h' \in H$ such that $error_{train}(h) < error_{train}(h')$ and $error_D(h) > error_D(h')$

Avoiding Overfitting
How can we avoid overfitting?
- stop growing when data split not statistically significant
- grow full tree, then post-prune
How to select “best” tree:
- Measure performance over training data
- Measure performance over separate validation set (examples from the training set that are put aside)
- MDL: minimize $\text{size(tree)} + \text{size(misclassifications(tree)}$

Reduced-Error Pruning
Split data into training and validation set
Do until further pruning is harmful:
1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy
- Produces smallest version of most accurate subtree
- What if data is limited?
Decision Tree Post-Pruning

- A standard method in C4.5, C5.0
- Construct a complete tree
  - For each node estimate what the error might be with and without the node (needs a conservative estimate of error since based on training data)
  - Prune any node where the expected error stays the same or drops
  - Greatly influenced by method for estimating likely errors

Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Converting a Tree to Rules

IF (Type=Car) AND (Doors=2) THEN +
IF (Type=SUV) AND (Tires=Whitewall) THEN +
IF (Type=Minivan) THEN -
… (what else?)

Continuous Valued Attributes

Create one (or more) corresponding discrete attributes based on continuous
- (EngineSize = 325) = true or false
- (EngineSize <= 330) = t or f (330 is “split” point)

Attributes with Many Values

Problem:
- If attribute has many values, Gain will select it
- Imagine if cars had PurchaseDate feature - likely all would be different

One approach: use GainRatio instead

\[
GainRatio(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}
\]

\[
SplitInformation(S, A) = -\sum_{i} \left| S_i \right| \log_{2} \left( \frac{\left| S_i \right|}{|S|} \right)
\]

where \( S_i \) is subset of \( S \) for which \( A \) has value \( v_i \)

Attributes with Costs

Consider
- medical diagnosis, BloodTest has cost $150
- robotics, Width_from_1ft has cost 23 second

How to learn consistent tree with low expected cost?

Approaches: replace gain by

Tan and Schlimmer (1990)
Nunez (1988)

\[
Gain'(S, A) = \frac{Gain(S, A)}{Cost(A)}
\]

\[
\text{Nunez (1988)}\quad 2^{\text{cost}(A) w} - 1
\]

where \( w \in [0,1] \) and determines importance of cost
Unknown Attribute Values

What if some examples missing values of \( A \)?

“?” in C4.5 data sets

Use training example anyway, sort through tree

- If node \( n \) tests \( A \), assign most common value of \( A \) among other examples sorted to node \( n \)
- Assign most common value of \( A \) among other examples with same target value
- Assign probability \( p_i \) to each possible value \( v_i \) of \( A \)
  - Assign fraction \( p_i \) of example to each descendant in tree

Classify new examples in same fashion