**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>2</td>
<td>+</td>
</tr>
<tr>
<td>Minivan</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>SUV</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

**Decision Trees**

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

How would you represent:

- \(\land, \lor, \text{XOR}\)
- \((A \land B) \lor (C \land \neg D \land E)\)
- M of N

**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

**Top-Down Induction of Decision Trees**

Main loop:
1. \(A = \text{the “best” decision attribute for next node}\)
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?

- \([29^+, 35^-]\)
- \([21^+, 5^-]\)
- \([8^+, 30^-]\)
- \([18^+, 33^-]\)
- \([11^+, 2^-]\)
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 $S$

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

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

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{r \in \text{values}(A)} \left( \frac{|S_r|}{|S|} \text{Entropy}(S_r) \right)$$

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>Blue</td>
<td>SUV</td>
<td>4</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Blue</td>
<td>SUV</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>Blue</td>
<td>Car</td>
<td>4</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Blue</td>
<td>Car</td>
<td>2</td>
<td>Blackwall</td>
<td>+</td>
</tr>
<tr>
<td>Green</td>
<td>SUV</td>
<td>4</td>
<td>Whitewall</td>
<td>+</td>
</tr>
<tr>
<td>Green</td>
<td>Minivan</td>
<td>4</td>
<td>Whitewall</td>
<td>+</td>
</tr>
<tr>
<td>Green</td>
<td>Minivan</td>
<td>2</td>
<td>Blackwall</td>
<td>-</td>
</tr>
<tr>
<td>Green</td>
<td>Minivan</td>
<td>4</td>
<td>Whitewall</td>
<td>+</td>
</tr>
</tbody>
</table>

Selecting Root Attribute

- $S$: [5+, 9-] $E = 0.940$
- $S$: [5+, 9-] $E = 0.940$

Gain($S$, Color) = 0.029

Gain($S$, Type) = 0.200

Selecting Root Attribute (cont)

Best attribute: Type (Gain = 0.200)
Selecting Next Attribute

Gain(S_{Car, Color}) = \frac{971}{5} = 194.2
Gain(S_{Car, Doors}) = \frac{971}{5} = 194.2
Gain(S_{Car, Tires}) = \frac{971}{5} = 194.2

Resulting Tree

Hypothesis Space Search by ID3

Hypothesis Space Search by ID3

Inductive Bias in ID3

Occam’s Razor

Why prefer short hypotheses?

Argument in favor:
- Fewer short hypotheses than long hypotheses
- Short hypotheses fit data unlikely to be coincidence
- Long hypotheses fit 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:
\[ \text{<Green,SUV,2,Blackwall>} + \]

What happens to decision tree below?

![Decision Tree Diagram](image)

Overfitting

Consider error of hypothesis \( h \) over
- training data: \( \text{error}_{\text{train}}(h) \)
- entire distribution \( D \) of data: \( \text{error}_{D}(h) \)

Hypothesis \( h \in H \) overfits the training data if there is an alternative hypothesis \( h' \in H \) such that \( \text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h') \) and \( \text{error}_{D}(h) > \text{error}_{D}(h') \)

Overfitting in Decision Tree Learning

![Accuracy vs. Size of Tree Graph](image)

Avoiding Overfitting

How can we avoid overfitting?
- stop growing when data split not statistically significant
- grow full tree, the 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
\[
\text{size}(\text{tree}) + \text{size}(\text{misclassifications}(\text{tree}))
\]

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?
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

Perhaps most frequently used method (e.g., C4.5)

Converting a Tree to Rules
IF (Type=Car) AND (Doors=2) THEN +
IF (Type=SUV) AND (Tires=Whitewall) THEN +
IF (Type=Minivan) THEN -

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)

How to pick best “split” point?
1. Sort continuous data
2. Look at points where class differs between two values
3. Pick the split point with the best gain

<table>
<thead>
<tr>
<th>EngineSize</th>
<th>285</th>
<th>290</th>
<th>295</th>
<th>310</th>
<th>330</th>
<th>330</th>
<th>345</th>
<th>360</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

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

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

\[
\text{SplitInformation}(S, A) = -\sum_{i=1}^{l} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
\]

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)

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

Nunez (1988)

\[
\frac{2^{\text{Gain}(S, A)}}{(\text{Cost}(A)+1)^w} - 1
\]

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 \( A \)

Classify new examples in same fashion
Decision Tree Summary

- simple (easily understood), powerful (accurate)
- highly expressive (complete hypothesis space)
- bias: preferential
  - search based on information gain (defined using entropy)
  - favors short hypotheses, high gain attributes near root
- issues:
  - overfitting
    - avoiding: stopping early, pruning
    - pruning: how to judge, what to prune (tree, rules, etc.)

Decision Tree Summary (cont)

- issues (cont):
  - attribute issues
    - continuous valued attributes
    - attributes with lots of values
    - attributes with costs
    - unknown values
- effective for discrete valued target functions
- handles noise