Decision Tree for PlayTennis
Decision Trees

Decision tree representation:

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

How would we 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

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences
Top-Down Induction of Decision Trees

Main loop:
1. $A \leftarrow$ the “best” decision attribute for next node
2. Assign $A$ as decision attribute for node
3. For each value of $A$, create new descendant of node
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?
Entropy

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$
- $p_\ominus$ is the proportion of negative examples in $S$
- Entropy measures the impurity of $S$

$$Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$
Entropy

$Entropy(S) = \text{expected number of bits needed to encode class (} \oplus \text{ or } \ominus \text{) of randomly drawn member of } S \text{ (under the optimal, shortest-length code)}$

Why?

Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability $p$.

So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of $S$:

\[ p_\oplus(-\log_2 p_\oplus) + p_\ominus(-\log_2 p_\ominus) \]

\[ Entropy(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus \]
Information Gain

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

\[ \text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \]

[Diagram of decision trees for A1 and A2 with labels and values]
## Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Selecting the Next Attribute

Which attribute is the best classifier?

Gain (S, Humidity )
\[
= 0.940 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 \\
= 0.151
\]

Gain (S, Wind)
\[
= 0.940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.0 \\
= 0.048
\]
Which attribute should be tested here?

\[S_{sunny} = \{D1,D2,D8,D9,D11\}\]

\[
\begin{align*}
\text{Gain (} S_{sunny}, \text{Humidity}) &= .970 - (3/5) 0.0 - (2/5) 0.0 = .970 \\
\text{Gain (} S_{sunny}, \text{Temperature}) &= .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570 \\
\text{Gain (} S_{sunny}, \text{Wind}) &= .970 - (2/5) 1.0 - (3/5) .918 = .019
\end{align*}
\]
Hypothesis Space Search by ID3
Hypothesis Space Search by ID3

• Hypothesis space is complete!
  – Target function surely in there...
• Outputs a single hypothesis (which one?)
  – Can’t play 20 questions...
• No back tracking
  – Local minima...
• Statistically-based search choices
  – Robust to noisy data...
• Inductive bias: approx “prefer shortest tree”
Inductive Bias in ID3

Note $H$ is the power set of instances $X$

→ Unbiased?

Not really...

- 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 hyps. than long hyps.
→ a short hyp that fits data unlikely to be coincidence
→ a long hyp that fits data might be coincidence

Argument opposed:

• There are many ways to define small sets of hyps
• e.g., all trees with a prime number of nodes that use attributes beginning with “Z”
• What’s so special about small sets based on size of hypothesis??
Overfitting in Decision Trees

Consider adding noisy training example #15:

*Sunny, Hot, Normal, Strong, PlayTennis = No*

What effect on earlier tree?
Overfitting

Consider error of hypothesis $h$ over

- training data: $error_{\text{train}}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{\text{train}}(h) < error_{\text{train}}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$
Overfitting in Decision Tree Learning

![Graph showing accuracy vs. size of tree (number of nodes)](image)

- **On training data**
- **On test data**

Accuracy on training data increases with the size of the tree, while accuracy on test data peaks at a smaller size and then decreases.
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 data set
- 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?
Effect of Reduced-Error Pruning
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

```
Outlook
  Sunny
  Overcast
    Yes
      Humidity
        High
          No
          Yes
        Normal
          No
          Yes

Rain
  Yes
      Wind
        Strong
          No
          Yes
        Weak
          No
          Yes
```
IF \((Outlook = Sunny) \land (Humidity = High)\) 
THEN \(PlayTennis = No\)

IF \((Outlook = Sunny) \land (Humidity = Normal)\) 
THEN \(PlayTennis = Yes\)

\[\ldots\]
Continuous Valued Attributes

Create a discrete attribute to test continuous

• \( Temperature = 82.5 \)
• \( (Temperature > 72.3) = t, f \)

<table>
<thead>
<tr>
<th>Temperature:</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis:</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Attributes with Many Values

Problem:

• If attribute has many values, Gain will select it
• Imagine using Date = Jun_3_1996 as attribute

One approach: use GainRatio instead

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

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

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

Consider

- medical diagnosis, \textit{BloodTest} has cost $150
- robotics, \textit{Width\_from\_1\_ft} has cost 23 sec.

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)

\[
\frac{Gain^2(S, A)}{Cost(A)}.
\]

- Nunez (1988)

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

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

What if some examples missing values of $A$?
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