Outline

- Decision tree representation
- ID3 learning algorithm
- Entropy and information gain
- Overfitting
- Enhancements
Decision Tree for \textit{PlayTennis}

- **Outlook**
  - Sunny
  - Overcast
  - Rain
    - **Humidity**
      - High
      - Normal
        - No
        - Yes
    - **Wind**
      - Strong
      - Weak
        - No
        - Yes
Decision Trees

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

- How would we represent:
  - $\land$, $\lor$, 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 (ID3, Table 3.1):
  - $A \leftarrow$ the “best” decision attribute for next node
  - Assign $A$ as decision attribute for node
  - For each value of $A$, create new descendant of node
  - Sort training examples to leaf nodes
  - If training examples perfectly classified
    - Then STOP
    - Else iterate over new leaf nodes
Which Attribute is Best?

\[29^+, 35^-\]

A1 = ?

\[21^+, 5^-\] \quad \text{t} \quad \text{f} \quad [8^+, 30^-]

\[29^+, 35^-\]

A2 = ?

\[18^+, 33^-\] \quad \text{t} \quad \text{f} \quad [11^+, 2^-]
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 $S$
  - $\text{Entropy}(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$
Entropy

- \( \text{Entropy}(S) = \) expected number of bits needed to encode class (\( \oplus \) or \( \ominus \)) of randomly drawn member of \( S \) (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 \( S \):
- \( p_\oplus (- \log_2 p_\oplus) + p_\ominus (- \log_2 p_\ominus) \)
- \( \text{Entropy}(S) \equiv - p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus \)
Information Gain

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

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

```
[29+, 35-]  A1=?
  | t  | f |
  | [21+, 5-]   [8+, 30-] |

[29+, 35-]  A2=?
  | t  | f |
  | [18+, 33-]    [11+, 2-] |
```
Training Examples: *PlayTennis*

<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?

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

- **Humidity**
  - **High**
    - \([3+,4-]\)
      - \(E = 0.985\)
  - **Normal**
    - \([6+,1-]\)
      - \(E = 0.592\)

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

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

- **Wind**
  - **Weak**
    - \([6+,2-]\)
      - \(E = 0.811\)
  - **Strong**
    - \([3+,3-]\)
      - \(E = 1.00\)

\[ \text{Gain (S, Wind)} = 0.940 - \left(\frac{8}{14}\right) \times 0.811 - \left(\frac{6}{14}\right) \times 1.0 = 0.048 \]
Selecting the Next Attribute

Which attribute should be tested here?

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

\[
\text{Gain} (S_{\text{Sunny}}, \text{Humidity}) = 0.970 - (3/5) 0.0 - (2/5) 0.0 = 0.970
\]

\[
\text{Gain} (S_{\text{Sunny}}, \text{Temperature}) = 0.970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = 0.570
\]

\[
\text{Gain} (S_{\text{Sunny}}, \text{Wind}) = 0.970 - (2/5) 1.0 - (3/5) 0.918 = 0.019
\]
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 backtracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: “prefer shortest tree”
Inductive Bias in ID3

- Note $H$ is the power set of instances $X$
  - Unbiased?
- Not really...
  - Preference for short trees with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* on the 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 hypothesis that fits data unlikely to be coincidence
    - Long hypothesis that fits data might 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's so special about small sets based on the size of the 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 $D$ of data: $error_{D}(h)$
- Hypothesis $h \in H$ **overfits** the training data if there is an alternative hypothesis $h' \in H$ such that
  - $error_{\text{train}}(h) < error_{\text{train}}(h')$
  - $error_{D}(h) > error_{D}(h')$
Overfitting in Decision Tree Learning

![Graph showing overfitting in decision tree learning](image)

- Accuracy on training data
- Accuracy on test data
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:
  - Evaluate impact on *validation* set of pruning each possible node (plus those below it)
  - 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

![Graph showing the effect of reduced-error pruning on accuracy vs. size of tree (number of nodes). The graph includes lines for training data and test data, with a specific note for test data during pruning.]
Rule Post-Pruning

- Generate decision tree, then
  - Convert tree to equivalent set of rules
  - Prune each rule independently of others
  - Sort final rules into desired sequence for use
- Perhaps most frequently used method (e.g., C4.5)
Converting a Tree to Rules

If (Outlook=Sunny) \land (Humidity=High) 
Then PlayTennis=No

If (Outlook=Sunny) \land (Humidity=Normal) 
Then PlayTennis=Yes

...
Continuous Valued Attributes

- Create a discrete attribute to test continuous values
  - Temperature = 82.5
  - \((\text{Temperature} > 72.3) = \text{true}, \text{false}\)

<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.

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

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

- where \( S_i \) is the 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 sec.
- How to learn a consistent tree with low expected cost?
- One approach: replace gain by
  - Tan and Schlimmer (1990)
    \[
    \frac{\text{Gain}^2(S,A)}{\text{Cost}(A)}
    \]
  - Nunez (1988)
    \[
    \frac{2^{\text{Gain}(S,A)} - 1}{(\text{Cost}(A) + 1)^w}
    \]
  - 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
Summary: Decision-Tree Learning

- Most popular symbolic learning method
- Learning discrete-valued functions
- Information-theoretic heuristic
- Handles noisy data
- Decision trees completely expressive
- Biased towards simpler trees
- ID3 $\rightarrow$ C4.5 $\rightarrow$ C5.0 (www.rulequest.com)
- J48 (WEKA) $\approx$ C4.5