Decision Trees

CptS 570 – Machine Learning
School of EECS
Washington State University
Decision Trees

- Nonparametric method
- Representation
- Learning algorithm
- Entropy and information gain
- Overfitting
- Enhancements
Decision Tree Example
Decision Tree for **weather.nominal**

- **Outlook**
  - Sunny
  - Humidity
    - High
      - No
    - Normal
      - Yes
  - Overcast
    - Yes
  - Rain
    - Wind
      - Strong
        - No
      - Weak
        - Yes
Each internal node tests one (univariate) or more (multivariate) features

Each branch corresponds to a test result
  ◦ Yes/no for numeric features or multivariate
  ◦ One for each value of discrete feature

Leaf node
  ◦ Classification: Class value
  ◦ Regression: Real value
Regression Tree
Multivariate Tree

\[ w_{11}x_1 + w_{12}x_2 + w_{10} > 0 \]
Learning Algorithm

- Create root node with all examples X
- Call GenerateTree (root, X)

```plaintext
GenerateTree (node, X)
    If node is "pure" enough
    Then assign class to node and return
    Else Choose "best" split feature F
        Foreach value v of F
            X' = examples in X where F = v
            Create childNode with examples X'
            GenerateTree (childNode, X')
```
Entropy (2-class)

- $S = \text{set of training examples}$
- $p_\oplus = \text{proportion of positive examples in } S$
- $p_\ominus = \text{proportion of negative examples in } S$
- Entropy measures the impurity of $S$
- $\text{Entropy}(S) \equiv - p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$
Entropy (2-class)

- Entropy(S) = expected number of bits needed to encode class (⊕ or ⊥) 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 ⊕ or ⊥ of random member of S:
  ◦ \( p_⊕ (− \log_2 p_⊕ ) + p_⊥ (− \log_2 p_⊥ ) \)
  ◦ \( \text{Entropy}(S) \equiv − p_⊕ \log_2 p_⊕ − p_⊥ \log_2 p_⊥ \)
Learning Algorithm

- When is a node “pure”?  
  - Entropy small enough
- Which feature is “best”?  
  - One that leads to most reduction in entropy  
  - I.e., most information gain
Information Gain

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

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

[Diagram of decision trees for \( A_1 \) and \( A_2 \) with values and labels]
## Training Examples: weather.nominal

<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 Best Feature

Gain (S, Humidity)
\[
= 0.940 - (7/14) \times 0.985 - (7/14) \times 0.592 \\
= 0.151
\]

Gain (S, Wind)
\[
= 0.940 - (8/14) \times 0.811 - (6/14) \times 1.0 \\
= 0.048
\]
Selecting the Next Feature

Which attribute should be tested here?

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

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

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

\[
\text{Gain} (S_{\text{Sunny}}, \text{Wind}) = 0.970 - \frac{2}{5} \times 1.0 - \frac{3}{5} \times 0.918 = 0.019
\]
Hypothesis Space Search
Hypothesis Space Search

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Does not keep all consistent hypotheses
- No backtracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: “prefer shortest tree”
Consider adding noisy training example #15:
- \( \langle \text{Sunny, Hot, Normal, Strong} \rangle, \text{PlayTennis} = \text{No} \)
- What effect on earlier tree?
Consider error of hypothesis $h$ over

- Training data: $error_X(h)$
- 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

- $error_X(h) < error_X(h')$
- $error_D(h) > error_D(h')$
Overfitting in Decision Tree Learning

![Graph showing accuracy over size of tree (number of nodes) for training and 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 \( \text{size(tree)} + \text{size(misclassifications(tree))} \)
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
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

- Allows pruning individual constraints rather than just entire subtrees

- Perhaps most frequently used method
If (Outlook=Sunny) \land (Humidity=High)
Then PlayTennis=No

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

...
Real-Valued Features

- Create a discrete feature to test real values
  - Temperature = 82.5
  - (Temperature > 72.3) = true, false
- Sort values and try splits on midpoints

<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>

(T < 40) (T < 44) (T < 54) (T < 66) (T < 76) (T < 85)
Features with Many Values

**Problem:**

- If feature has many values, Gain will select it
- Imagine using Day as feature for weather dataset

**One approach:** Use GainRatio instead

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

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

- where \( S_i \) is the subset of \( S \) for which \( F \) has value \( v_i \)
Features 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?
- Approach: Modify Gain to include cost

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

where \( w \in [0,1] \) determines importance of cost
What if some examples missing values of \( F \)?

Use training example anyway, sort through tree

◦ If node \( n \) tests \( F \), assign most common value of \( F \) among other examples sorted to node \( n \)

◦ Assign most common value of \( F \) among other examples with same target value

◦ Assign probability \( p_i \) to each possible value \( v_i \) of \( F \)
  • Assign fraction \( p_i \) of example to each descendant in tree

Classify new examples in same fashion
Summary: Decision Trees

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