Chapter 3: 
Decision Tree Learning

CS 536: Machine Learning
Littman (Wu, TA)

Classification Learning
Instances are vectors of attribute values.
A concept is a function that maps instances to categories (T, F, say).
A target concept is the concept we want to learn.
A hypothesis class is the set of concepts we consider.
A sample of instances (or training set) is our source of information about the target concept.
Our candidate concept is usually evaluated by how well it classifies a separate sample of instances (the testing set).

Decision–Tree Learning
[read Chapter 3]
[some of Chapter 2 might help...]
[recommended exercises 3.1, 3.2]
• Decision tree representation
• ID3 learning algorithm
• Entropy, Information gain
• Overfitting

Decision Tree for PlayTennis

\[
\text{Outlook} \\
| Sunny \quad Overcast \quad Rain \\
| Yes \quad Yes \\
\text{Humidity} \\
| High \quad Normal \\
| No \quad Yes \\
\text{Wind} \\
| Strong \quad Weak \\
| No \quad Yes
\]
Predicting C-Section Risk

Learned from medical records of 1000 women
Negative examples are C-sections

\[ [833+,167-] \cdot .83+ .17- \]

Fetal_Presentation = 1: \[ [822+,116-] \cdot .88+ .12- \]
| Previous_Csection = 0: \[ [767+,81-] \cdot .90+ .10- \]
| | Primiparous = 0: \[ [399+,13-] \cdot .97+ .03- \]
| | Primiparous = 1: \[ [368+,68-] \cdot .84+ .16- \]
| | | Fetal_Distress = 0: \[ [334+,47-] \cdot .88+ .12- \]
| | | | Birth_Weight < 3349: \[ [201+,10.6-] \cdot .95+ .05 \]
| | | | Birth_Weight >= 3349: \[ [133+,3.64-] \cdot .78+ .2 \]
| | | Fetal_Distress = 1: \[ [34+,21-] \cdot .62+ .38- \]
| | Previous_Csection = 1: \[ [55+,35-] \cdot .61+ .39- \]
Fetal_Presentation = 2: \[ [3+,29-] \cdot .11+ .89- \]
Fetal_Presentation = 3: \[ [8+,22-] \cdot .27+ .73- \]

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

Whence 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

Main loop:
1. \( A \leftarrow \text{the “best” decision attribute for next node} \)
2. Assign \( A \) as decision attribute for \( \text{node} \)
3. For each value of \( A \), create new descendant of \( \text{node} \)
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?

Which Attribute is Best?

Measuring Entropy

- $S$ is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in $S$
- $p_\otimes$ is the proportion of negative examples in $S$

Entropy measures the impurity of $S$

$Entropy(S) = -p_\oplus \log p_\oplus - p_\otimes \log p_\otimes$

Entropy Function

Entropy Function

Entropy

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

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 $\otimes$ of a random member of $S$:

$p_\oplus (- \log p_\oplus) + p_\otimes (- \log p_\otimes)$
Information Gain

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

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

Here, \( S_v \) is the set of training instances remaining from \( S \) after restricting to those for which attribute \( A \) has value \( v \).

Which Attribute is Best?

Selecting the Next Attribute

Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Hum.</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>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Nml</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Nml</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>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Nml</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Nml</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>Nml</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>
Comparing Attributes

\( S_{\text{sunny}} = \{D1,D2,D8,D9,D11\} \)

- \( \text{Gain} (S_{\text{sunny}}, \text{Humidity}) = 0.970 - (3/5) 0.0 - (2/5) 0.0 = 0.970 \)
- \( \text{Gain} (S_{\text{sunny}}, \text{Temp}) = 0.970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = 0.570 \)
- \( \text{Gain} (S_{\text{sunny}}, \text{Wind}) = 0.970 - (2/5) 1.0 - (3/5) 0.918 = 0.019 \)

Hypothesis Space Search by ID3

ID3:
- representation : trees
- scoring : entropy
- search : greedy

What is ID3 Optimizing?

How would you find a tree that minimizes:
- misclassified examples?
- expected entropy?
- expected number of tests?
- depth of tree given a fixed accuracy?
- etc.?

How decide if one tree beats another?
**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**

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: $error_{train}(h)$
- entire distribution $D$ of data: $error_{D}(h)$

Hypothesis $h$ in $H$ **overfits** training data if there is an alternative hypothesis $h'$ in $H$ such that
- $error_{train}(h) < error_{train}(h')$, and
- $error_{D}(h) > error_{D}(h')$

Overfitting in Learning

How can we avoid overfitting?
- stop growing when data split not statistically significant
- grow full tree, then post-prune (DP alg!)

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:
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)

Effect of Pruning

Converting Tree to Rules
The Rules

IF (Outlook = Sunny) \land (Humidity = High)
THEN PlayTennis = No
IF (Outlook = Sunny) \land (Humidity = Normal)
THEN PlayTennis = Yes

Attributes with Many Values

Problem:
• If one attribute has many values compared to the others, Gain will select it
• Imagine using Date = Jun_3_1996 as attribute
One approach: use GainRatio instead
\[ GainRatio(S,A) = Gain(S,A) / SplitInfo(S,A) \]
\[ SplitInfo(S,A) = -\sum_{i=1}^{c} |S_i|/|S| \log_2 |S_i|/|S| \]
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 sec.
How to learn a consistent tree with low expected cost? Find min cost tree.
Another approach: replace gain by
• Tan and Schlimmer (1990)
\[ Gain^2(S,A)/Cost(A) \]
• Nunez (1988) \([w \in [0,1]: \text{importance}]\)
\[ (2^{Gain(S,A)}-1)/(Cost(A)+1)^w \]

Continuous Valued Attributes

Create a discrete attribute to test continuous
• \( Temp = 82.5 \)
• \((Temp > 72.3) = t, f\)

<table>
<thead>
<tr>
<th>Temp: 40 48 60 72 80 90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis: No No Yes Yes Yes No</td>
</tr>
</tbody>
</table>
**Unknown Attribute Values**

Some examples missing values of $A$?
Use training example anyway, sort it
- 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 $A$ (perhaps as above)
  - assign fraction $p_i$ of example to each descendant in tree
- Classify new examples in same fashion