Chapter 3: Decision Tree Learning

CS 536: Machine Learning
Littman (Wu, TA)

Administration

Books?
New web page: http://www.cs.rutgers.edu/~mlittman/courses/ml03/
• schedule
• lecture notes
• assignment info. (when available)
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

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).
Predicting C–Section Risk

Learned from medical records of 1000 women

Negative examples are C–sections

<table>
<thead>
<tr>
<th>Fetal_Presentation = 1:</th>
<th>822+,116−</th>
<th>.88+ .12−</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous_Csection = 0:</td>
<td>767+,81−</td>
<td>.90+ .10−</td>
</tr>
<tr>
<td></td>
<td>Primiparous = 0:</td>
<td>399+,13−</td>
</tr>
<tr>
<td></td>
<td>Primiparous = 1:</td>
<td>368+,68−</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fetal_Distress = 0:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Previous_Csection = 1:</td>
<td>55+,35−</td>
</tr>
<tr>
<td>Fetal_Presentation = 2:</td>
<td>3+,29−</td>
<td>.11+ .89−</td>
</tr>
<tr>
<td>Fetal_Presentation = 3:</td>
<td>8+,22−</td>
<td>.27+ .73−</td>
</tr>
</tbody>
</table>
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:
- \( \overline{\Phi}, \overline{\Phi} \), XOR
- \((A \overline{\Phi} B) (C \overline{\Phi} D \overline{\Phi} E)\)
- M 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. $A$ is 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?

Which attribute is best:
- $A_1$?
  - $t$: $[21+, 5-]$  
  - $f$: $[8+, 30-]$  
- $A_2$?
  - $t$: $[18+, 33-]$  
  - $f$: $[11+, 2-]$
Measuring Entropy

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

Entropy measures the impurity of $S$

$\text{Entropy}(S) = -p_\oplus \log p_\oplus - p \log p$

Entropy Function

![Graph showing the entropy function for different values of $p_\oplus$]
Entropy

\[ \text{Entropy}(S) = \text{expected number of bits needed to encode class (\(\oplus\) or \(\ominus\)) 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 of a random member of \(S\):

\[ p_\oplus (- \log p_\oplus -) + p (- \log p) \]

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?

[29+, 35-]  A1=?  [29+, 35-]  A2=?

[21+, 5-]  t  [8+, 30-]  f

[18+, 33-]  t  [11+, 2-]  f

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>
Selecting the Next Attribute

Which attribute is the best classifier?

Gain(S, Humidity) = .940 - (7/14).985 - (7/14).592 = .151
Gain(S, Wind) = .940 - (8/14).811 - (6/14)1.0 = .048

Attribute Bottom Left?

{D1, D2, ..., D14}
[9+,5-]

Outlook

Sunny

Overcast

Rain

{D1,D2,D8,D9,D11}
[2+,3-]

Yes

{D3,D7,D12,D13}
[4+,0-]

{D4,D5,D6,D10,D14}
[3+,2-]
Comparing Attributes

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

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

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

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

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

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

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

![Graph 1: Accuracy vs. Size of Tree (number of nodes)]

Overfitting in Learning

![Graph 2: Accuracy vs. Size of Tree (number of nodes)]
Avoiding Overfitting

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
  \[ \text{size}(\text{tree}) + \text{size}(\text{misclassifications}(\text{tree})) \]

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 Pruning

- 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 Tree to Rules

The 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

- Temp = 82.5
- (Temp > 72.3) = t, f

<table>
<thead>
<tr>
<th>Temp</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 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

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

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

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

Consider
• medical diagnosis, 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)
  \[ \text{Gain}^2(S,A)/\text{Cost}(A) \]
• Nunez (1988) [w in [0,1]: importance]
  \[ (2^{\text{Gain}(S,A)-1})/(\text{Cost}(A)+1)^w \]

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