Machine Learning III
Decision Tree Induction

CSE 473

Two Strategies for ML

- **Restriction bias**: use prior knowledge to specify a restricted hypothesis space.
  - Version space algorithm over conjunctions.
- **Preference bias**: use a broad hypothesis space, but impose an ordering on the hypotheses.
  - Decision trees.

Decision Trees

- **Convenient Representation**
  - Developed with learning in mind
  - Deterministic
- **Expressive**
  - Equivalent to propositional DNF
  - Handles discrete and continuous parameters
- **Simple learning algorithm**
  - Handles noise well
  - Classify as follows
    - Constructive (build DT by adding nodes)
    - Eager
    - Batch (but incremental versions exist)

Concept Learning

- **E.g. Learn concept “Edible mushroom”**
  - Target Function has two values: T or F
- **Represent concepts as decision trees**
- **Use hill climbing search**
- **Thru space of decision trees**
  - Start with simple concept
  - Refine it into a complex concept as needed

Experience: “Good day for tennis”

<table>
<thead>
<tr>
<th>Day Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>s</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>c</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>m</td>
<td>h</td>
<td>w</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>c</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>m</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>m</td>
<td>n</td>
<td>s</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>m</td>
<td>h</td>
<td>s</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>w</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>s</td>
</tr>
</tbody>
</table>
Decision Tree Representation

Good day for tennis?

Leaves = classification
Arcs = choice of value for parent attribute

Outlook
- Sunny
- Overcast
- Rain

Humidity
- Normal
- High

Wind
- Strong
- Weak

Yes
No

Decision tree is equivalent to logic in disjunctive normal form
G-Day ⇔ (Sunny ∧ Normal) ∨ Overcast ∨ (Rain ∧ Weak)

Decision Tree Hypothesis Space

If the features are continuous, internal nodes may test the value of a feature against a threshold.

Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows:
- depth 1 ("decision stump") can represent any boolean function of one feature.
- depth 2 Any boolean function of two features, some boolean functions involving three features (e.g., (x₁ ∧ x₂) ∨ (¬x₁ ∧ ¬x₂))
- etc.

DT Learning as Search

• Nodes
- Decision Trees
• Operators
- Tree Refinement: Sprouting the tree
• Initial node
- Smallest tree possible: a single leaf
• Heuristic?
- Information Gain
• Goal?
- Best tree possible (???)

What is the Simplest Tree?

How good?

[10+, 4-] Means: correct on 10 examples, incorrect on 4 examples
Successors

Yes

Humid

Wind

Outlook

Temp

Which attribute should we use to split?

To be decided:

- How to choose best attribute?
  - Information gain
  - Entropy (disorder)
- When to stop growing tree?

Disorder is bad
Homogeneity is good

Entropy (disorder) is bad
Homogeneity is good

- Let $S$ be a set of examples
- $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$
  where $P$ is proportion of pos example and $N$ is proportion of neg examples and $0 \log 0 = 0$
- Example: $S$ has 10 pos and 4 neg
  $\text{Entropy}([10+, 4-]) = -(10/14) \log_2(10/14) - (4/14) \log_2(4/14)$
  $= 0.863$

Information Gain

- Measure of expected reduction in entropy
- Resulting from splitting along an attribute

$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} (|S_v| / |S|) \text{Entropy}(S_v)$
**Gain of Splitting on Wind**

Values(wind)=weak, strong

\[ S = \{10+, 4-\} \]

\[ S_{\text{weak}} = \{6+, 2-\} \]

\[ S_{\text{s}} = \{3+, 3-\} \]

\[ \text{Gain}(S, \text{wind}) = \text{Entropy}(S) - \sum_{v \in \{\text{weak, s}\}} (|S_v| / |S|) \text{Entropy}(S_v) \]

\[ = \text{Entropy}(S) - 8/14 \text{Entropy}(S_{\text{weak}}) - 6/14 \text{Entropy}(S_{\text{s}}) \]

\[ = 0.863 - (8/14) 0.811 - (6/14) 1.00 \]

\[ = -0.029 \]

\[ v \% \{\text{weak, s}\} \]

\[ S_{\text{weak}} = \{6+, 2-\} \]

\[ S_{\text{s}} = \{3+, 3-\} \]

**Gain of Splitting on Humidity**

Gain(S, Humid) \[=0.375\]

Gain(S, Outlook) \[=0.401\]

Gain(S, Temp) \[=-0.029\]

**Is...**

- Entropy([4+, 3-]) = .985
- Entropy([7,0]) =
- Gain = 0.863 - .985/2 = 0.375

**Evaluating Attributes**

**Resulting Tree ....**

Good day for tennis?

\[ \text{Outlook} \]

\[ \text{Sunny} \quad \text{Overcast} \quad \text{Rain} \]

\[ \text{No} \quad [2+, 3-] \quad \text{Yes} \quad [4+] \quad \text{No} \quad [2+, 3-] \]

**Recurse!**

\[ \text{Day} \quad \text{Temp} \quad \text{Humid} \quad \text{Wind} \quad \text{Tennis?} \]

\[ d1 \quad h \quad h \quad \text{weak} \quad n \]
\[ d2 \quad h \quad h \quad s \quad n \]
\[ d8 \quad m \quad h \quad \text{weak} \quad n \]
\[ d9 \quad c \quad n \quad \text{weak} \quad \text{yes} \]
\[ d11 \quad m \quad n \quad s \quad \text{yes} \]
One Step Later...

Outlook
- Sunny
- Overcast
- Rain

Humidity
- Yes
- No

Yes [4+]
No [2+, 3-]

Decision Tree Algorithm

**BuildTree**(TrainingData)
**Split**(TrainingData)

**Split**(D)
- If (all points in D are of the same class)
  - Then Return
- For each attribute A
  - Evaluate splits on attribute A
  - Use best split to partition D into D1, D2

**Movie Recommendation**

- **Features?**

<table>
<thead>
<tr>
<th>Rambo</th>
<th>Matrix</th>
<th>Rambo 2</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Issues**

- **Content vs. Social**
- **Non-Boolean Attributes**
- **Missing Data**
- **Scaling up**

**Missing Data 1**

- Don't use this instance for learning?
- Assign attribute ...
  - most common value at node, or
  - most common value, ... given classification

**Fractional Values**

- 75% h and 25% n
- Use in gain calculations
- Further subdivide if other missing attributes
- Same approach to classify test ex with missing attr
  - Classification is most probable classification
  - Summing over leaves where it got divided

- [0.75+, 3-]
- [1.25+, 0-]
Non-Boolean Features

• Features with multiple discrete values
  Construct a multi-way split
  Test for one value vs. all of the others?
  Group values into two disjoint subsets?

• Real-valued Features
  Discretize?
  Consider a threshold split using observed values?

Attributes with many values

Problem:
• If attribute has many values, Gain will select it
• Imagine using Date = Jun.3.1996 as attribute

• So many values that it
  Divides examples into tiny sets
  Each set likely uniform → high info gain
  But poor predictor...
• Need to penalize these attributes

One approach: Gain ratio

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

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

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

\( \text{SplitInfo} \cong \text{entropy of } S \text{ wrt values of } A \)

\( \downarrow \) attribs with entropy of \( S \text{ wrt target value} \)

e.g. if \( A \) splits \( S \) uniformly into \( n \) sets

\( \text{SplitInformation} = \log_2(n) \ldots = 1 \text{ for Boolean} \)

Machine Learning Outline

• Machine learning:
• Supervised learning
• Overfitting
  What is the problem?
  Reduced error pruning
• Ensembles of classifiers

Overfitting

\[ \text{Accuracy} \]

On training data

\[ \text{On test data} \]

Number of Nodes in Decision tree

Overfitting 2

\[ \text{error rate} \]

test data

training data

model complexity

Figure from w.w.cohen
Overfitting...

- DT is *overfit* when exists another DT and DT has *smaller* error on training examples, but DT has *bigger* error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small

---

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
- Add complexity penalty to performance measure

---

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

---

Machine Learning Outline

- *Machine learning:*
- *Supervised learning*
- *Overfitting*
- *Ensembles of classifiers*
  - Bagging
  - Cross-validated committees
  - Boosting