Machine Learning III
Decision Tree Induction
CSE 473
Machine Learning Outline

• Machine learning:
  √ Function approximation
  √ Bias

• Supervised learning
  √ Classifiers & concept learning
  √ Version Spaces (restriction bias)
    Decision-trees induction (pref bias)

• Overfitting

• Ensembles of classifiers
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</th>
<th>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>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>c</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>c</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>m</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>c</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>m</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>m</td>
<td>h</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>m</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
</tbody>
</table>
Decision Tree Representation

Good day for tennis?

Leaves = classification
Arcs = choice of value for parent attribute

Humidity

Sunny

Overcast

Rain

Wind

Yes

Normal

High

Strong

Weak

Yes

No

No

Yes

Decision tree is equivalent to logic in disjunctive normal form

G-Day \iff (Sunny \land Normal) \lor Overcast \lor (Rain \land Weak)
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 Tree Hypothesis Space

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

```
    Outlook
    /    \
Sunny  Overcast  Rain
    |      |      |
Humidity  Yes  Wind
    |      |
  > 75%  <= 75%  > 20  <= 20
    |      |      |
 No    Yes    No    Yes
```
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., \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)\))

- **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?

<table>
<thead>
<tr>
<th>Day Outlook</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>h</td>
<td>n</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>m</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>

How good?

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

Bad

No
Better

Good

© Daniel S. Weld
Entropy (disorder) is bad
Homogeneity is good

- Let $S$ be a set of examples
- 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
  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

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

Where \( \text{Entropy}(S) = -P \log_2(P) - N \log_2(N) \)
Gain of Splitting on Wind

Values(wind)=weak, strong
S = [10+, 4-]
S_{weak} = [6+, 2-]
S_s = [3+, 3-]

Gain(S, wind)
= Entropy(S) - \sum (|S_v| / |S|) Entropy(S_v)
\quad \forall v \in \{\text{weak, } s\}

= Entropy(S) - 8/14 \text{ Entropy}(S_{\text{weak}})
- 6/14 \text{ Entropy}(S_s)
= 0.863 - (8/14) 0.811 - (6/14) 1.00
= -0.029

© Daniel S. Weld
Gain of Split on Humidity

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

\[
\text{Gain}(S, A) = \text{Entropy}(S) - \sum \left( \frac{|S_v|}{|S|} \right) \text{Entropy}(S_v)
\]

\[
\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)
\]
Is...

- Entropy([4+,3-]) = .985
- Entropy([7,0]) =
- Gain = 0.863 - 0.985/2 = 0.375
Evaluating Attributes

Gain(S, Humid) = 0.375
Gain(S, Outlook) = 0.401
Gain(S, Temp) = 0.029
Gain(S, Wind) = -0.029

Value is actually different than this, but ignore this detail.
Good day for tennis?

**Outlook**

- Sunny
  - No: [2+, 3-]

- Overcast
  - Yes: [4+]

- Rain
  - No: [2+, 3-]
Recurse!

<table>
<thead>
<tr>
<th>Day</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>h</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d8</td>
<td>m</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>c</td>
<td>n</td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>
One Step Later...

- **Outlook**
  - Sunny
  - Overcast
  - Rain

- **Humidity**
  - Normal
    - Yes [2+]  
    - No [3-]
  - High
    - 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
Split (D1)
Split (D2)
Movie Recommendation

• Features?

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rambo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matrix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rambo 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>
Issues

• Content vs. Social

• Non-Boolean Attributes

• Missing Data

• Scaling up
### Missing Data 1

<table>
<thead>
<tr>
<th>Day</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>h</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d8</td>
<td>m</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>c</td>
<td></td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Day</th>
<th>Temp</th>
<th>Humid</th>
<th>Wind</th>
<th>Tennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>h</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>h</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d8</td>
<td>m</td>
<td>h</td>
<td>weak</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>c</td>
<td></td>
<td>weak</td>
<td>yes</td>
</tr>
<tr>
<td>d11</td>
<td>m</td>
<td>n</td>
<td>s</td>
<td>yes</td>
</tr>
</tbody>
</table>

- 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
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 $\rightarrow$ high info gain
  But poor predictor...
- Need to penalize these attributes
One approach: Gain ratio

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

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

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

(Contrast with entropy of \( S \) wrt target value)

\( \downarrow \) attribs with many uniformly distrib values

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

\( \text{SplitInformation} = \log_2(n) \)... = 1 for Boolean
• Machine learning:
• Supervised learning
• Overfitting
  What is the problem?
  Reduced error pruning
• Ensembles of classifiers
Overfitting

On training data
On test data

Accuracy

0.9
0.8
0.7
0.6

Number of Nodes in Decision tree
Overfitting 2

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
Effect of Reduced-Error Pruning

The graph shows the accuracy of a model as a function of the size of the tree (number of nodes). The accuracy is measured on both training data and test data. The graph indicates that pruning the tree can reduce the size while maintaining or improving accuracy. The annotation "Cut tree back to here" suggests a point where the tree size should be reduced for optimal performance.
Machine Learning Outline

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