Today:
• Big Picture 1
• Decision tree learning
• Overfitting
• Cross validation

Suggested Reading:
• Mitchell Chapter 3
• Bishop 14.4

Course Webpage: http://mlcourse.org
Decision Tree Learning

Problem Setting:
• Set of possible instances $X$
  – each instance $x$ in $X$ is vector of discrete-valued features $x = <x_1, x_2 \ldots x_n>$
• Unknown target function $f : X \rightarrow Y$
  – $Y$ is discrete-valued
• Set of function hypotheses $H = \{ h | h : X \rightarrow Y \}$
  – each hypothesis $h$ is a decision tree

Input:
• Training examples $\{<x^{(i)}, y^{(i)}>\}$ of unknown target function $f$

Output:
• Hypothesis $h \in H$ that best approximates target function $f$
Top-Down Induction of Decision Trees

node = Root

Main loop:
1. $A \leftarrow$ 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?

$A_1 =$

\begin{align*}
[29+, 35-] & \quad t \quad \substack{[21+, 5-] \quad 8+, 30- \quad f} \\
[29+, 35-] & \quad t \quad \substack{[18+, 33-] \quad 11+, 2- \quad f}
\end{align*}

$A_2 =$
Final Decision Tree for
\[ f: \langle \text{Outlook, Temperature, Humidity, Wind} \rangle \rightarrow \text{PlayTennis?} \]
Which Tree Should We Output?

- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Occam’s razor: prefer the simplest hypothesis that fits the data
Two zero-error decision trees:

\( f: \langle \text{Outlook, Temperature, Humidity, Wind} \rangle \rightarrow \text{PlayTennis?} \)
Consider learning function $f: X \rightarrow \{0, 1\}$, where $X = <x_1 \ x_2 \ ... \ x_n>$ and each $x_i$ is Boolean valued (0 or 1).

Hypotheses $H$ considered by the learning algorithm are all possible decision trees over $x_1 \ ... \ x_n$.
Suppose \( X = \langle x_1, x_2, x_3 \rangle \)
Where each \( x_i \) is Boolean (0 or 1)

This *full tree* sorts each distinct \( X \) to its own leaf

How many different ways can this tree label the examples?

(i.e., how many different functions can we represent with this full tree structure?)
Why Prefer Short Hypotheses? (Occam’s Razor)

Arguments in favor:

Arguments opposed:
Why Prefer Short Hypotheses? (Occam’s Razor)

Argument in favor:
• Fewer short hypotheses than long ones → a short hypothesis that fits the data is less likely to be a statistical coincidence → highly probable that some sufficiently complex hypothesis will fit the data

Argument opposed:
• Also fewer hypotheses with prime number of nodes and attributes beginning with “Z”
• What’s so special about “short” hypotheses?

Bottom line: This bias toward shorter trees (simpler functions) is widely used, and widely successful
Overfitting
Noisy training data* can lead to overfitting, and worse future predictions.

Original data and resulting tree:

<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>

Adding one noisy example:

D15 Rain Hot Normal Strong Yes
Noisy training data* can lead to overfitting, and worse future predictions

Original data and resulting tree:

Adding one noisy example:
D15 Rain Hot Normal Strong Yes

* Even without noise, overfitting occurs due to statistical anomalies in training data
Overfitting

Consider a hypothesis $h$ and its

- Error rate over training data: $\text{error}_{\text{train}}(h)$
- True error rate over all data: $\text{error}_{\text{true}}(h)$
Overfitting

Consider a hypothesis $h$ and its

- Error rate over training data: $\text{error}_{\text{train}}(h)$
- True error rate over all data: $\text{error}_{\text{true}}(h)$

We say $h$ overfits the training data if

$$\text{error}_{\text{true}}(h) > \text{error}_{\text{train}}(h)$$

Amount of overfitting =

$$\text{error}_{\text{true}}(h) - \text{error}_{\text{train}}(h)$$
Overfitting in Decision Tree Learning

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
Reduced Error Pruning

• Split data into *training set* and *validation set*
• Train a tree to classify *training set* as well as possible

• Do until further pruning reduces *validation set* accuracy:
  1. For each internal tree node, consider making it a leaf node (pruning the tree below it)
  2. Greedily chose the above pruning step that best improves error over *validation set*

Produces smallest version of the most accurate (over the validation set) pruned tree
Effect of Reduced-Error Pruning

![Graph showing the effect of reduced-error pruning on accuracy over the size of the tree.](image-url)
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)
Converting A Tree to Rules

```
Outlook
  Sunny
    Humidity
      High
        No
      Normal
        Yes
  Overcast
    Rain
      Yes
  Wind
    Strong
      No
    Weak
      Yes
```
Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

<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>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Decision Forests

Key idea:
1. learn a collection of many trees
2. classify by taking a weighted vote of the trees

Empirically successful. Widely used in industry.
• human pose recognition in Microsoft kinect
• medical imaging – cortical parcellation
• classify disease from gene expression data

How to train different trees
1. Train on different random subsets of data
2. Randomize the choice of decision nodes
Decision Forests often use an ensemble of Decision Stumps

A decision stump is simply a decision tree with depth 1:
You should know:

- Well posed function approximation problems:
  - Instance space, $X$
  - Sample of labeled training data $\{ <x^{(i)}, y^{(i)}> \}$
  - Hypothesis space, $H = \{ f: X \rightarrow Y \}$

- Learning is a search/optimization problem over $H$
  - Various objective functions to define the goal
    - minimize training error
    - minimize validation error
    - among hypotheses that minimize error, select simplest tree (?)

- Decision tree learning
  - Greedy top-down learning of decision trees (ID3, C4.5, ...)
  - Overfitting and post-pruning
  - Extensions… to continuous values, probabilistic classification
Further Reading…
Questions to think about (0)

• How can we use decision trees to make probabilistic predictions (i.e., $P(Y=1|X)$ instead of simply predict that $Y=1$ or $Y=0$?

[Hint: go back and look at the tree for predicting C-section birth risk]
Questions to think about

• Consider target function $f: \langle x_1, x_2 \rangle \rightarrow y$, where $x_1$ and $x_2$ are real-valued, $y$ is Boolean (0 or 1)
  
  – What is the set of decision surfaces describable with decision trees that use each attribute at most once?
Questions to think about (2)

• ID3 and C4.5 are heuristic algorithms that search through the space of decision trees. Why not just do an exhaustive search over all possible trees?
Questions to think about (3)

• Why use Information Gain to select attributes in decision trees? What other criteria seem reasonable, and what are the tradeoffs in making this choice?
Questions to think about (4)

• What if one of the attributes of an instance is unobserved, but you want to apply the decision tree anyway. (e.g., a decision tree for medical diagnosis, but this patient didn’t get one of the blood tests used). What can you do to apply the tree anyway?

[hint: you know how to train trees to predict target attributes…]