Overview of Decision Trees, Ensemble Methods and Reinforcement Learning

CMSC 678

UMBC
Outline

Decision Trees

Ensemble Methods
  Bagging
  Random Forests

Reinforcement Learning
Decision Trees

“20 Questions”: http://20q.net/

Goals: 1. Figure out what questions to ask
  2. In what order
  3. Determine how many questions are enough
  4. What to predict at the end

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Rating is the label

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features

Rating is the label

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
## Example: Learning a decision tree

Course ratings dataset

**Questions** are features

**Responses** are feature values

**Rating** is the **label**

Idea: Predict the label by forming a tree where each node branches on values of particular **features**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features
Responses are feature values
Rating is the label

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features
Responses are feature values
Rating is the label

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features
Responses are feature values
Rating is the label

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>+2</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>+1</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>+1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-1</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-1</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>-2</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>y</td>
</tr>
</tbody>
</table>

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features
Responses are feature values
Rating is the label

Adapted from Hamed Pirsiavash
Example: Learning a decision tree

Course ratings dataset

Questions are features
Responses are feature values
Rating is the label

Easy?

Easy: yes
Easy: no

AI?
AI: yes
AI: no

Sys?
Sys: yes
Sys: no

Rating
+2 y y n y n
+2 y y n y n
+2 n y n n n
+2 n n n y n
+2 n y n n y
+1 y y n n y
+1 y y n n n
+1 n y n y n
+1 n y n n n
0 n n n n y
0 y n n y y
0 y n n n n
0 y y y y n
-1 y y y n y
-1 n n y y n
-1 n n y n y
-1 y n y n y
-2 n n y y n
-2 n y y n y
-2 y n y n n
-2 y n y n y

Adapted from Hamed Pirsiavash
Predicting with a Decision Tree is Done Easily and Recursively

**Algorithm 2**  
\textsc{DecisionTreeTest}(tree, test point)

1. \textbf{if} tree is of the form \textsc{Leaf}(guess) \textbf{then}
   \hspace{1cm} \textbf{return} guess

2. \textbf{else if} tree is of the form \textsc{Node}(f, left, right) \textbf{then}
   \hspace{1cm} \textbf{if} \ f = yes \textbf{ in} test point \textbf{then}
   \hspace{1cm} \hspace{1cm} \textbf{return} \textsc{DecisionTreeTest}(left, test point)
   \hspace{1cm} \textbf{else}
   \hspace{1cm} \hspace{1cm} \textbf{return} \textsc{DecisionTreeTest}(right, test point)
   \hspace{1cm} \textbf{end if}

3. \textbf{end if}
There Are Many Ways to Learn a Decision Tree

1. Greedy/Count: What is the most accurate feature at each decision point?
   1. See CIML Ch. 1 (and next slides)

2. Maximize *information gain* at each step
   1. Most popular approaches: ID3, C4.5

3. Account for statistical significance
   1. Example: Chi-square automatic interaction detection (CHAID)

4. Other task-specific ones (including clustering based)
Algorithm 1 \texttt{DecisionTreeTrain}(data, remaining features)

1: \texttt{guess} $\leftarrow$ most frequent answer in \texttt{data} \hspace{1em} // default answer for this data
2: \textbf{if} the labels in \texttt{data} are unambiguous \textbf{then}
3: \hspace{1em} \texttt{return} \hspace{1em} \texttt{Leaf}($\texttt{guess}$) \hspace{1em} // base case: no need to split further
4: \textbf{else if} remaining features is empty \textbf{then}
5: \hspace{1em} \texttt{return} \hspace{1em} \texttt{Leaf}($\texttt{guess}$) \hspace{1em} // base case: cannot split further
6: \textbf{else} \hspace{1em} // we need to query more features
7: \hspace{2em} \textbf{for all} \hspace{1em} \texttt{f} $\in$ remaining features \textbf{do}
8: \hspace{3em} \texttt{NO} $\leftarrow$ the subset of \texttt{data} on which \texttt{f}=\texttt{no}
9: \hspace{3em} \texttt{YES} $\leftarrow$ the subset of \texttt{data} on which \texttt{f}=\texttt{yes}
10: \hspace{3em} \texttt{score}[$\texttt{f}$] $\leftarrow$ \# of majority vote answers in \texttt{NO}
11: \hspace{3em} \hspace{3em} + \# of majority vote answers in \texttt{YES}
12: \hspace{3em} \hspace{3em} // the accuracy we would get if we only queried on \texttt{f}
13: \hspace{2em} \textbf{end for}
14: \hspace{2em} \texttt{f} $\leftarrow$ the feature with maximal \texttt{score}($\texttt{f}$)
15: \hspace{2em} \texttt{NO} $\leftarrow$ the subset of \texttt{data} on which \texttt{f}=\texttt{no}
16: \hspace{2em} \texttt{YES} $\leftarrow$ the subset of \texttt{data} on which \texttt{f}=\texttt{yes}
17: \hspace{2em} \texttt{left} $\leftarrow$ \texttt{DecisionTreeTrain}(\texttt{NO, remaining features} \setminus \{\texttt{f}\})
18: \hspace{2em} \texttt{right} $\leftarrow$ \texttt{DecisionTreeTrain}(\texttt{YES, remaining features} \setminus \{\texttt{f}\})
19: \hspace{2em} \texttt{return} \hspace{1em} \texttt{Node}($\texttt{f}$, \texttt{left}, \texttt{right})
20: \textbf{end if}
Algorithm 1 \textbf{DecisionTreeTrain}(data, remaining features)

1: \texttt{guess} $\leftarrow$ most frequent answer in data \hspace{1em} // default answer for this data
2: \textbf{if} the labels in data are unambiguous \textbf{then}
3: \hspace{1em} \texttt{return} \ Leaf(\texttt{guess}) \hspace{1em} // base case: no need to split further
4: \textbf{else if} remaining features is empty \textbf{then}
5: \hspace{1em} \texttt{return} \ Leaf(\texttt{guess}) \hspace{1em} // base case: cannot split further
6: \textbf{else} \hspace{1em} // we need to query more features
7: \hspace{2em} \textbf{for all} \texttt{f} $\in$ remaining features \textbf{do}
8: \hspace{3em} \texttt{NO} $\leftarrow$ the subset of data on which \texttt{f}=\texttt{no}
9: \hspace{3em} \texttt{YES} $\leftarrow$ the subset of data on which \texttt{f}=\texttt{yes}
10: \hspace{3em} \texttt{score}[\texttt{f}] $\leftarrow$ \# of majority vote answers in \texttt{NO} + \# of majority vote answers in \texttt{YES} \hspace{1em} // the accuracy we would get if we only queried on \texttt{f}
12: \hspace{2em} \textbf{end for}
13: \hspace{1em} \texttt{f} $\leftarrow$ the feature with maximal \texttt{score}(\texttt{f})
14: \hspace{1em} \texttt{NO} $\leftarrow$ the subset of data on which \texttt{f}=\texttt{no}
15: \hspace{1em} \texttt{YES} $\leftarrow$ the subset of data on which \texttt{f}=\texttt{yes}
16: \hspace{1em} \texttt{left} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{NO}, remaining features \setminus \{\texttt{f}\})
17: \hspace{1em} \texttt{right} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{YES}, remaining features \setminus \{\texttt{f}\})
18: \hspace{1em} \texttt{return} \ Node(\texttt{f}, \texttt{left}, \texttt{right})
19: \textbf{end if}
Algorithm 1 DecisionTreeTrain\((data, remaining\ features)\)

1: \textit{guess} \leftarrow \text{most frequent answer in } data \quad \text{// default answer for this data}
2: \textbf{if} the labels in \textit{data} are unambiguous \textbf{then}
3: \hspace{1em} \textbf{return} \ \text{Leaf}(\textit{guess}) \quad \text{// base case: no need to split further}
4: \textbf{else if} \ \textit{remaining features} is empty \textbf{then}
5: \hspace{1em} \textbf{return} \ \text{Leaf}(\textit{guess}) \quad \text{// base case: cannot split further}
6: \textbf{else} \quad \text{// we need to query more features}
7: \hspace{1em} \textbf{for all } f \in \textit{remaining features} \textbf{ do}
8: \hspace{2em} NO \leftarrow \text{the subset of } data \text{ on which } f = \text{no}
9: \hspace{2em} YES \leftarrow \text{the subset of } data \text{ on which } f = \text{yes}
10: \hspace{2em} \text{score}[f] \leftarrow \# \text{ of majority vote answers in } NO
11: \hspace{2em} \quad + \# \text{ of majority vote answers in } YES
12: \hspace{2em} \quad \text{// the accuracy we would get if we only queried on } f
13: \hspace{2em} \textbf{end for}
14: \hspace{1em} f \leftarrow \text{the feature with maximal } \text{score}(f)
15: \hspace{1em} NO \leftarrow \text{the subset of } data \text{ on which } f = \text{no}
16: \hspace{1em} YES \leftarrow \text{the subset of } data \text{ on which } f = \text{yes}
17: \hspace{1em} left \leftarrow \text{DecisionTreeTrain}(NO, \text{remaining features} \setminus \{f\})
18: \hspace{1em} right \leftarrow \text{DecisionTreeTrain}(YES, \text{remaining features} \setminus \{f\})
19: \hspace{1em} \textbf{return} \ \text{Node}(f, left, right)
20: \textbf{end if}
Algorithm 1 DecisionTreeTrain\((data, remaining\ features)\)

1: \(guess \leftarrow\) most frequent answer in \(data\) // default answer for this data
2: \textbf{if} the labels in \(data\) are unambiguous \textbf{then}
3: \hspace{1em} \textbf{return} \(\text{Leaf}(guess)\) // base case: no need to split further
4: \textbf{else if} remaining features is empty \textbf{then}
5: \hspace{1em} \textbf{return} \(\text{Leaf}(guess)\) // base case: cannot split further
6: \textbf{else} // we need to query more features
7: \hspace{1em} \textbf{for all} \(f \in \text{remaining features}\) \textbf{do}
8: \hspace{2em} \(NO \leftarrow\) the subset of \(data\) on which \(f=\text{no}\)
9: \hspace{2em} \(YES \leftarrow\) the subset of \(data\) on which \(f=\text{yes}\)
10: \hspace{2em} \(\text{score}[f] \leftarrow\) # of majority vote answers in \(NO\)
11: \hspace{2em} + # of majority vote answers in \(YES\)
12: \hspace{2em} // the accuracy we would get if we only queried on \(f\)
13: \hspace{1em} \textbf{end for}
14: \hspace{1em} \(f \leftarrow\) the feature with maximal \(\text{score}(f)\)
15: \hspace{1em} \(NO \leftarrow\) the subset of \(data\) on which \(f=\text{no}\)
16: \hspace{1em} \(YES \leftarrow\) the subset of \(data\) on which \(f=\text{yes}\)
17: \hspace{1em} \(\text{left} \leftarrow\) DecisionTreeTrain\((NO, \text{remaining features} \setminus \{f\})\)
18: \hspace{1em} \(\text{right} \leftarrow\) DecisionTreeTrain\((YES, \text{remaining features} \setminus \{f\})\)
19: \textbf{return} \(\text{Node}(f, \text{left}, \text{right})\)
\textbf{end if}
Outline

Decision Trees

**Ensemble Methods**
- Bagging
- Random Forests

Reinforcement Learning
Ensembles

Key Idea: “Wisdom of the crowd“

groups of people can often make better decisions than individuals

Apply this to ML

Learn multiple classifiers and combine their predictions
Combining Multiple Classifiers by Voting

Train several classifiers and take majority of predictions
Combining Multiple Classifiers by Voting

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging
Combining Multiple Classifiers by Voting

Train several classifiers and take majority of predictions

For regression use mean or median of the predictions

For ranking and collective classification use some form of averaging

A common family of approaches is called bagging
Option 1: Split the data into K pieces and train a classifier on each

Q: What can go wrong with option 1?
Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Q: What can go wrong with option 1?
A: Small sample → poor performance
Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Option 2: Bootstrap aggregation (bagging) resampling

Q: What can go wrong with option 1?
A: Small sample → poor performance
Bagging: Split the Data

Option 1: Split the data into K pieces and train a classifier on each

Option 2: Bootstrap aggregation (bagging) resampling

Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from $D$

Q: What can go wrong with option 1?

A: Small sample → poor performance
Bagging: Split the Data

Option 1: Split the data into $K$ pieces and train a classifier on each.

Option 2: Bootstrap aggregation (bagging) resampling

- Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from $D$.
- Train classifiers on each dataset and average their predictions.

Q: What can go wrong with option 1?
A: Small sample $\rightarrow$ poor performance.

Given a dataset $D$...

get new datasets $\hat{D}$ by random sampling with replacement from $D$.
Why does averaging work?

Averaging reduces the variance of estimators

\[ y = f(x) + \epsilon \]
\[ f(x) = \sin(\pi x) \]
\[ \epsilon = N(0, \sigma^2) \]
\[ \sigma = 0.1 \]
Why does averaging work?

Averaging reduces the **variance** of estimators

\[ y = f(x) + \epsilon \]
\[ f(x) = \sin(\pi x) \]
\[ \epsilon = N(0, \sigma^2) \]
\[ \sigma = 0.1 \]

\[ g_n(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n \]

---

**y**: observed data

**\( g_i \)**: Learned polynomial regression

**f**: Generating line

---

Courtesy Hamed Pirsiavash
Why does averaging work?

Averaging reduces the variance of estimators

\[ y = f(x) + \epsilon \]
\[ f(x) = \sin(\pi x) \]
\[ \epsilon \sim N(0, \sigma^2) \]
\[ \sigma = 0.1 \]

50 samples

\[ g_n(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_n x^n \]

Averaging is a form of regularization: each model can individually overfit but the average is able to overcome the overfitting

Courtesy Hamed Pirsiavash
Bagging Decision Trees

How would it work?
Bagging Decision Trees

How would it work?

Bootstrap sample $S$ samples $\{(X_1, Y_1), \ldots, (X_S, Y_S)\}$
Train a tree $t_s$ on $(X_s, Y_s)$
At test time: $\hat{y} = \text{avg}(t_1(x), \ldots t_S(x))$
Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size $k$ and pick the best among these.

Train decision trees of depth $d$.

Average results from multiple randomly trained trees.

Q: What’s the difference between bagging decision trees and random forests?
Random Forests

Bagging trees with one modification

At each split point, choose a random subset of features of size \( k \) and pick the best among these

Train decision trees of depth \( d \)

Average results from multiple randomly trained trees

Q: What’s the difference between bagging decision trees and random forests?

A: Bagging ➔ highly correlated trees (reuse good features)
Random Forests: Human Pose Estimation
(Shotton et al., CVPR 2011)

**Training:** 3 trees, 20 deep, 300k training images per tree, 2000 training example pixels per image, 2000 candidate features $\theta$, and 50 candidate thresholds $\tau$ per feature (Takes about 1 day on a 1000 core cluster)
Outline

Decision Trees

Ensemble Methods
  Bagging
  Random Forests

Reinforcement Learning
There’s an entire book!

Reinforcement Learning

agent

environment

Robot image: openclipart.org

Reinforcement Learning

agent

take action

environment

Robot image: openclipart.org

Reinforcement Learning

- **Agent**
- **Environment**
- **Take action**
- **Get new state and/or reward**

Robot image: openclipart.org

Reinforcement Learning

agent

take action

get new state and/or reward

environment

Robot image: openclipart.org

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, p, \gamma)\)
Markov Decision Process:
Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, p, \gamma)\)

- set of possible states
- set of possible actions
- get new state and/or reward
- take action
- environment
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- $\mathcal{S}$: set of possible states
- $\mathcal{A}$: set of possible actions
- $\mathcal{R}$: reward function
- $p$: transition probability
- $\gamma$: discount factor

Robot image: openclipart.org

Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: 

\((\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)\)

- set of possible states
- reward of (state, action) pairs
- state-action transition distribution
- set of possible actions
Markov Decision Process:
Formalizing Reinforcement Learning

agent
take action
get new state and/or reward
environment

Markov Decision Process: $\left(S, A, R, p, \gamma\right)$

- set of possible states
- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor
Markov Decision Process: (\mathcal{S}, \mathcal{A}, \mathcal{R}, \pi, \gamma)

- set of possible states
- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor

Start in initial state \( s_0 \)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: 

\[ (S, A, R, \pi, \gamma) \]

- set of possible states
- set of possible actions
- reward of (state, action) pairs
- state-action transition distribution
- discount factor

Start in initial state \( s_0 \)
for \( t = 1 \) to ...
choose action \( a_t \)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \( (S, A, R, \pi, \gamma) \)

- set of possible states
- reward of (state, action) pairs
- discount factor
- state-action transition distribution
- set of possible actions

Start in initial state \( s_0 \)
for \( t = 1 \) to ...:

choose action \( a_t \)
“move” to next state \( s_t \sim \pi(\cdot|s_{t-1}, a_t) \)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- set of possible states
- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to ...:
    choose action \(a_t\)
    “move” to next state \(s_t \sim \pi(\cdot|s_{t-1}, a_t)\)
    get reward \(r_t = R(s_t, a_t)\)
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- set of possible states
- set of possible actions
- state-action transition distribution
- reward of (state, action) pairs
- discount factor

Start in initial state \(s_0\)
for \(t = 1\) to \(...\):
  - choose action \(a_t\)
  - “move” to next state \(s_t \sim \pi(\cdot|s_{t-1}, a_t)\)
  - get reward \(r_t = R(s_t, a_t)\)

objective: maximize time-discounted reward
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

- \(S\): set of possible states
- \(A\): set of possible actions
- \(R\): reward of (state, action) pairs
- \(\pi\): state-action transition distribution
- \(\gamma\): discount factor

Start in initial state \(s_0\)
for \(t = 1\) to ...:
- choose action \(a_t\)
- “move” to next state \(s_t \sim \pi(\cdot|s_{t-1}, a_t)\)
- get reward \(r_t = R(s_t, a_t)\)

Objective: maximize discounted reward

\[
\max_{\pi} \sum_{t>0} \gamma^t r_t
\]
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: 

\[(S, \mathcal{A}, \mathcal{R}, \pi, \gamma)\]

- set of possible states
- reward of (state, action) pairs
- state-action transition distribution
- set of possible actions
- discount factor

Start in initial state \(s_0\)

for \(t = 1\) to ...

choose action \(a_t\)

"move" to next state \(s_t \sim \pi(\cdot|s_{t-1}, a_t)\)

get reward \(r_t = R(s_t, a_t)\)

objective: maximize discounted reward

\[
\max_{\pi} \sum_{t>0} \gamma^t r_t
\]

"solution": the policy \(\pi^*\) that maximizes the expected (average) time-discounted reward
Markov Decision Process: Formalizing Reinforcement Learning

Markov Decision Process: \((S, A, R, \pi, \gamma)\)

Start in initial state \(s_0\)

for \(t = 1\) to ...:

choose action \(a_t\)

“move” to next state \(s_t \sim \pi(\cdot | s_{t-1}, a_t)\)

get reward \(r_t = R(s_t, a_t)\)

objective: maximize discounted reward

\[
\max_{\pi} \sum_{t>0} \gamma^t r_t
\]

“solution” \(\pi^* = \arg\max_{\pi} \mathbb{E} \left[ \sum_{t>0} \gamma^t r_t ; \pi \right]\)
Designing Rewards is *Highly* Task Dependent

Rewards indicate **what** we want to accomplish, NOT **how** we want to accomplish it

**shaping**
positive reward often very “far away”
rewards for achieving subgoals (domain knowledge)
also: adjust initial policy or initial value function

Example: robot in a maze
episodic task, not discounted, +1 when out, 0 for each step

Example: chess
GOOD: +1 for winning, -1 losing
BAD: +0.25 for taking opponent’s pieces
  high reward even when lose

Slide courtesy/adapted: Peter Bodík
Overview: Learning Strategies

Dynamic Programming

Q-learning

Monte Carlo approaches
Q-learning

\[ Q : (s, a) \rightarrow \mathbb{R} \]

Goal: learn a function that computes a “goodness” score for taking a particular action \( a \) in state \( s \)
Deep/Neural Q-learning

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

Approach: Form (and learn) a neural network to model our optimal Q function
Deep/Neural Q-learning

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

- Learn weights (parameters) \( \theta \) of our neural network

Approach: Form (and learn) a neural network to model our optimal Q function
Outline

Decision Trees

Ensemble Methods
  Bagging
  Random Forests

Reinforcement Learning