Decision Trees
+
k-Nearest Neighbors
<table>
<thead>
<tr>
<th>Q:</th>
<th>Why don’t my entropy calculations match those on the slides?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:</td>
<td>H(Y) is conventionally reported in “bits” and computed using log base 2. e.g., H(Y) = (- P(Y=0) \log_2 P(Y=0) - P(Y=1) \log_2 P(Y=1))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q:</th>
<th>When and how do we decide to stop growing trees? What if the set of values an attribute could take was really large or even infinite?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:</td>
<td>We’ll address this question for discrete attributes today. If an attribute is real-valued, there’s a clever trick that only considers O(L) splits where L = # of values the attribute takes in the training set. Can you guess what it does?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q:</th>
<th>Why is entropy based on a sum of p(.) log p(.) terms?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:</td>
<td>We don’t have time for a full treatment of why it has to be this, but we can develop the right intuition with a few examples...</td>
</tr>
</tbody>
</table>
Reminders

• Homework 1: Background
  – Out: Wed, Jan 17
  – Due: Wed, Jan 24 at 11:59pm
  – unique policy for this assignment: we will grant (essentially) any and all extension requests

• Homework 2: Decision Trees
  – Out: Wed, Jan 24
  – Due: Mon, Feb 5 at 11:59pm
DECISION TREES
Tennis Example

Dataset:

<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>

Figure from Tom Mitchell
Tennis Example

Which attribute yields the best classifier?
Tennis Example

Which attribute yields the best classifier?
Tennis Example

Which attribute yields the best classifier?

Figure from Tom Mitchell
Tennis Example

Which attribute should be tested here?

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

\[
\begin{align*}
\text{Gain} (S_{\text{Sunny}}, \text{Humidity}) &= .970 - (3/5)0.0 - (2/5)0.0 = .970 \\
\text{Gain} (S_{\text{Sunny}}, \text{Temperature}) &= .970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = .570 \\
\text{Gain} (S_{\text{Sunny}}, \text{Wind}) &= .970 - (2/5)1.0 - (3/5)918 = .019
\end{align*}
\]
Decision Tree Learning Example

In-Class Exercise
1. Which attribute would misclassification rate select for the next split?
2. Which attribute would information gain select for the next split?

Dataset:
Output Y, Attributes A and B

<table>
<thead>
<tr>
<th>Y</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
## Decision Tree Learning Example

**Dataset:**
Output Y, Attributes A and B

<table>
<thead>
<tr>
<th>Y</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Decision Trees

Chalkboard

– ID3 as Search
– Inductive Bias of Decision Trees
– Occam’s Razor
Overfitting and Underfitting

**Underfitting**
- The model...
  - is too simple
  - is unable captures the trends in the data
  - exhibits too much bias
- *Example*: majority-vote classifier (i.e. depth-zero decision tree)
- *Example*: a toddler (that has **not** attended medical school) attempting to carry out medical diagnosis

**Overfitting**
- The model...
  - is too complex
  - is fitting the noise in the data
  - or fitting random statistical fluctuations inherent in the “sample” of training data
  - does not have enough bias
- *Example*: our “memorizer” algorithm responding to an “orange shirt” attribute
- *Example*: medical student who simply memorizes patient case studies, but does not understand how to apply knowledge to new patients
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

Figure from Tom Mitchell
How to Avoid Overfitting?

For Decision Trees...

1. Do not grow tree beyond some **maximum depth**
2. Do not split if splitting criterion (e.g. Info. Gain) is **below some threshold**
3. Stop growing when the split is **not statistically significant**
4. Grow the entire tree, then **prune**
Reduced-Error Pruning

Split data into \textit{training} and \textit{validation} set

Create tree that classifies \textit{training} set correctly

Do until further pruning is harmful:

1. Evaluate impact on \textit{validation} set of pruning each possible node (plus those below it)

2. Greedily remove the one that most improves \textit{validation} set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?
Effect of Reduced-Error Pruning

- Split data into training and validation set.
- Create tree that classifies training set correctly.

Slide from Tom Mitchell
Questions

• Will ID3 always include all the attributes in the tree?
• What if some attributes are real-valued? Can learning still be done efficiently?
• What if some attributes are missing?
Decision Trees (DTs) in the Wild

• DTs are one of the most popular classification methods for practical applications
  – Reason #1: The learned representation is **easy to explain** a non-ML person
  – Reason #2: They are **efficient** in both computation and memory

• DTs can be applied to a wide variety of problems including **classification, regression, density estimation**, etc.

• **Applications of DTs** include...
  – medicine, molecular biology, text classification, manufacturing, astronomy, agriculture, and many others

• **Decision Forests** learn many DTs from random subsets of features; the result is a very powerful example of an **ensemble method** (discussed later in the course)
You should be able to...

1. Implement Decision Tree training and prediction
2. Use effective splitting criteria for Decision Trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in Decision Tree learning
KNN Outline

• **Classification**
  – Binary classification
  – 2D examples
  – Decision rules / hypotheses

• **k-Nearest Neighbors (KNN)**
  – Nearest Neighbor classification
  – k-Nearest Neighbor classification
  – Distance functions
  – Case Study: KNN on Fisher Iris Data
  – Case Study: KNN on 2D Gaussian Data
  – Special cases
  – Choosing k

• **Experimental Design**
  – Train error vs. test error
  – Train / validation / test splits
  – Cross-validation
CLASSIFICATION
Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

<table>
<thead>
<tr>
<th>Species</th>
<th>Sepal Length</th>
<th>Sepal Width</th>
<th>Petal Length</th>
<th>Petal Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>4.9</td>
<td>3.6</td>
<td>1.4</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>5.3</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>4.9</td>
<td>2.4</td>
<td>3.3</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>5.7</td>
<td>2.8</td>
<td>4.1</td>
<td>1.3</td>
</tr>
<tr>
<td>1</td>
<td>6.3</td>
<td>3.3</td>
<td>4.7</td>
<td>1.6</td>
</tr>
<tr>
<td>1</td>
<td>6.7</td>
<td>3.0</td>
<td>5.0</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set
Fisher Iris Dataset
Classification

Chalkboard:

– Binary classification
– 2D examples
– Decision rules / hypotheses
K-NEAREST NEIGHBORS
k-Nearest Neighbors

**Chalkboard:**

- KNN for binary classification
- Distance functions
- Efficiency of KNN
- Inductive bias of KNN
- KNN Properties