Review Session II
Logic and Reasoning

- John likes any food
- Peanuts can be eaten.
- Anything eaten is food.

Prove: John likes peanuts

\[ \neg \text{likes}(John, \text{peanuts}) \]

Conjecture: \( \text{likes}(John, \text{peanuts}) \)

Negation: \( \{ \neg \text{likes}(John, \text{peanuts}) \} \)

\[
\begin{align*}
\neg \text{likes}(John, \text{peanuts}) & \quad \neg \text{food}(x) \ , \ \text{likes}(John,x) \\
\text{food}(x) \Rightarrow \text{likes}(John,x) & \quad \{ \text{eatable(peanuts)} \} \\
\text{eatable(peanuts)} & \quad \{ \neg \text{eatable}(x) \ , \ \text{food}(x) \} \\
\{ \neg \text{food}(x) \} & \quad \{ \text{eatable}(\text{peanuts}) \} \\
\{ \neg \text{eatable}(x) \ , \ \text{food}(x) \} & \quad \{ \text{food}(\text{peanuts}) \} \\
\{ \neg \text{food}(\text{peanuts}) \} & \quad \{ \text{food}(\text{peanuts}) \} \\
\text{NIL} &
\end{align*}
\]
Decision Trees with Information Gain

<table>
<thead>
<tr>
<th>Gray</th>
<th>Large</th>
<th>LongNeck</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>E</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>G</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>E</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>E</td>
</tr>
</tbody>
</table>

Entropy of root:
\[-\frac{3}{4} \log_2(\frac{3}{4}) - \frac{1}{4} \log_2(\frac{1}{4})
\]
\[= -.75(-.415) -.25(-2)
\]
\[= .811\]

Split on Gray:
Y: \{E,E\} Entropy 0
N: \{G,E\} Entropy 1
Gain: .811 -.5(0)-.5(1)
= .311

Split on Large:
Always Y
\{E,G,E,E\}, same as root.
Gain of Zero.

Split on LongNeck:
Y: \{G\} Entropy 0
N: \{E,E,E\} Entropy 0
Gain: .811 – 0 = .811***
Idea of Boosting
ADABoost

- ADABoost boosts the accuracy of the original learning algorithm.

- If the original learning algorithm does slightly better than 50% accuracy, ADABoost with a large enough number of classifiers is guaranteed to classify the training data perfectly.
ADABOOST Weight Updating
(from Fig 18.34 text)

/* First find the sum of the weights of the misclassified samples */
for j = 1 to N do /* go through training samples */
    if h[m](x_j) <> y_j then error <- error + w_j

/* Now use the ratio of error to 1-error to change the weights of the correctly classified samples */
for j = 1 to N do
    if h[m](x_j) = y_j then w[j] <- w[j] * error/(1-error)
Example

• Start with 4 samples of equal weight .25.
• Suppose 1 is misclassified. So error = .25.
• The ratio comes out .25/.75 = .33
• The correctly classified samples get weight of .25*.33 = .0825

What’s wrong? What should we do?

We want them to add up to 1, not .4975.

Answer: To normalize, divide each one by their sum (.4975).
Neural Nets

\[-2 \times .5 + 2 \times .4 = -.2\]  
\[g(-.2) = -1\]

\[-1 \times .4 + 1 \times .2 = -.2\]  
\[g(-.2) = -1\]

\[-2 \times .5 + 2 \times .6 = .2\]  
\[g(.2) = 1\]
SVMs
K-means: mean $\mu$

EM: mean $\mu$, covariance $\Sigma$, weight $W$
Yi Li’s EM Learning

• Method 1: one Gaussian model per object class

• Method 2: for each class, first use the positive instances to obtain Gaussian clusters in each feature space (color, texture, structure, etc)

• Then use the CLUSTERS to obtain fixed length feature vectors for positive and negative instances of that class and train a neural net
CNNs

• Convolution

• Pooling

• ReLU