Administrivia

- Posters! Tue Apr 24 and Thu Apr 26
- Idea!
- Presentation!
- 2-wide x 4-high landscape pages
Learning so far...
<table>
<thead>
<tr>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Will Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>0-10</td>
<td>$y_1=\text{yes}$</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>30-60</td>
<td>$y_2=\text{no}$</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Some</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0-10</td>
<td>$y_3=\text{yes}$</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Thai</td>
<td>10-30</td>
<td>$y_4=\text{yes}$</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
<td>$y_5=\text{no}$</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0-10</td>
<td>$y_6=\text{yes}$</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>0-10</td>
<td>$y_7=\text{no}$</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0-10</td>
<td>$y_8=\text{yes}$</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>&gt;60</td>
<td>$y_9=\text{no}$</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>10-30</td>
<td>$y_{10}=\text{no}$</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>0-10</td>
<td>$y_{11}=\text{no}$</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>30-60</td>
<td>$y_{12}=\text{yes}$</td>
</tr>
</tbody>
</table>
\[ h_w(x) = w_1x + w_0 \]

\[ L(h_w) = \sum_{j=1}^{N} L_2(y_j, h_w(x_j)) \]

\[ = \sum_{j=1}^{N} (y - h_w(x))^2 \]

\[ = \sum_{j=1}^{N} (y - w_1x + w_0)^2 \]
Carl Friedrich Gauss
(1777–1855)
Linear Regression

Find \( \mathbf{w} = [w_0, w_1] \) that minimizes \( L(h_{\mathbf{w}}) \)

\[
\mathbf{w}^* = \arg\min_{\mathbf{w}} L(h_{\mathbf{w}}) \\
= \arg\min_{\mathbf{w}} \sum_{j=1}^{N} (y - w_1 x + w_0)^2
\]
Gradient Descent

\[ w \leftarrow \text{any point in parameter space} \]

\[
\text{loop until convergence do}
\]

\[
\text{for each } w_i \text{ in } w \text{ do}
\]

\[
w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} L(w)
\]

**Update rule**

**Gradient of loss function along } w_i \text{ axis**

**Learning rate**
Gradient Descent In Weight Space

\[ \mathbf{w} = [w_0, w_1] \]

\[ \mathbf{w}^* = [w_0, w_1] \]
Linear Classifier

\[ w_0 + w_1 x_1 + w_2 x_2 = 0 \]

\[ w \cdot x = 0 \]

All instances of one class are above the line: \( w \cdot x > 0 \)

All instances of one class are below the line: \( w \cdot x < 0 \)

\[ h_w(x) = \text{Threshold}(w \cdot x) \]
Hard Threshold

\[ \text{Threshold}(z) = 1 \text{ if } z \geq 0 \]
\[ = 0 \text{ otherwise} \]
Logistic Threshold

\[ \text{Logistic}(z) = \frac{1}{1 + e^{-z}} \]
"Neuron"

\[ a_j = g(in_j) \]

Where:
- \( a_0 = 1 \)
- \( a_i \)
- \( w_{0,j} \)
- \( w_{i,j} \)
- Input Links
- Bias Weight
- \( in_j \)
- \( g \) is the activation function
- Output Links
- Output
Bags  Agent, process, disease, ...

Candies  Actions, effects, symptoms, results of tests, ...

\( D_1 = \) ▼
\( D_2 = \) ▼
\( D_3 = \) ▼

Observations

Goal  Predict next candy  Predict agent’s next move
      Predict next output of process  Predict disease given symptoms and tests
Bayesian Learning

\[ P(X|d) = \alpha \sum_i P(X|h_i)P(h_i|d) \]

\[ = \alpha \sum_i P(X|h_i)P(d|h_i)P(h_i) \]

Prediction of the hypothesis

Likelihood of the data under the hypothesis

Hypothesis prior
Maximum A Posteriori (MAP)

\[ h_{MAP} = \arg\max_{h_i} P(h_i \mid d) \]

\[ P(X \mid d) \approx P(X \mid h_{MAP}) \]
Maximum Likelihood Hypothesis

- Assume uniform hypothesis prior
- No hypothesis preferred to any other \textit{a priori} (e.g., all equally complex)

\[ h_{MAP} = \arg \max_{h_i} P(h_i \mid d) \]
\[ = \arg \max_{h_i} P(d \mid h_i) = h_{ML} \]
Alarm

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>.95</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>.94</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>.29</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>.001</td>
</tr>
</tbody>
</table>

Burglary

P(B) = .001

Earthquake

P(E) = .002

JohnCalls

<table>
<thead>
<tr>
<th>A</th>
<th>P(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>.90</td>
</tr>
<tr>
<td>f</td>
<td>.05</td>
</tr>
</tbody>
</table>

MaryCalls

<table>
<thead>
<tr>
<th>A</th>
<th>P(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>.70</td>
</tr>
<tr>
<td>f</td>
<td>.01</td>
</tr>
</tbody>
</table>
Maximum Likelihood Hypothesis

$$\arg\max_{\Theta} P(d \mid h_\Theta)$$
Log Likelihood

\[ P(d \mid h_{\Theta}) = \prod_{j} P(d_j \mid h_{\Theta}) \]

\[ = \Theta^c \cdot (1 - \Theta)^l \]

\[ L(d \mid h_{\Theta}) = \log P(d \mid h_{\Theta}) = \sum_{j} \log P(d_j \mid h_{\Theta}) \]

\[ = c \log \Theta + l \log(1 - \Theta) \]
Naive Bayes Models

\{ \text{terrorist, tourist} \}

\begin{center}
\begin{tikzpicture}

\node (class) {Class}
\node (arrival) [below left of=class] {Arrival Mode}
\node (one_way) [below right of=class] {One-way Ticket}
\node (furtive) [below right of=one_way] {Furtive Manner}

\draw [->] (class) -- (arrival);
\draw [->] (class) -- (one_way);
\draw [->] (class) -- (furtive);
\draw [->] (furtive) -- (one_way);
\end{tikzpicture}
\end{center}
Learning Naive Bayes Models

• Naive Bayes model with $n$ Boolean attributes requires $2^n + 1$ parameters

• Maximum likelihood hypothesis $h_{ML}$ can be found with no search

• Scales to large problems

• Robust to noisy or missing data
Smoking
Diet
Exercise
Symptom
1
Symptom
2
Symptom
3

(a) (b)

HeartDisease

Symptom
1
Symptom
2
Symptom
3

222
54
666

78 parameters

708 parameters
Hidden (Latent) Variables

- Can dramatically reduce the number of parameters required to specify a Bayes net
- Reduces amount of data required to learn the parameters
- Values of hidden variables not present in training data (observations)
- “Complicates” the learning problem
EM: Expectation Maximization

- Repeat

- E: Use the current values of the parameters to compute the expected values of the hidden variables

- M: Recompute the parameters to maximize the log-likelihood of the data given the values of the variables (observed and hidden)
Reinforcement Learning
Project Pigeon

Project Pigeon was a classified research-and-development program during World War II. It was developed at a time when electronic guidance systems did not exist, and the only compensation for the inaccuracy of bombs was dropping them in quantity. This ingenuous application of shaping would have dramatically increased the accuracy of bombs and decreased civilian casualties. Despite favorable performance tests, however, the National Defense Research Committee ended the project—it seems they couldn't get over the idea that pigeons would be guiding their bombs.

1. Pigeons were trained to peck at targets on aerial photographs. Once a certain level of proficiency was obtained, pigeons were jacketed and mounted inside tubes.

2. The pigeons in their tubes were inserted into the nosecone of the bomb. Each nosecone used three pigeons in a type of voting system, whereby the pigeon pecks of two birds in agreement would overrule the errant pigeon pecks of a single bird.

3. Sealed in the bomb, the pigeons could see through glass lenses in the nosecone.

4. Once the bomb was released, the pigeons would begin pecking at their view of the target. Their pecks shifted the glass lens off-center, which adjusted the bomb's tail surfaces and, correspondingly, its trajectory.
Reinforcement Learning
The Problem with Learning from Examples
Where do the examples come from?
### Input Attributes

<table>
<thead>
<tr>
<th>Input Attributes</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Qty</td>
<td>Bar</td>
<td>Fri</td>
<td>Hun</td>
<td>Pat</td>
<td>Price</td>
<td>Rain</td>
<td>Res</td>
<td>Type</td>
<td>Est</td>
<td>Age</td>
<td></td>
</tr>
<tr>
<td>(x_1)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
<td>(y_1=\text{yes})</td>
</tr>
<tr>
<td>(x_2)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0-10</td>
<td>(y_2=\text{no})</td>
</tr>
<tr>
<td>(x_3)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Some</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0-10</td>
<td>(y_3=\text{yes})</td>
</tr>
<tr>
<td>(x_4)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Thai</td>
<td>10-30</td>
<td>(y_4=\text{yes})</td>
</tr>
<tr>
<td>(x_5)</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
<td>(y_5=\text{no})</td>
</tr>
<tr>
<td>(x_6)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0-10</td>
<td>(y_6=\text{yes})</td>
</tr>
<tr>
<td>(x_7)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0-10</td>
<td>(y_7=\text{no})</td>
</tr>
<tr>
<td>(x_8)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0-10</td>
<td>(y_8=\text{yes})</td>
</tr>
<tr>
<td>(x_9)</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>&gt;60</td>
<td>(y_9=\text{no})</td>
</tr>
<tr>
<td>(x_{10})</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>0-30</td>
<td>(y_{10}=\text{no})</td>
</tr>
<tr>
<td>(x_{11})</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>0-10</td>
<td>(y_{11}=\text{no})</td>
</tr>
<tr>
<td>(x_{12})</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>30-60</td>
<td>(y_{12}=\text{no})</td>
</tr>
</tbody>
</table>
But we need feedback!
Reward
(a.k.a Reinforcement)

• The positive or negative feedback one obtains from in response to action

• In animals:
  • Pain, hunger: negative reward
  • Pleasure, food: positive reward

• In computers...?
Markov Decision Process

• Sequential decision problem
• Fully observable, stochastic environment
• Set of states $S$ with initial state $s_0$
• Markovian transition model: $P(s' \mid s, a)$
• Additive rewards: $R(s)$
Policy

- A policy $\pi$ specifies what the agent should do for any state the agent might reach: $\pi(s)$
- Each time a policy is executed, it leads to a different history
- Quality of a policy is its expected utility
- Optimal policy $\pi^*$ maximizes expected utility
Optimal Policy

$$U_{\pi}(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

$$\pi^*_s = \arg\max_{\pi} U^{\pi}(s)$$
Computing Policies

- Value Iteration
  - Easy to understand (AIMA 17.2.2)
  - Converges to unique set of solutions to Bellman equations

- Policy Iteration
  - Searches space of policies, rather than refining values of utilities
  - More tractable
Markov Decision Process

• Sequential decision problem
• Fully observable, stochastic environment
• Set of states $S$ with initial state $s_0$
• Markovian transition model: $P(s' | s, a)$
• Additive rewards: $R(s)$

Learn!
Reinforcement Learning

- Learn a policy that tells you what to do without knowing
  - How actions work
  - How the environment behaves
  - How you get rewarded
Passive Learning

• Fixed policy $\pi : \pi(s)$ says what to do

• Learn $U_\pi(s)$: how good this policy is
\[ R(s) = -0.04, \quad \gamma = 1 \]

\[ U^\pi(s) ? \]
Policy Iteration

• Repeat

• Policy Evaluation: Given a policy, compute its expected utility

• Policy Improvement: Compute a new MEU policy by checking for better action in any state, given EU
Policy Evaluation

\[ U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s') \]
(1,1) → (1,2) → (1,3) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)  
-0.04 -0.04 -0.04 -0.04 -0.04 -0.04 -0.04 +1

(1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (3,2) → (3,3) → (4,3)  
-0.04 -0.04 -0.04 -0.04 -0.04 -0.04 -0.04 +1

(1,1) → (2,1) → (3,1) → (4,2)  
-0.04 -0.04 -0.04 -1
Direct Utility Estimation

• In each trial, compute reward-to-go for each state visited in the trial

• Keep track of average reward-to-go for every state

• In the limit, converges to true expected utility of the policy $U^\pi(s)$
Utilities of states are not independent!

\[ U^\pi(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^\pi(s') \]
Adaptive Dynamic Programming

- Keep track of observed frequencies of state-action pairs and their outcomes
- Approximate unknown transition model $P(s' | s, a)$ using observed frequencies
- Use that in standard policy evaluation to compute utility of policy
$U^{\pi}(1,3) = 0.84$

$U^{\pi}(2,3) = 0.92$

(1,1) → (1,2) → (1,3) → (1,2) → (1,3) → (2,3) → (3,3) → (4,3)

-0.04 -0.04 -0.04 -0.04 -0.04 -0.04 -0.04 +1

(1,1) → (1,2) → (1,3) → (2,3) → (3,3) → (3,2) → (3,3) → (4,3)

-0.04 -0.04 -0.04 -0.04 -0.04 -0.04 -0.04 +1

(1,1) → (2,1) → (3,1) → (4,2)

-0.04 -0.04 -0.04 -1
Temporal-Difference (TD) Learning

- At each step, update utility estimates using difference between successive states:

\[
U^\pi(s) \leftarrow U^\pi(s) + \alpha (R(s) + \gamma U^\pi(s') - U^\pi(s))
\]

Learning Rate
Where did that policy come from???
Possible Strategy

- Learn outcome model for actions based on observed frequencies (like ADP)
- Compute utility of optimal policy using value or policy iteration (at each observation)
- Use computed optimal policy to select next action
How could following the optimal policy not result in optimal behavior?

The learned model is just an approximation of the true environment. What is optimal in the learned model may not really be optimal in the environment.
Paradox?

• Need to explore unexplored states (since they may be better than where we've been)

• But they may be worse than our current optimal

• And after a while they probably will be
Active Learning

• Need to tradeoff
  • **Exploitation**: Maximizing immediate reward by following current utility estimates
  • **Exploration**: Improving utility estimates to maximize long-term reward
GLIE

- “Greedy in the limit of infinite exploration”
- Eventually, follow the optimal policy
- Examples:
  - Choose a random action $\frac{1}{t}$ of the time
  - Give some weight to actions you haven’t tried very often, while avoiding actions with strong estimates of low utility
Exploration Function

\[ U^+(s) \leftarrow R(s) + \gamma \max_a f(\sum_{s'} P(s' | s, a)U^+(s'), N(s, a)) \]

\[ f(u, n) = \begin{cases} R^+, & \text{if } n < N_e \\ u, & \text{otherwise} \end{cases} \]
Reinforcement Learning

- Doesn’t require labelled examples (training data)
- Learn a policy that tells you what to do without knowing
  - How actions work
  - How the environment behaves
  - How you get rewarded
For Next Time:
Posters!
(Don’t Be Late)