To play or not to play?
Sometimes $|X|$ is too big for exact inference
- $|X|$ may be too big to even store $P(X_t | e_{1:t})$
  E.g. when $X$ is continuous

Solution: Approximate inference
- Track a set of samples of $X$
- Samples are called particles

Number of samples for $X=x$ is proportional to probability of $x$
Particle Filtering
Step 1: Elapse Time

- Each particle $x$ is moved by sampling its next position using the transition model:
  $$x' = \text{sample}(P(X'|x))$$
  - Samples’ frequencies reflect the transition probabilities
  - In example, most samples move clockwise, but some move in another direction or stay in place

- This step captures passage of time
Particle Filtering
Step 2: Observe

Weight particles according to evidence

- **Assign weights** $w$ to samples based on the new observed evidence $e$

$$w(x) = P(e|x)$$

- In example, true ghost position is shown in red outline; samples closer to ghost get higher weight (bigger size of circles) based on noisy distance emission model
Particle Filtering
Step 3: Resample

- N times, we choose from our weighted sample distribution (i.e. randomly select with replacement)
  - Each sample selected with probability proportional to its weight

- Now the update is complete for this time step, continue with the next one

Old Particles:
- (1,3) $w=0.1$
- (3,2) $w=0.9$
- (3,2) $w=0.9$
- (3,1) $w=0.4$
- (2,3) $w=0.3$
- (2,2) $w=0.4$
- (3,3) $w=0.4$
- (3,3) $w=0.4$
- (3,2) $w=0.9$
- (2,3) $w=0.3$

New Particles:
- (3,2) $w=1$
- (3,2) $w=1$
- (3,2) $w=1$
- (2,3) $w=1$
- (2,2) $w=1$
- (3,2) $w=1$
- (3,1) $w=1$
- (3,3) $w=1$
- (3,2) $w=1$
- (3,1) $w=1$
Particle Filtering Summary

- Represent current belief $P(X \mid \text{evidence to date})$ as a set of $N$ samples (actual values $x$).
- For each new observation $e$:
  1. **Sample transition**, once for each current particle $x$:
     \[
     x' = \text{sample}(P(X' \mid x))
     \]
  2. For each new sample $x'$, **compute importance weights** for the new evidence $e$:
     \[
     w(x') = P(e \mid x')
     \]
  3. Finally, **resample** the importance weights to create $N$ new particles.
Example 1

Particle filter, uniform initial beliefs, 25 particles
Example 2

Particle filter, uniform initial beliefs, 300 particles
Big Data is data that is too large, complex and dynamic for any conventional data tools to capture, store, manage and analyze. The right use of Big Data allows analysts to spot trends and gives niche insights that help create value and innovation much faster than conventional methods.

The “three V’s”, i.e. the Volume, Variety and Velocity of the data coming in is what creates the challenge.

**Volume**
- >3,500 North America
- >2,000 Europe
- >250 China
- >400 Japan
- >200 Middle East
- >50 Latin America
- >50 India

Amount of Big Data stored across the world (in petabytes)

**Variety**
- **People to People**
  - Netizens, Virtual Communities, Social Networks, Web Logs...
- **People to Machine**
  - Archives, Medical Devices, Digital TV, E-Commerce, Smart Cards, Bank Cards, Computers, Mobiles...
- **Machine to Machine**
  - Sensors, GPS Devices, Bar Code Scanners, Surveillance Cameras, Scientific Research...

**Velocity**
- **Emails**
  - 2.9 million emails sent every second
- **YouTube**
  - 20 hours of video uploaded every minute
- **Twitter**
  - 50 million tweets per day

Enter... Machine Learning
Varieties of Machine Learning

- **Supervised learning**: correct answers for each input is provided, goal is to *generalize* to new data
  - E.g., decision trees, neural networks

- **Unsupervised learning**: correct answers not given, must *discover patterns* in input data
  - E.g., clustering, principal component analysis

- **Reinforcement learning**: occasional *rewards* (or punishments) given to guide behavior
  - We’ve covered this already! (Q-learning, MDPs)
Supervised learning

- Goal: Construct a function $h$ from training data to approximate the hidden function $f$ that is generating the data
  - $h$ is consistent if it agrees with $f$ on all training examples

Given: Data points $(x, f(x))$ (training examples)

What kind of function would you fit?

Curve fitting (aka regression)
Supervised learning example

$h = \text{Straight line?}$
Supervised learning example

What about a quadratic function?

What about this little fella?
Supervised learning example

Finally, a function that satisfies all! (consistent function)
Supervised learning example

But so does this one…
Ockham’s Razor Principle

Prefer the simplest hypothesis consistent with data

- Related to KISS principle ("keep it simple stupid")
- *Smooth* blue function preferable over wiggly yellow one
- If noise known to exist in data, even linear might be better (the lowest x might be due to noise)
Types of Supervised Learning

- **Classification**: Output is discrete (e.g., Yes/No, Class 1 or Class 2 or Class 3, etc.)
  - Decision trees
  - K-nearest neighbor
  - Linear Classifiers
  - Support Vector Machines (SVMs)
  - Cross validation

- **Regression**: Output is continuous
  - Linear regression and Neural networks
    - Backpropagation learning algorithm
**Goal:** Learn the function “PlayTennis?” from example data

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Humid</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>s</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d2</td>
<td>s</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d4</td>
<td>r</td>
<td>h</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d5</td>
<td>r</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d6</td>
<td>r</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d7</td>
<td>o</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d8</td>
<td>s</td>
<td>h</td>
<td>w</td>
<td>n</td>
</tr>
<tr>
<td>d9</td>
<td>s</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d10</td>
<td>r</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d11</td>
<td>s</td>
<td>n</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d12</td>
<td>o</td>
<td>h</td>
<td>s</td>
<td>y</td>
</tr>
<tr>
<td>d13</td>
<td>o</td>
<td>n</td>
<td>w</td>
<td>y</td>
</tr>
<tr>
<td>d14</td>
<td>r</td>
<td>h</td>
<td>s</td>
<td>n</td>
</tr>
</tbody>
</table>

- **Outlook** = sunny (s), overcast (o), or rain (r)
- **Humidity** = high (h), or normal (n)
- **Wind** = weak (w) or strong (s)
A Decision Tree for the Same Data

Decision Tree for “PlayTennis?”

Leaves = classification output
Arcs = choice of value for parent attribute

Decision tree equivalent to logical statement in disjunctive normal form
PlayTennis ⇔ (Sunny ∧ Normal) ∨ Overcast ∨ (Rain ∧ Weak)
Decision Trees

- **Input:** Set of attributes describing an object or situation

- **Output:** Predicted output value for the input

- Decision tree is *consistent* if it produces the correct output on all training examples

- Input and output can be *discrete or continuous*
Example: Decision Tree for Continuous Values

Input: Continuous-valued attributes \((x_1, x_2)\)
Output: 0 or 1

How do we branch on attribute values \(x_1\) and \(x_2\) to partition the space and generate correct outputs?
Example: Classification of Continuous Valued Inputs

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
Expressiveness of Decision Trees

- Decision trees can express any function of the input attributes.
- E.g., Boolean functions, truth table row = path to leaf:

- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example
  - But most likely won't generalize to new examples
- Prefer to find more compact decision trees
Learning Decision Trees

- Example: When should I wait for a table at a restaurant?
Learning Decision Trees

- **Example:** When should I wait for a table at a restaurant?

- Attributes (features) relevant to *Wait?* decision:
  1. **Alternate**: is there an alternative restaurant nearby?
  2. **Bar**: is there a comfortable bar area to wait in?
  3. **Fri/Sat**: is today Friday or Saturday?
  4. **Hungry**: are we hungry?
  5. **Patrons**: number of people in the restaurant (None, Some, Full)
  6. **Price**: price range ($, $$, $$$)
  7. **Raining**: is it raining outside?
  8. **Reservation**: have we made a reservation?
  9. **Type**: kind of restaurant (French, Italian, Thai, Burger)
  10. **WaitEstimate**: estimated waiting time (0-10, 10-30, 30-60, >60)
A “personal” decision tree

- A decision tree for *Wait?* based on personal “rules of thumb”:
Input Data for Learning

- Past examples when I did/did not wait for a table:

<table>
<thead>
<tr>
<th>Example</th>
<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>Target</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_5$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_6$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_8$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Next Time

- Learning Decision Trees from Data
- Preventing Overfitting and Generalization
  - Cross-Validation
- To Do:
  - Project 4
  - Read Chapter 18