Recap: Reasoning Over Time

- **Markov models**
  
  \[ P(X_1) \quad P(X|X_{-1}) \]

- **Hidden Markov models**

  \[
  \begin{array}{c}
  X_1 \quad X_2 \quad X_3 \quad X_4 \quad \ldots \\
  E_1 \quad E_2 \quad E_3 \quad E_4
  \end{array}
  \]

### Table: Event Probabilities

<table>
<thead>
<tr>
<th>X</th>
<th>E</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>rain</td>
<td>umbrella</td>
<td>0.9</td>
</tr>
<tr>
<td>rain</td>
<td>no umbrella</td>
<td>0.1</td>
</tr>
<tr>
<td>sun</td>
<td>umbrella</td>
<td>0.2</td>
</tr>
<tr>
<td>sun</td>
<td>no umbrella</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Recap: Filtering

Elapse time: compute $P(X_t | e_{1:t-1})$

$$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$

Observe: compute $P(X_t | e_{1:t})$

$$P(x_t|e_{1:t}) \propto P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$

Belief: $<P(rain), P(sun)>$

- $P(X_1) <0.5, 0.5>$ Prior on $X_1$
- $P(X_1 | E_1 = umbrella) <0.82, 0.18>$ Observe
- $P(X_2 | E_1 = umbrella) <0.63, 0.37>$ Elapse time
- $P(X_2 | E_1 = umb, E_2 = umb) <0.88, 0.12>$ Observe

[Demo: Ghostbusters Exact Filtering]
Video of Ghostbusters Exact Filtering (Reminder)
Particle Filtering

Diagram showing a map with various paths and labels indicating directions and locations.
Particle Filtering

- Filtering: approximate solution
- Sometimes $|X|$ is too big to use exact inference
  - $|X|$ may be too big to even store $B(X)$
  - E.g. $X$ is continuous
- Solution: approximate inference
  - Track samples of $X$, not all values
  - Samples are called particles
  - Time per step is linear in the number of samples
  - But: number needed may be large
  - In memory: list of particles, not states
- This is how robot localization works in practice
- Particle is just new name for sample
Our representation of $P(X)$ is now a list of $N$ particles (samples)
- Generally, $N \ll |X|$
- Storing map from $X$ to counts would defeat the point

$P(x)$ approximated by number of particles with value $x$
- So, many $x$ may have $P(x) = 0!$
- More particles, more accuracy

For now, all particles have a weight of 1
Each particle is moved by sampling its next position from the transition model

\[ x' = \text{sample}(P(X'|x)) \]

- This is like prior sampling - samples’ frequencies reflect the transition probabilities
- Here, most samples move clockwise, but some move in another direction or stay in place
- This captures the passage of time
  - If enough samples, close to exact values before and after (consistent)
- Slightly trickier:
  - Don’t sample observation, fix it
  - Similar to likelihood weighting, downweight samples based on the evidence

\[ w(x) = P(e|x) \]

\[ B(X) \propto P(e|X)B'(X) \]

- As before, the probabilities don’t sum to one, since all have been downweighted (in fact they now sum to \((N \text{ times})\) an approximation of \(P(e)\))
Particle Filtering: Resample

- Rather than tracking weighted samples, we resample.
- N times, we choose from our weighted sample distribution (i.e. draw with replacement).
- This is equivalent to renormalizing the distribution.
- Now the update is complete for this time step, continue with the next one.
Recap: Particle Filtering

- **Particles**: track samples of states rather than an explicit distribution

```
Particles: (3,3) (2,3) (3,3) (3,2) (3,3) (3,2) (1,2) (3,3) (3,3) (2,3) (2,3)
Elapse
```

```
Particles: (3,3) (2,3) (3,2) (3,1) (3,3) (3,2) (1,3) (3,3) (3,3) (3,2) (2,2)
Weight
```

```
Particles: (3,3) w=.4 (2,3) w=.2 (3,2) w=.9 (3,1) w=.4 (3,3) w=.4 (3,2) w=.9 (1,3) w=.1 (2,3) w=.2 (3,2) w=.9 (2,2) w=.4
Resample
```

```
(New) Particles: (3,2) (2,3) (3,2) (1,3) (3,3) (3,2) (3,2) (3,2) (3,2) (3,2)
[Demos: ghostbusters particle filtering]
```
Video of Demo – Moderate Number of Particles
Video of Demo - One Particle
Video of Demo - Huge Number of Particles
Robot Localization

- In robot localization:
  - We know the map, but not the robot’s position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store \( B(X) \)
  - Particle filtering is a main technique
Particle Filter Localization (Sonar)

Global localization with sonar sensors

[Video: global-sonar-uw-annotated.avi]
Robot Mapping

- **SLAM: Simultaneous Localization And Mapping**
  - We do not know the map or our location
  - State consists of position AND map!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

[Demo: PARTICLES-SLAM-mapping1-new.avi]
Particle Filter SLAM - Video 1

[Demo: PARTICLES-SLAM-mapping1-new.avi]
Dynamic Bayes Nets
Dynamic Bayes Nets (DBNs)

- We want to track multiple variables over time, using multiple sources of evidence.
- Idea: Repeat a fixed Bayes net structure at each time.
- Variables from time $t$ can condition on those from $t-1$.

- Dynamic Bayes nets are a generalization of HMMs.

[Demo: pacman sonar ghost DBN model (L15D6)]
A particle is a complete sample for a time step

**Initialize:** Generate prior samples for the $t=1$ Bayes net
- Example particle: $G_1^a = (3,3)$ $G_1^b = (5,3)$

**Elapse time:** Sample a successor for each particle
- Example successor: $G_2^a = (2,3)$ $G_2^b = (6,3)$

**Observe:** Weight each *entire* sample by the likelihood of the evidence conditioned on the sample
- Likelihood: $P(E_1^a | G_1^a) \times P(E_1^b | G_1^b)$

**Resample:** Select prior samples (tuples of values) in proportion to their likelihood
Most Likely Explanation
HMMs: MLE Queries

- HMMs defined by
  - States $X$
  - Observations $E$
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X|X_{-1})$
  - Emissions: $P(E|X)$

- New query: most likely explanation:

- New method: the Viterbi algorithm

$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$
State Trellis

- State trellis: graph of states and transitions over time

- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t|x_{t-1})P(e_t|x_t)$
- Each path is a sequence of states
- The product of weights on a path is that sequence’s probability along with the evidence
- Forward algorithm computes sums of paths, Viterbi computes best paths
Forward / Viterbi Algorithms

Forward Algorithm (Sum)

\[ f_t[x_t] = P(x_t, e_{1:t}) \]

\[ = P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) f_{t-1}[x_{t-1}] \]

Viterbi Algorithm (Max)

\[ m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t}) \]

\[ = P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}] \]
Speech Recognition
Digitizing Speech
Speech in an Hour

- Speech input is an acoustic waveform

Figure: Simon Arnfield, http://www.psyc.leeds.ac.uk/research/cogn/speech/tutorial/
Spectral Analysis

- Frequency gives pitch; amplitude gives volume
  - Sampling at ~8 kHz (phone), ~16 kHz (mic) (kHz=1000 cycles/sec)
- Fourier transform of wave displayed as a spectrogram
  - Darkness indicates energy at each frequency
- Time slices are translated into acoustic feature vectors (~39 real numbers per slice)

- These are the observations $E$, now we need the hidden states $X$
Speech State Space

- **HMM Specification**
  - $P(E|X)$ encodes which acoustic vectors are appropriate for each phoneme (each kind of sound)
  - $P(X|X')$ encodes how sounds can be strung together

- **State Space**
  - We will have one state for each sound in each word
  - Mostly, states advance sound by sound
  - Build a little state graph for each word and chain them together to form the state space $X$
States in a Word

Word Model

Observation Sequence
(spectral feature vectors)

\[
\begin{align*}
\text{start}_0 & \quad a_{01} \quad n_1 \\
& \quad b_1(o_1) \quad a_{12} \quad b_1(o_2) \quad iy_2 \quad a_{22} \quad b_2(o_3) \quad b_2(o_5) \\
& \quad a_{23} \quad d_3 \quad a_{33} \quad a_{34} \quad \text{end}_4 \\
& \quad b_3(o_6) \\
& \quad o_1 \quad o_2 \quad o_3 \quad o_4 \quad o_5 \quad o_6 \\
& \quad \ldots \quad \ldots
\end{align*}
\]
Transitions with a Bigram Model

Figure: Huang et al, p. 618

Training Counts

\[
\hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162} = 0.0006
\]
Decoding

- Finding the words given the acoustics is an HMM inference problem.
- Which state sequence $x_{1:T}$ is most likely given the evidence $e_{1:T}$?

\[ x_{1:T}^* = \arg \max_{x_{1:T}} P(x_{1:T} | e_{1:T}) = \arg \max_{x_{1:T}} P(x_{1:T}, e_{1:T}) \]

- From the sequence $x$, we can simply read off the words.
Next Time: Bayes’ Nets!