Recap: Reasoning Over Time

- Markov models

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

- Hidden Markov models

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

\[
P(E|X)
\]

<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 $P(X_t \mid e_{1:t-1})$

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

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

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

Belief: $\langle P(\text{rain}), P(\text{sun}) \rangle$

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

Particle Filtering

- Sometimes $|X|$ is too big to use exact inference
  - $|X|$ may be too big to even store $B(X)$
  - E.g. $X$ is continuous
  - $|X|^2$ may be too big to do updates

- 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
Representation: Particles

- 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 $x$ will have $P(x) = 0!$
  - More particles, more accuracy

- For now, all particles have a weight of 1

---

Particle Filtering: Elapse Time

- 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 probs
- Here, most samples move clockwise, but some move in another direction or stay in place

- This captures the passage of time
  - If we have enough samples, close to the exact values before and after (consistent)
Particle Filtering: Observe

- Slightly trickier:
  - Don’t do rejection sampling (why not?)
  - We don’t sample the observation, we fix it
  - This is similar to likelihood weighting, so we downweight our samples based on the evidence
    \[ w(x) = P(e|x) \]
    \[ B(X) \propto P(e|X)B'(X) \]
  - Note that, as before, the probabilities don’t sum to one, since most have been downweighted (in fact they sum to 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 analogous to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

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

New Particles:
(2,1) w=1
(2,1) w=1
(2,1) w=1
(3,2) w=1
(2,2) w=1
(2,1) w=1
(1,1) w=1
(3,1) w=1
(2,1) w=1
(1,1) w=1
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 $\mathbb{B}(X)$
  - Particle filtering is a main technique

[Demos]

P4: Ghostbusters 2.0 (beta)

- **Plot:** Pacman's grandfather, Grandpac, learned to hunt ghosts for sport.
- He was blinded by his power, but could hear the ghosts' banging and clanging.
- **Transition Model:** All ghosts move randomly, but are sometimes biased
- **Emission Model:** Pacman knows a “noisy” distance to each ghost

[Demo]
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 \( t-1 \)

```
G_1^a \quad G_1^b
\downarrow \quad \downarrow
E_1^a \quad E_1^b
```

```
G_2^a \quad G_2^b
\downarrow \quad \downarrow
E_2^a \quad E_2^b
```

```
G_3^a \quad G_3^b
\downarrow \quad \downarrow
E_3^a \quad E_3^b
```

- DBNs with evidence at leaves are (in principle) HMMs

---

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Procedure: “unroll” the network for \( T \) time steps, then eliminate variables until \( P(X_T|e_{1:T}) \) is computed

```
G_1^a \quad G_1^b
\downarrow \quad \downarrow
E_1^a \quad E_1^b
```

```
G_2^a \quad G_2^b
\downarrow \quad \downarrow
E_2^a \quad E_2^b
```

```
G_3^a \quad G_3^b \quad G_3^b
\downarrow \quad \downarrow \quad \downarrow
E_3^a \quad E_3^b
```

- Online belief updates: Eliminate all variables from the previous time step; store factors for current time only
DBN Particle Filters

- 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: \( P(E_1^a | G_1^a) \) \( P(E_1^b | G_1^b) \)
- **Resample**: Select prior samples (tuples of values) in proportion to their likelihood

---

SLAM

- **SLAM** = Simultaneous Localization And Mapping
  - We do not know the map or our location
  - Our belief state is over maps and positions!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods

---

[Demo] [DEMOS]
HMMs: MLE Queries

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

- Query: most likely explanation:

\[ \arg \max_{X_{1:t}} P(x_{1:t}|e_{1:t}) \]

State Path 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 the seq’s probability
- Can think of the Forward (and now Viterbi) algorithms as computing sums of all paths (best paths) in this graph
### Viterbi Algorithm

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

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

$= \max_{x_{1:t-1}} P(x_{1:t-1}, e_{1:t-1}) P(x_t | x_{t-1}) P(e_t | x_t)$

$= P(e_t | x_t) \max_{x_{t-1}} P(x_t | x_{t-1}) m_{t-1}[x_{t-1}]$

### Example

<table>
<thead>
<tr>
<th>$Rain_1$</th>
<th>$Rain_2$</th>
<th>$Rain_3$</th>
<th>$Rain_4$</th>
<th>$Rain_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8182</td>
<td>0.1818</td>
<td>0.5155</td>
<td>0.0361</td>
<td>0.0334</td>
</tr>
<tr>
<td>0.0491</td>
<td>0.1237</td>
<td>0.0173</td>
<td>0.0120</td>
<td>0.0024</td>
</tr>
</tbody>
</table>

$m_{1:1}$ to $m_{1:5}$ represent the probabilities along each path.