Today

- **HMMs**
  - Particle filters
  - Demos!
  - Most-likely-explanation queries

- **Applications**:
  - Robot localization / mapping
  - Speech recognition (later)

Recap: Reasoning Over Time

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

- Hidden Markov models
  \[
P(E|X)
\]

Inference: Base Cases

- Assume we have current belief \( P(X | \text{evidence to date}) \)
  \[
  B(X_t) = P(X_t|e_{t-1})
  \]

- Then, after one time step passes:
  \[
  P(X_{t+1}|e_{t+1}) = \sum_{x_t} P(X_{t+1}|x_t, e_{t+1}) P(x_t|e_t)
  \]
  Or compactly:
  \[
  B'(X_{t+1}) = \sum_{x_t} P(X'|x_t) B(x_t)
  \]

- Basic idea: beliefs get “pushed” through the transitions
  - With the “B” notation, we have to be careful about what time step the belief is about, and what evidence it includes
Example: Passage of Time

- As time passes, uncertainty "accumulates" (Transition model: ghosts usually go clockwise)

```
T = 1
T = 2
T = 5
```

Inference: Base Cases

```
P(X_t | e_t)
P(e_t | e_{t-1}) = P(e_t | e_{t-1}) / P(e_t)
\propto_X P(X_t | e_{t-1}) P(X_t | e_{t})
= P(e_t | X_t) P(X_t | e_{t})
```

Example: Observation

- As we get observations, beliefs get reweighted, uncertainty "decreases"

```
Before observation
After observation
```

Filtering

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

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

```
Observe: compute P(X_t | e_t)
```

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

Particle Filtering

```
Belief: \{P(rain), P(sun)\}
```

```
P(X_t)
<0.5, 0.5> Prior on X_t

P(X_t | E_1 = umbrella) <0.82, 0.18>
Observe

P(X_t | E_1 = umbrella) <0.85, 0.17>
Elapse time

P(X_t | E_2 = umb, E_2 = sun) <0.88, 0.12>
Observe

[Demo: Ghostbusters Exact Filtering (1150)]
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
  - In memory: list of particles, not states
- This is how robot localization works in practice
- Particle is just new name for sample

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 $x$ may 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
  $$ x' \sim \text{sample}(P(X|x)) $$
- 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)

Particle Filtering: Observe

- Slightly trickier:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, downweight samples based on the evidence
    $$ w(x) = P(o|x) $$
    $$ B(X) \propto P(o|X)B'(X) $$
  - As before, the probabilities don't sum to one, since all have been downweighted (in fact they now sum to $(N \times)$ an approximation of $P(o)$)

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
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

Particle Filter Localization (Laser)

- Particle Filter Localization (Laser)

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

Particle Filter SLAM – Video 1

- Particle Filter SLAM – Video 1

Particle Filter SLAM – Video 2

- Particle Filter SLAM – Video 2
Dynamic Bayes Nets

- 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

Pacman – Sonar (P4)

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

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: \( P(E_t^a | G_t^a) \cdot P(E_t^b | G_t^b) \)
- 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: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$

- New method: the Viterbi algorithm

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)
  - $f_1(x_1) = P(x_1, e_{1:1})$
  - $f_t(x_t) = P(e_t|x_t)\sum_{x_{t-1}} f_{t-1}(x_{t-1})P(x_t|x_{t-1})$

- Viterbi Algorithm (Max)
  - $m_t(x_t) = \max_{x_{t-1}} P(x_{1:t-1}, x_t, e_{1:t})$
  - $m_t(x_t) = P(e_t|x_t)\max_{x_{t-1}} P(x_{1:t-1})m_{t-1}(x_{t-1})$