Reasoning over Time or Space

- Often, we want to **reason about a sequence** of observations
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring

- Need to introduce time (or space) into our models
Example Markov Chain: Weather

- States: $X = \{\text{rain, sun}\}$

  - Initial distribution:

<table>
<thead>
<tr>
<th>$X_0$</th>
<th>$P(X_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>1</td>
</tr>
<tr>
<td>rain</td>
<td>0.0</td>
</tr>
</tbody>
</table>

  - CPT $P(X_t \mid X_{t-1})$:

<table>
<thead>
<tr>
<th>$X_{t-1}$</th>
<th>$X_t$</th>
<th>$P(X_t \mid X_{t-1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>sun</td>
<td>0.9</td>
</tr>
<tr>
<td>sun</td>
<td>rain</td>
<td>0.1</td>
</tr>
<tr>
<td>rain</td>
<td>sun</td>
<td>0.3</td>
</tr>
<tr>
<td>rain</td>
<td>rain</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Two new ways of representing the same CPT.
Markov Chains

- Value of $X$ at a given time is called the state

- Transition probabilities (dynamics): $P(X_t | X_{t-1})$ specify how the state evolves over time

$P(X_1)$ \hspace{2cm} $P(X_t | X_{t-1})$

$P(X_t) =$?
Markov Assumption

- Basic conditional independence:
  - Given the present, the future is independent of the past!
  - Each time step only depends on the previous
  - This is called the (first order) Markov property
Example Markov Chain: Weather

- Initial distribution: 1.0 sun

- What is the probability distribution after one step?

\[ P(X_2 = \text{sun}) = \sum_{x_1} P(x_1, X_2 = \text{sun}) = \sum_{x_1} P(X_2 = \text{sun}|x_1)P(x_1) \]

\[ P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain}) \]

\[ 0.9 \cdot 1.0 + 0.3 \cdot 0.0 = 0.9 \]
Mini-Forward Algorithm

- Question: What’s $P(X)$ on some day $t$?

$$P(x_1) = \text{known}$$

$$P(x_t) = \sum_{x_{t-1}} P(x_{t-1}, x_t)$$

$$= \sum_{x_{t-1}} P(x_t \mid x_{t-1}) P(x_{t-1})$$

 Forward simulation
### Example Run of Mini-Forward Algorithm

- **From initial observation of sun**

  \[
  \begin{align*}
  \mathbf{P}(X_1) &= \langle 1.0 \rangle, \\
  \mathbf{P}(X_2) &= \langle 0.9 \rangle, \\
  \mathbf{P}(X_3) &= \langle 0.84 \rangle, \\
  \mathbf{P}(X_4) &= \langle 0.804 \rangle, \\
  \mathbf{P}(X_\infty) &= \langle 0.75 \rangle, \\
  \end{align*}
  \]

- **From initial observation of rain**

  \[
  \begin{align*}
  \mathbf{P}(X_1) &= \langle 0.0 \rangle, \\
  \mathbf{P}(X_2) &= \langle 0.3 \rangle, \\
  \mathbf{P}(X_3) &= \langle 0.48 \rangle, \\
  \mathbf{P}(X_4) &= \langle 0.588 \rangle, \\
  \mathbf{P}(X_\infty) &= \langle 0.75 \rangle, \\
  \end{align*}
  \]

- **From yet another initial distribution \( \mathbf{P}(X_1) \):**

  \[
  \begin{align*}
  \mathbf{P}(X_1) &= \langle \frac{p}{1 - p} \rangle, \\
  \mathbf{P}(X_\infty) &= \langle 0.75 \rangle, \\
  \end{align*}
  \]
Stationary Distribution

- **For most chains:**
  - Influence of the initial distribution gets less and less over time.
  - The distribution we end up in is independent of the initial distribution

- **Stationary distribution:**
  - The distribution we end up with is called the stationary distribution $P_\infty$
  - It satisfies
    $$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$$
Example: Stationary Distribution

○ Question: What’s $P(X)$ at time $t = \infty$?

$$P_\infty(\text{sun}) = P(\text{sun} | \text{sun})P_\infty(\text{sun}) + P(\text{sun} | \text{rain})P_\infty(\text{rain})$$

$$P_\infty(\text{rain}) = P(\text{rain} | \text{sun})P_\infty(\text{sun}) + P(\text{rain} | \text{rain})P_\infty(\text{rain})$$

$$P_\infty(\text{sun}) = 0.9P_\infty(\text{sun}) + 0.3P_\infty(\text{rain})$$

$$P_\infty(\text{rain}) = 0.1P_\infty(\text{sun}) + 0.7P_\infty(\text{rain})$$

$$P_\infty(\text{sun}) = 3P_\infty(\text{rain})$$

$$P_\infty(\text{rain}) = \frac{1}{3}P_\infty(\text{sun})$$

Also: $P_\infty(\text{sun}) + P_\infty(\text{rain}) = 1$

$P_\infty(\text{sun}) = \frac{3}{4}$

$P_\infty(\text{rain}) = \frac{1}{4}$
Hidden Markov Models
Hidden Markov Models

- Markov chains not so useful for most agents
  - Need observations to update your beliefs

- Hidden Markov models (HMMs)
  - Underlying Markov chain over states $X_i$
  - You observe outputs (effects) at each time step
Example: Weather HMM

An HMM is defined by:

- Initial distribution: $P(X_1)$
- Transitions: $P(X_t \mid X_{t-1})$
- Emissions: $P(E_t \mid X_t)$

<table>
<thead>
<tr>
<th>$R_{t-1}$</th>
<th>$R_t$</th>
<th>$P(R_t \mid R_{t-1})$</th>
<th>$R_t$</th>
<th>$U_t$</th>
<th>$P(U_t \mid R_t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+r</td>
<td>+r</td>
<td>0.7</td>
<td>+r</td>
<td>+u</td>
<td>0.9</td>
</tr>
<tr>
<td>+r</td>
<td>-r</td>
<td>0.3</td>
<td>+r</td>
<td>-u</td>
<td>0.1</td>
</tr>
<tr>
<td>-r</td>
<td>+r</td>
<td>0.3</td>
<td>-r</td>
<td>+u</td>
<td>0.2</td>
</tr>
<tr>
<td>-r</td>
<td>-r</td>
<td>0.7</td>
<td>-r</td>
<td>-u</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Example: Ghostbusters HMM

- $P(X_1) = \text{uniform}$

- $P(X|X') = \text{usually move clockwise, but sometimes move in a random direction or stay in place}$

- $P(R_{ij}|X) = \text{sensor model: red means close, green means far away.}$

- $P(X_1)$:
  
  \[
  \begin{array}{ccc}
  1/9 & 1/9 & 1/9 \\
  1/9 & 1/9 & 1/9 \\
  1/9 & 1/9 & 1/9 \\
  \end{array}
  \]

- $P(X|X' = \langle 1, 2 \rangle)$:
  
  \[
  \begin{array}{ccc}
  1/6 & 1/6 & 1/2 \\
  0 & 1/6 & 0 \\
  0 & 0 & 0 \\
  \end{array}
  \]
Ghostbusters Basic Dynamics
Ghostbusters – Circular Dynamics -- HMM
Ghostbusters Circular Dynamics
Ghostbusters Whirlpool Dynamics
Conditional Independence

- HMMs have two important independence properties:
  - Markovian assumption of hidden process
  - Current observation independent of all else given current state

- Does this mean that evidence variables are guaranteed to be independent?
  - [No, they tend to correlated by the hidden state]
Real HMM Examples

- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)

- Machine translation HMMs:
  - Observations are words (tens of thousands) in language translating from
  - States are words in language translating to
Inference tasks

Prediction: $P(X_{t+k} \mid e_{1:t})$

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

Smoothing: $P(X_k \mid e_{1:t}), k < t$

Explanation: $P(X_{1:t} \mid e_{1:t})$
Filtering

Filtering: Tracking the distribution $P(X_t \mid e_1, \ldots, e_t)$ (called the belief state) over time
- $P_0(X)$ initial state (usually uniform)
- As time passes, or we get observations, update belief state

Discrete state-space (HMMs):
- Exact Inference: Forward Algorithm
- Approximate Inference: Particle Filtering

Continuous state-space (dynamical systems):
- Exact Inference: Kalman Filtering (OOS, see EE 126 or EE 221A for details)
Example: Robot Localization

Sensor model: can read in which directions there is a wall, never more than 1 mistake

Motion model: may not execute action with small prob.
Example: Robot Localization

Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake
Example: Robot Localization

\[ t=2 \]
Example: Robot Localization

\[ t=3 \]
Example: Robot Localization

Prob

0

1

t=4
Example: Robot Localization

t=5

Prob

0 1
Inference: Find State Given Evidence

- We are given evidence at each time and want to know $P(X_t | e_{1:t})$
- Idea: start with $P(X_1)$ and derive $P(X_t | e_{1:t})$ in terms of $P(X_{t-1} | e_{1:t-1})$
- Two steps: Passage of time + Incorporate Evidence

\[
P(X_{t+1} | e_{1:t}) \\
P(X_t | e_{1:t}) \\
P(X_{t+1} | e_{1:t+1})
\]
Inference: Base Cases

\[ P(X_1|e_1) \]

\[ P(X_1|e_1) = \frac{P(X_1, e_1)}{\sum_{x_1} P(x_1, e_1)} \]

\[ P(X_1|e_1) = \frac{P(e_1|X_1)P(X_1)}{\sum_{x_1} P(e_1|x_1)P(x_1)} \]

\[ P(X_2) \]

\[ P(X_2) = \sum_{x_1} P(x_1, X_2) \]

\[ P(X_2) = \sum_{x_1} P(X_2|x_1)P(x_1) \]
Passage of Time

- Assume we have current belief $P(X \mid \text{evidence to date})$

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

- Then, after one time step passes:

$P(X_{t+1}|e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t|e_{1:t})$

$= \sum_{x_t} P(X_{t+1}|x_t, e_{1:t})P(x_t|e_{1:t})$

$= \sum_{x_t} P(X_{t+1}|x_t)P(x_t|e_{1:t})$

- Basic idea: beliefs get “pushed” through the transitions
Assume we have current belief $P(X \mid \text{previous evidence})$:

$$P(X_{t+1} \mid e_{1:t})$$

Then, after evidence comes in:

$$P(X_{t+1} \mid e_{1:t+1}) = \frac{P(X_{t+1}, e_{t+1} \mid e_{1:t})}{P(e_{t+1} \mid e_{1:t})}$$

$$\propto_{X_{t+1}} P(X_{t+1}, e_{t+1} \mid e_{1:t})$$

$$= P(e_{t+1} \mid e_{1:t}, X_{t+1}) P(X_{t+1} \mid e_{1:t})$$

$$= P(e_{t+1} \mid X_{t+1}) P(X_{t+1} \mid e_{1:t})$$

- Basic idea: beliefs “rewighted” by likelihood of evidence
- Unlike passage of time, we have to renormalize
Example: Passage of Time

- As time passes, uncertainty “accumulates”

(Transition model: ghosts usually go clockwise)
Example: Observation

- As we get observations, beliefs get reweighted, uncertainty “decreases”

\[ B(X) \propto P(e|X)B'(X) \]
Example: $U_1 = +u$, $U_2 = +u$

\[
P(R_0 = +r) = 0.5 \\
P(R_0 = -r) = 0.5 \\
P(R_1 = +r) = 0.5 \\
P(R_1 = +r | +u_1) = 0.818 \\
P(R_1 = +r | +u_2) = 1.0 \\
P(R_1 = -r | +u_1) = 0.182 \\
P(R_1 = -r | +u_2) = 0.0 \\
P(R_2 = +r | +u_1) = 0.627 \\
P(R_2 = +r | +u_1, +u_2) = 0.883 \\
P(R_2 = -r | +u_1) = 0.373 \\
P(R_2 = -r | +u_1, +u_2) = 0.117
\]
Online Belief Updates

- Every time step, we start with current $P(X \mid \text{evidence})$.
- We update for time:

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

- We update for evidence:

$$P(x_t \mid e_{1:t}) \propto_X P(x_t \mid e_{1:t-1}) \cdot P(e_t \mid x_t)$$

- The forward algorithm does both at once (and doesn't normalize)
The Forward Algorithm

- We are given evidence at each time and want to know
  \[ P(X_t|e_{1:t}) \]

- We can derive the following updates
  \[
  P(x_t|e_{1:t}) \propto X_t P(x_t, e_{1:t}) \\
  = \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\
  = \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t) \\
  = P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})
  \]

We can normalize as we go if we want to have \( P(x|e) \) at each time step, or just once at the end...
Video of Demo Pacman – Sonar (with beliefs)
Most Likely Explanation
HMMs: MLSE 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
Most likely explanation = most probable path

- **State trellis**: graph of states and transitions over time

  - Each arc represents some transition $X_{t-1} \rightarrow X_t$
  - Each arc has weight $P(x_t \mid x_{t-1}) P(e_t \mid x_t)$ (arcs to initial states have weight $P(x_0)$)
  - The **product** of weights on a path is proportional to that state seq’s probability
  - Forward algorithm: sums of paths
  - **Viterbi algorithm**: best paths
    - Dynamic Programming: solve subproblems, combine them as you go along

\[
\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t}) = \arg\max_{x_{1:t}} P(x_{1:t}, e_{1:t}) = \arg\max_{x_{1:t}} P(x_{1:t}, e_{1:t}) = \arg\max_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t)
\]
Forward Algorithm (Sum)
For each state at time $t$, keep track of the total probability of all paths to it

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

Viterbi Algorithm (Max)
For each state at time $t$, keep track of the maximum probability of any path to it

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

Time complexity? \( O(|X|^2 T) \)

Space complexity? \( O(|X| T) \)

Number of paths? \( O(|X|^T) \)
argmax of product of probabilities
= argmin of sum of negative log probabilities
= minimum-cost path

Viterbi is essentially uniform cost graph search
Viterbi Algorithm Pseudocode

function VITERBI(O, S, Π, Y, A, B) : X
    for each state $i = 1, 2, \ldots, K$ do
        $T_1[i, 1] \leftarrow \pi_i \cdot B_{iy_1}$
        $T_2[i, 1] \leftarrow 0$
    end for
    for each observation $j = 2, 3, \ldots, T$ do
        for each state $i = 1, 2, \ldots, K$ do
            $T_1[i, j] \leftarrow \max_k (T_1[k, j - 1] \cdot A_{ki} \cdot B_{iy_j})$
            $T_2[i, j] \leftarrow \arg\max_k (T_1[k, j - 1] \cdot A_{ki} \cdot B_{iy_j})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $x_T \leftarrow s_{z_T}$
    for $j = T, T - 1, \ldots, 2$ do
        $z_{j-1} \leftarrow T_2[z_j, j]$
        $x_{j-1} \leftarrow s_{z_{j-1}}$
    end for
    return $X$
end function

Observation Space $O = \{o_1, o_2, \ldots, o_N\}$
State Space $S = \{s_1, s_2, \ldots, s_K\}$
Initial probabilities $\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$
Observations $Y = (y_1, y_2, \ldots, y_T)$
Transition Matrix $A \in \mathbb{R}^{K \times K}$
Emission Matrix $B \in \mathbb{R}^{K \times N}$

Matrix $T_1[i, j]$ stores probabilities of most likely path so far with $x_j = s_i$

Matrix $T_2[i, j]$ stores $x_{j-1}$ of most likely path so far with $x_j = s_i$
Particle Filtering
Approximate Inference on HMMs

- When \(|X|\) is more than \(10^6\) or so (e.g., 3 ghosts in a 10x20 world), exact inference becomes infeasible.
- Likelihood weighting fails completely – number of samples needed grows *exponentially* with \(T\).

![Graph showing time step vs. Avg absolute error for different methods (LW(25), LW(100), LW(1000), LW(10000), ER/SOF(25))](image)

![Diagram of a sequence of states (\(X_0, X_1, X_2, X_3\)) with transitions labeled \(E_1, E_2, E_3\).](diagram)
We need a new idea!

- The problem: sample state trajectories go off into low-probability regions, ignoring the evidence; too few “reasonable” samples
- Solution: kill the bad ones, make more of the good ones
- This way the population of samples stays in the high-probability region
- This is called **resampling** or survival of the fittest
Particle Filtering

- **Filtering: approximate solution**
- **Sometimes |X| is too big to use exact inference**
  - |X| may be too big to even store P(X | e_{1:T})
- **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**
Our representation of $P(X)$ is now a list of $N$ particles (samples)
- Generally, $N \ll |X|$

$P(x)$ approximated by number of particles with value $x$
- So, many $x$ may have $P(x) = 0!$
- More particles, more accuracy
- Usually we want a low-dimensional marginal
  - E.g., “Where is ghost 1?” rather than “Are ghosts 1,2,3 in [2,6], [5,6], and [8,11]?”

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 – sample’s 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)
After observing Evidence $e_{t+1}$:

- 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 $(N \times \text{an approximation of } P(e)$))
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

![Particle Filtering Diagram]
Video of Demo – Moderate Number of Particles
Video of Demo – One Particle
Video of Demo – Huge Number of Particles
In robot localization:

- Know the map, but not the robot’s position
- Observations may be vectors of range finder readings
- State space and readings typically continuous (very fine grid) and so we cannot store $P(X_t \mid e_{1:t})$
- Particle filtering is a main technique
Particle Filter Localization (Sonar)

Global localization with sonar sensors

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

[Dieter Fox, et al.]
Particle Filter Localization (Laser)

[Dieter Fox, et al.]
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]
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
## DBNs and HMMs

- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables
- Sparse dependencies => exponentially fewer parameters in DBN
  - E.g., 20 state variables, 3 parents each;
    - DBN has $20 \times 2^3 = 160$ parameters, HMM has $2^{20} \times 2^{20} \approx 10^{12}$ parameters
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

- 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: $P(E_1^a | G_1^a) \times P(E_1^b | G_1^b)$

- **Resample**: Select prior samples (tuples of values) in proportion to their likelihood