CS 188: Artificial Intelligence

Hidden Markov Models

Summer 2024: Eve Fleisig & Evgeny Pobachienko

[Slides adapted from Saagar Sanghavi, Dan Klein, Pieter Abbeel, Anca Dragan, Stuart Russell]

Reasoning over Time or Space

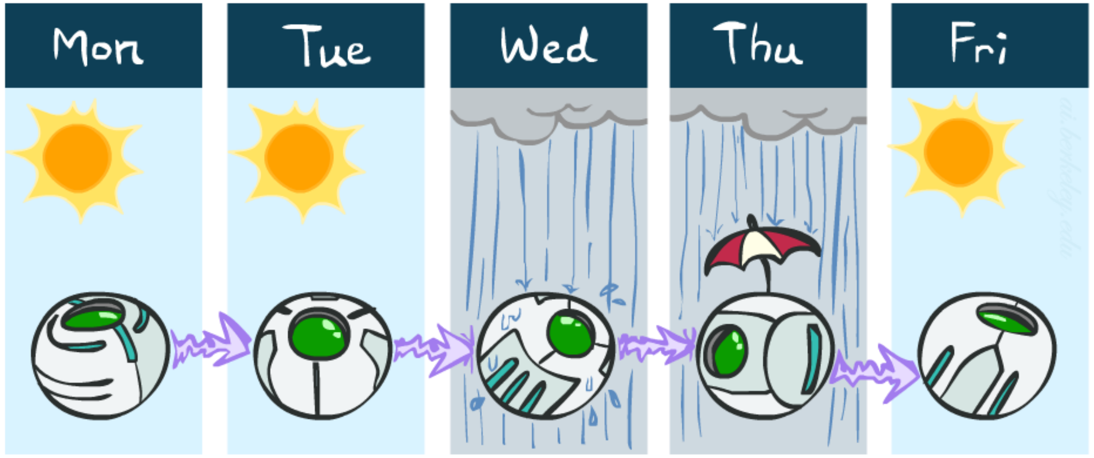
- Often, we want to reason about a sequence of observations
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring
- Need to introduce time (or space) into our models

Example Markov Chain: Weather

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

- Initial distribution:

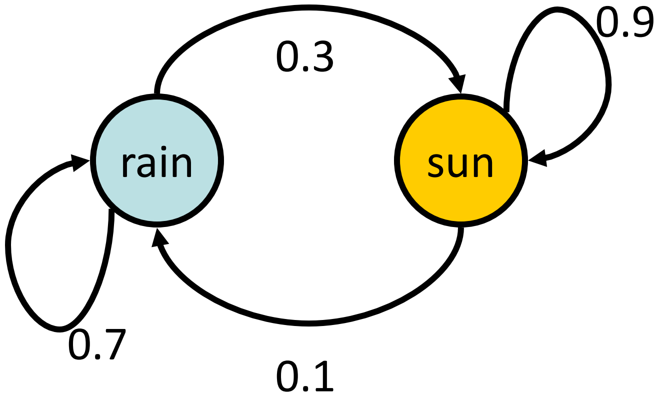
P(X_0)	
sun	rain
1	0.0

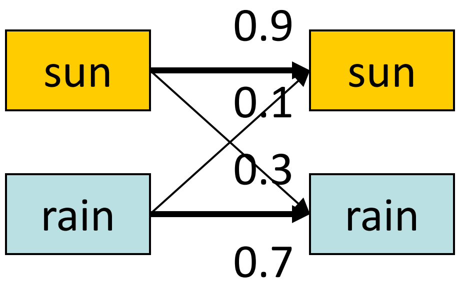


- CPT $P(X_t \mid X_{t-1})$:

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

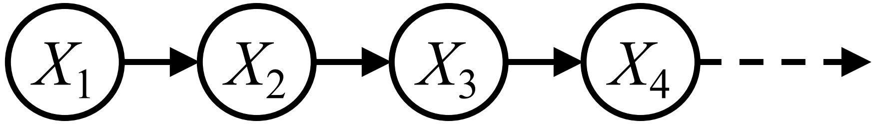
Two new ways of representing the same CPT





Markov Chains

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

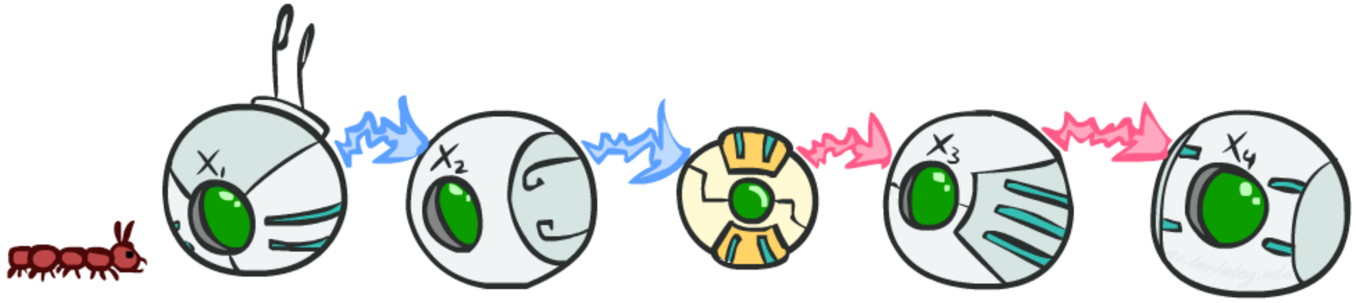


$$P(X_1) \qquad P(X_t|X_{t-1})$$

$$P(X_t) = ?$$

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

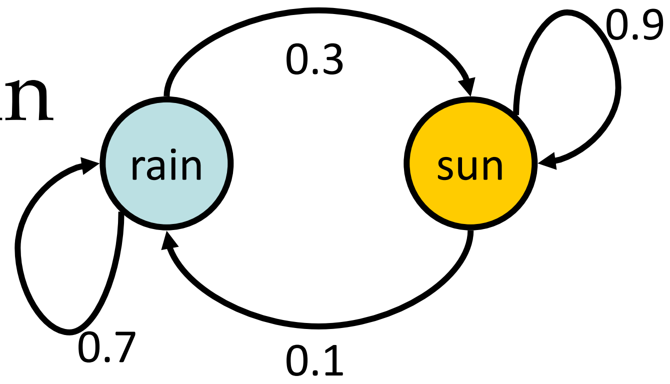
Markov Assumption



- 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



X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

- What is the probability distribution after one step?

$$P(X_2 = sun) = \sum_{x_1} P(x_1, X_2 = sun) = \sum_{x_1} P(X_2 = sun \mid x_1) P(x_1)$$

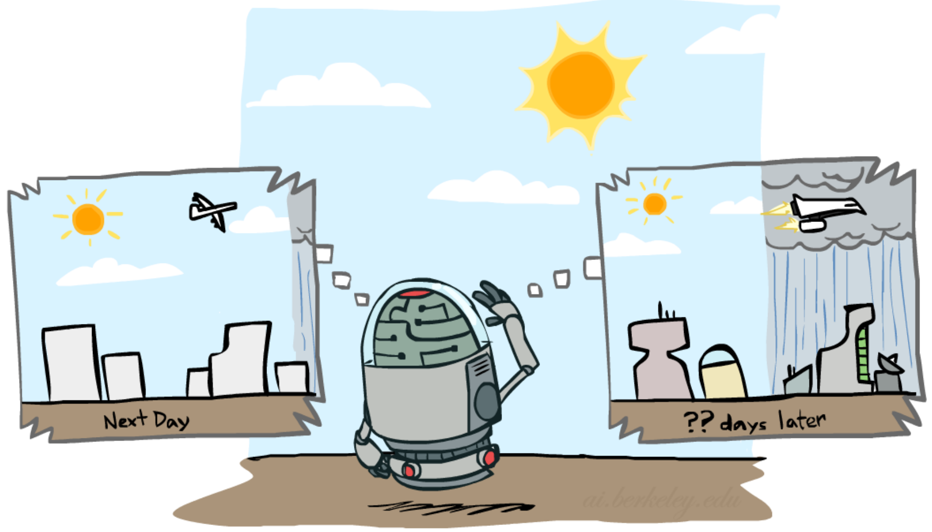
$$P(X_2 = \text{sun}) = P(X_2 = \text{sun} \mid X_1 = \text{sun}) P(X_1 = \text{sun}) + P(X_2 = \text{sun} \mid X_1 = \text{rain}) P(X_1 = \text{rain})$$

$$0.9 \cdot 1.0 + 0.3 \cdot 0.0 = 0.9$$

Mini-Forward Algorithm

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

$$X_1 \to X_2 \to X_3 \to X_4 \dashrightarrow$$



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

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

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

Forward simulation

Example Run of Mini-Forward Algorithm

- From initial observation of sun

$$\left\langle \begin{array}{c} 1.0 \\ 0.0 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.9 \\ 0.1 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.84 \\ 0.16 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.804 \\ 0.196 \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} 0.75 \\ 0.25 \end{array} \right\rangle$$

$\mathrm{P}(X_1)$ $\mathrm{P}(X_2)$ $\mathrm{P}(X_3)$ $\mathrm{P}(X_4)$ $\mathrm{P}(X_\infty)$

- From initial observation of rain

$$\left\langle \begin{array}{c} 0.0 \\ 1.0 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.3 \\ 0.7 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.48 \\ 0.52 \end{array} \right\rangle \quad \left\langle \begin{array}{c} 0.588 \\ 0.412 \end{array} \right\rangle \Rightarrow \left\langle \begin{array}{c} 0.75 \\ 0.25 \end{array} \right\rangle$$

$\mathrm{P}(X_1)$ $\mathrm{P}(X_2)$ $\mathrm{P}(X_3)$ $\mathrm{P}(X_4)$ $\mathrm{P}(X_\infty)$

- From yet another initial distribution $\mathrm{P}(X_1)$:

$$\left\langle \begin{array}{c} p \\ 1-p \end{array} \right\rangle \quad \cdots \quad \Rightarrow \left\langle \begin{array}{c} 0.75 \\ 0.25 \end{array} \right\rangle$$

$\mathrm{P}(X_1)$ $\mathrm{P}(X_\infty)$

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

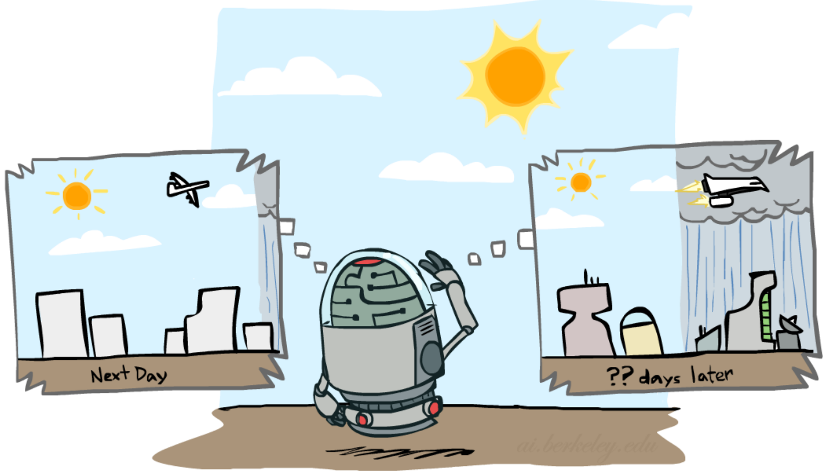
- **Stationary distribution:**
  - The distribution we end up with is called the stationary distribution P_∞ of the chain
  - It satisfies

$$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$$

Example: Stationary Distribution

- Question: What's P(X) at time t = infinity?

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \dashrightarrow$$



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

$$P_\infty(sun) = 0.9P_\infty(sun) + 0.3P_\infty(rain)$$
$$P_\infty(rain) = 0.1P_\infty(sun) + 0.7P_\infty(rain)$$

$$P_\infty(sun) = 3P_\infty(rain)$$
$$P_\infty(rain) = 1/3P_\infty(sun)$$

X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

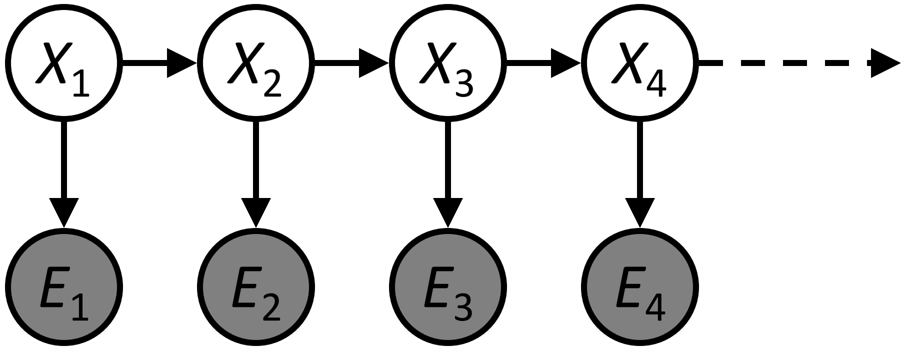
Also: $P_\infty(sun) + P_\infty(rain) = 1$ $\Rightarrow$

$$P_\infty(sun) = 3/4$$
$$P_\infty(rain) = 1/4$$

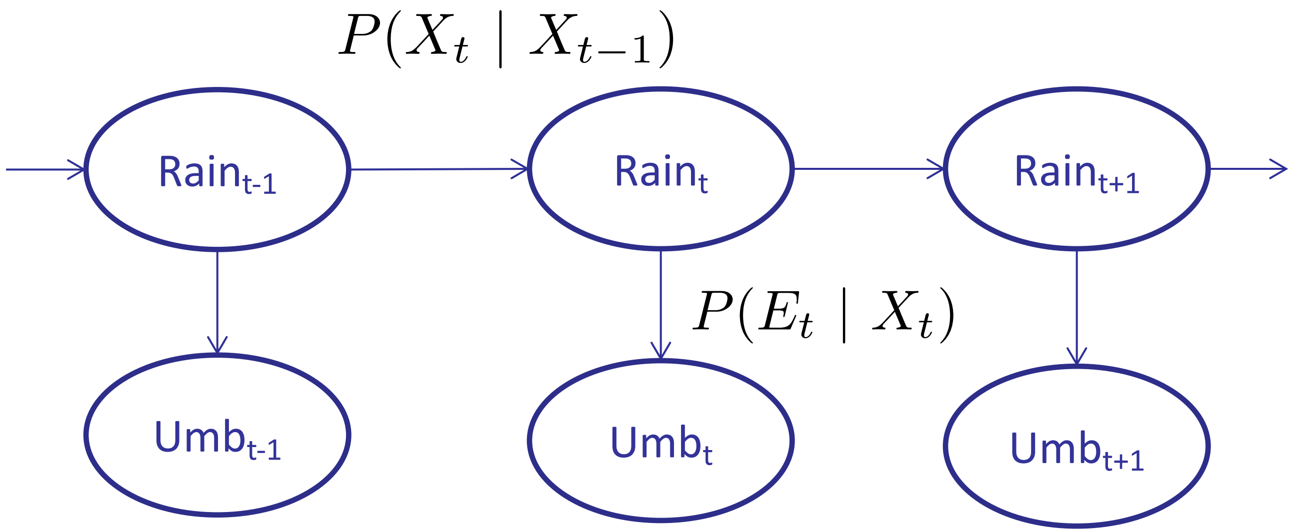
Hidden Markov Models

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



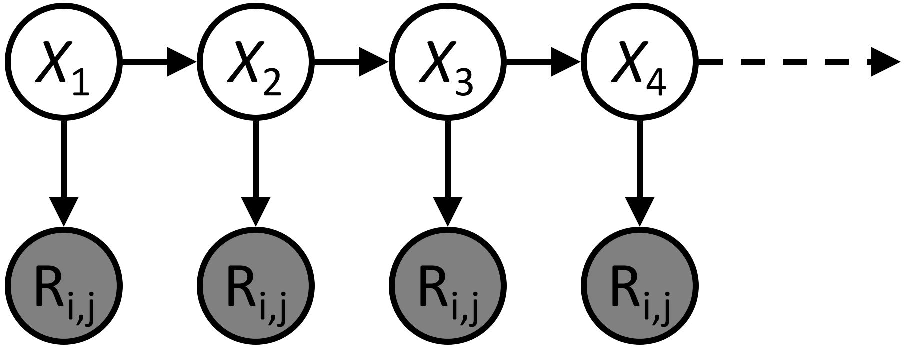
- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t \mid X_{t-1})$
  - Emissions: $P(E_t \mid X_t)$

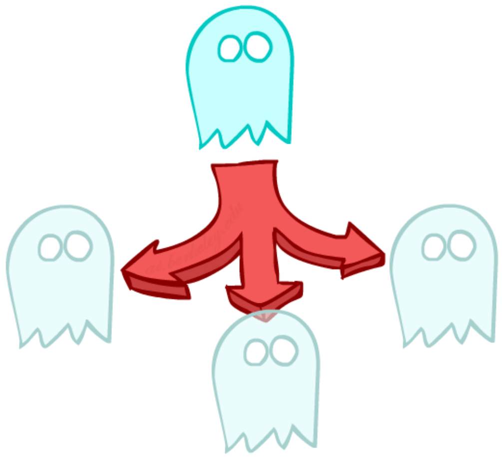
R_{t-1}	R_t	$P(R_t \mid R_{t-1})$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

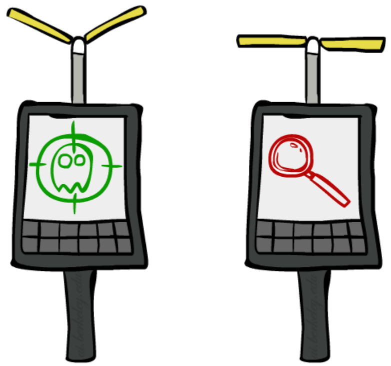
R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Example: Ghostbusters HMM

- $P(X_1)$ = uniform
- $P(X|X')$ = usually move clockwise, but sometimes move in a random direction or stay in place
- $P(R_{ij}|X)$ = sensor model: red means close, green means far away.







1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_1)$

1/6	1/6	1/2
0	1/6	0
0	0	0

$P(X|X' = <1,2>)$

Ghostbusters Basic Dynamics

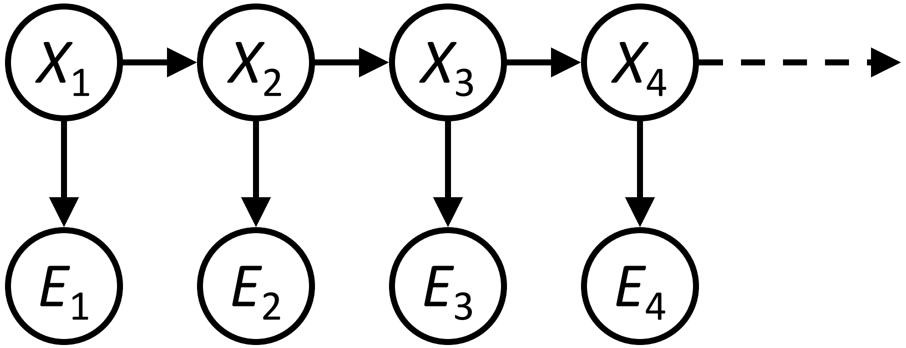
Ghostbusters – Circular Dynamics -- HMM

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

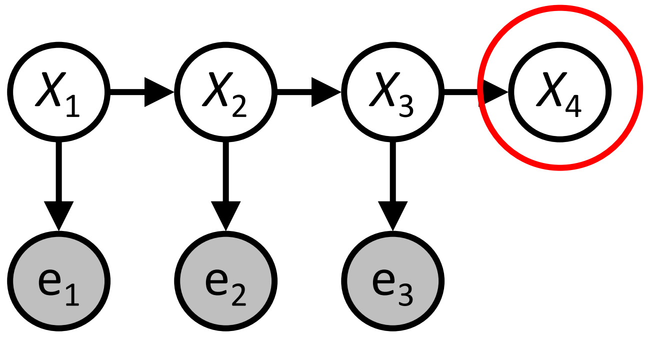
- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

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

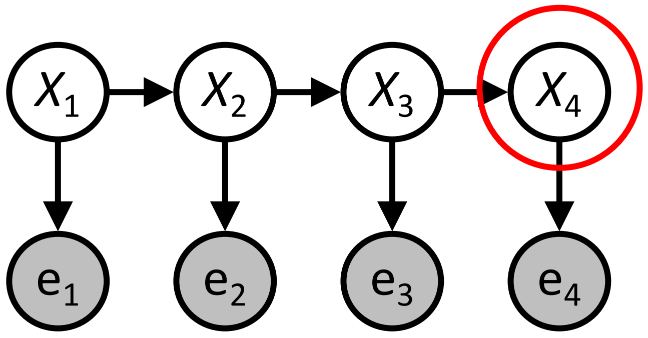
- Machine translation HMMs:
  - Observations are words (tens of thousands) in language translating from
  - States are words in language translating to

Inference tasks

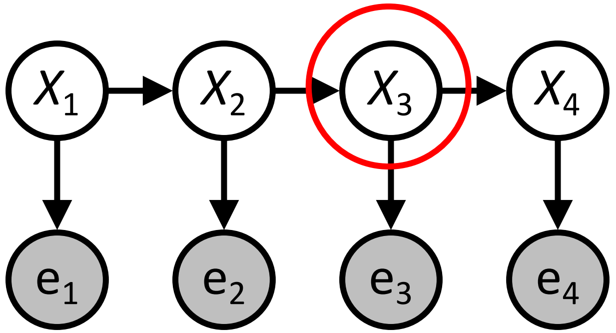
Prediction: $P(X_{t+k} \mid e_{1:t})$



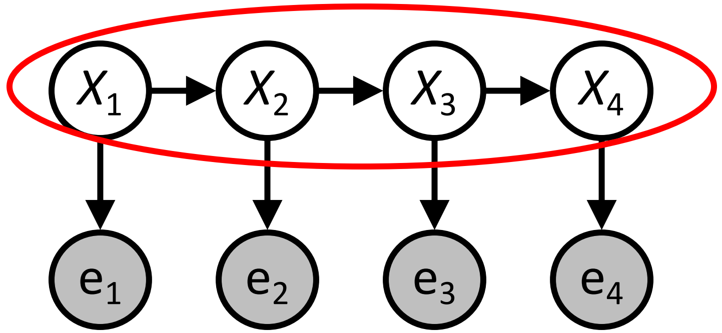
Filtering: $P(X_t \mid e_{1:t})$



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



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

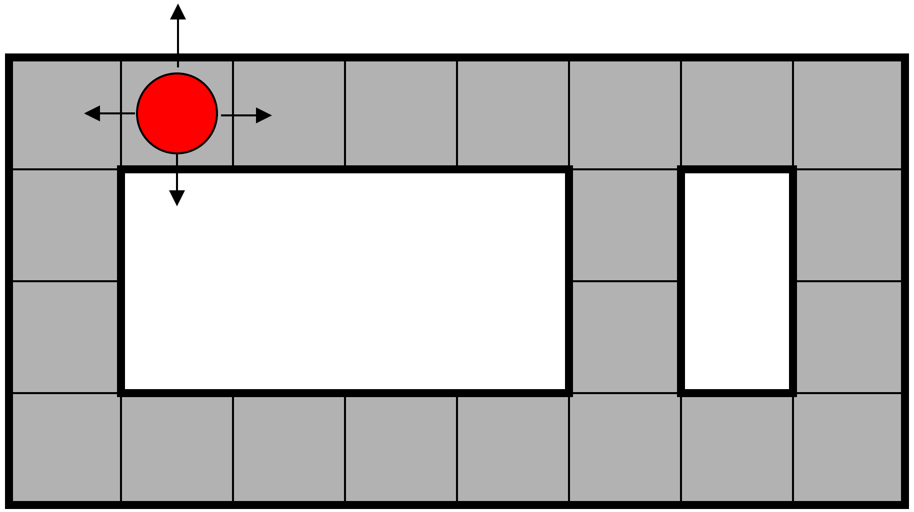


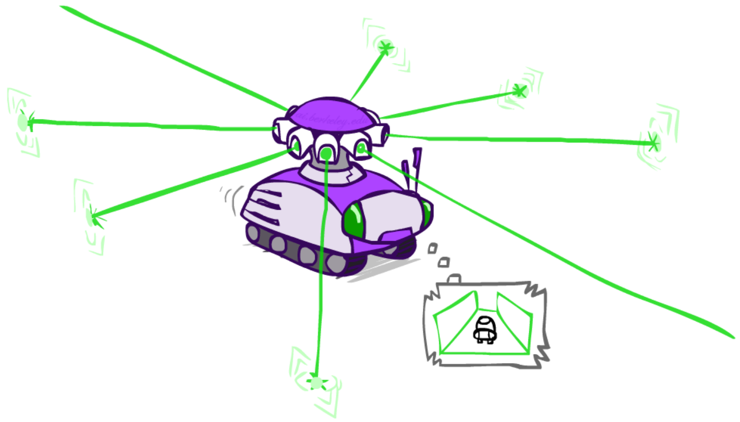
Filtering

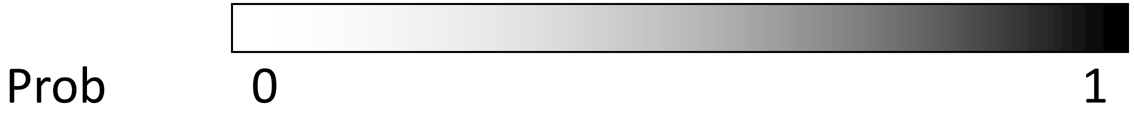
- Filtering: Tracking the distribution $P(X_t \mid e_1, \ldots, e_t)$ (called the belief state) over time
  - $P_0(X)$ initial state (usually uniform)
  - As time passes, or we get observations, update belief state

- 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





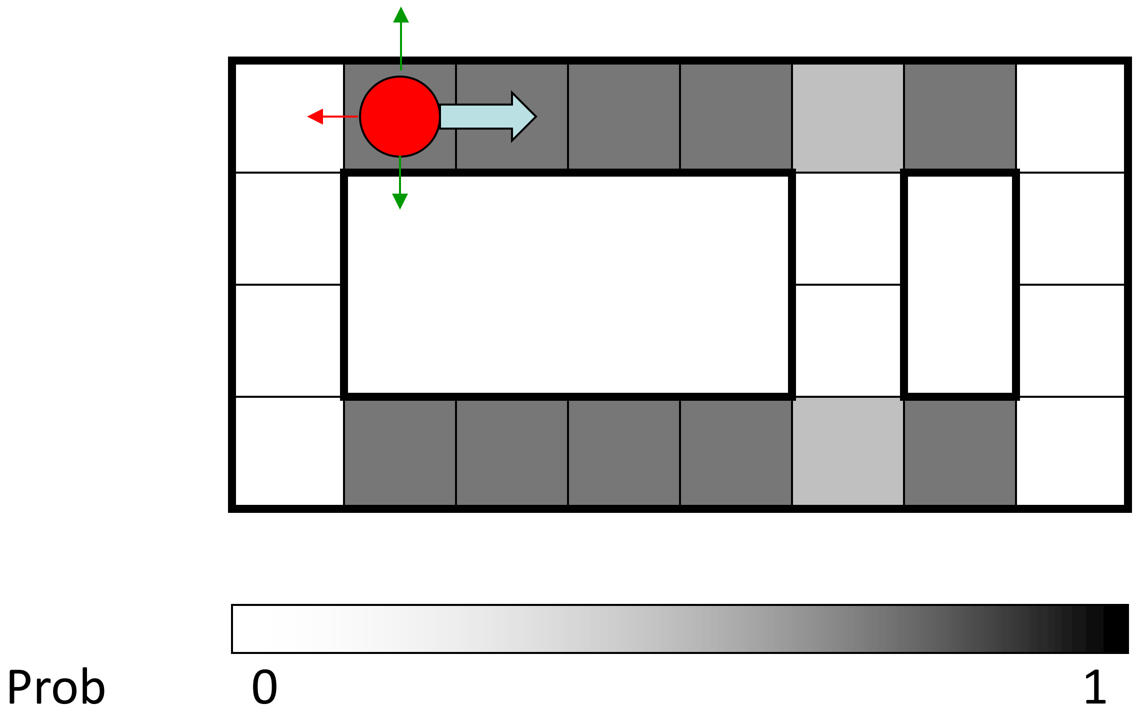


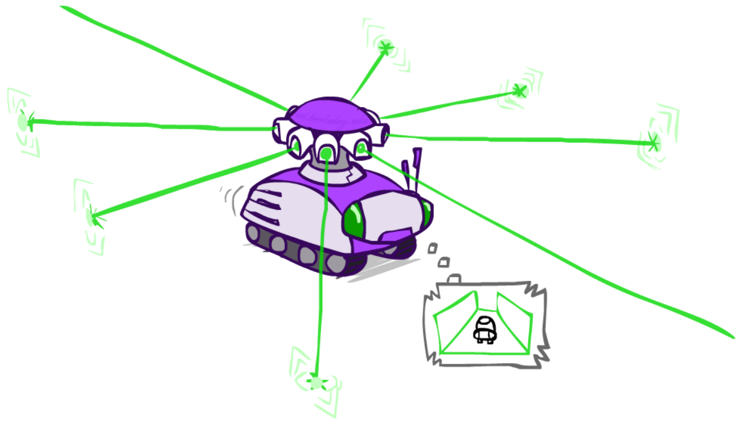
t=0

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

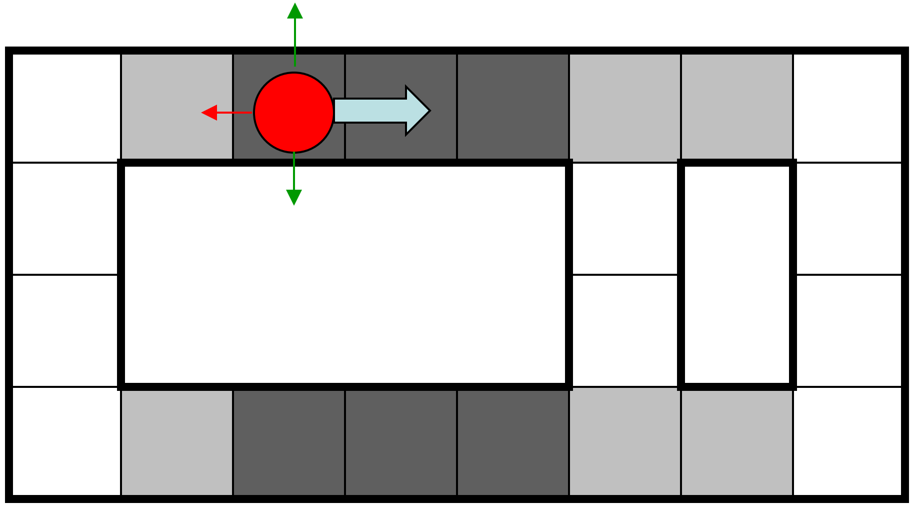


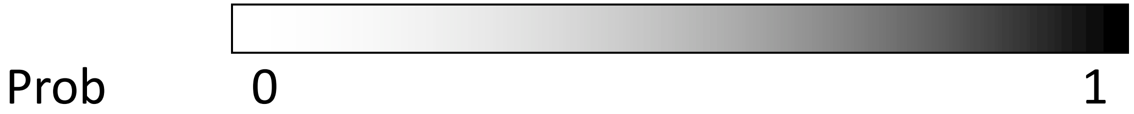


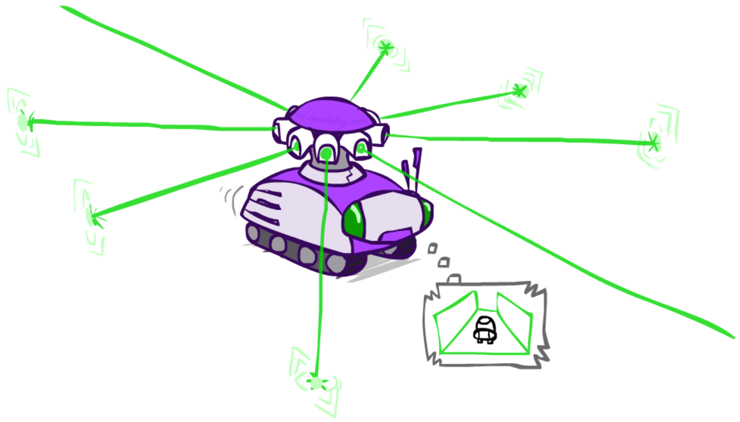
t=1

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

Example: Robot Localization

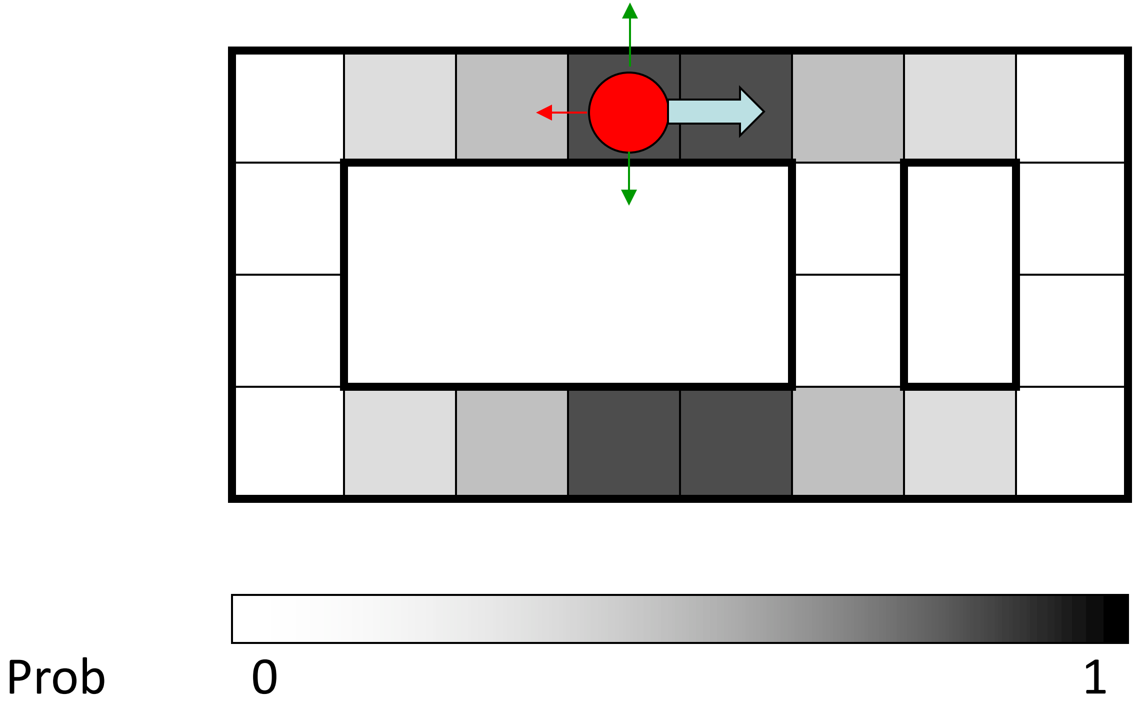


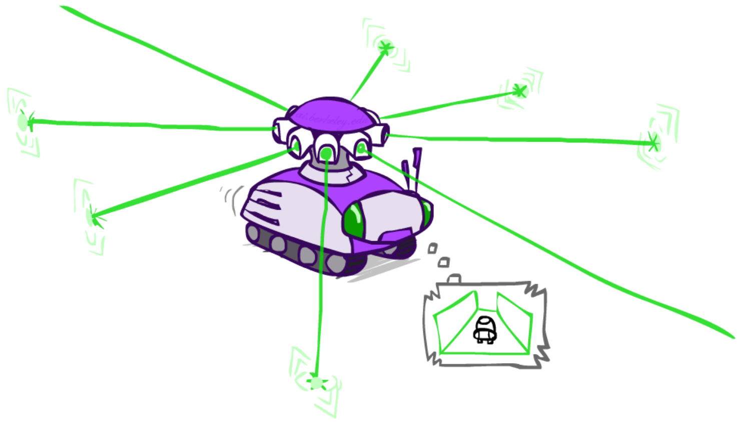




t=2

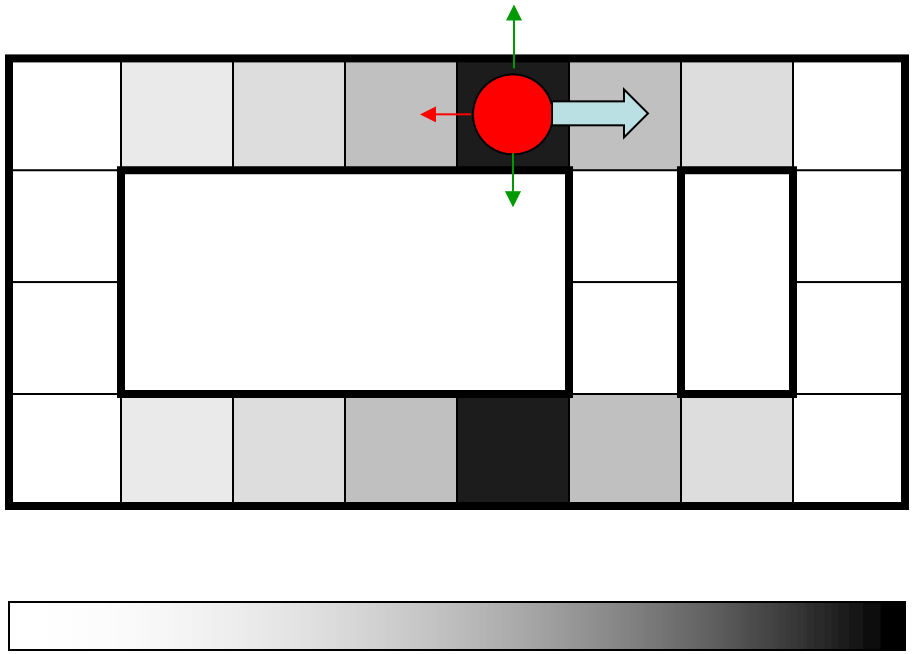
Example: Robot Localization





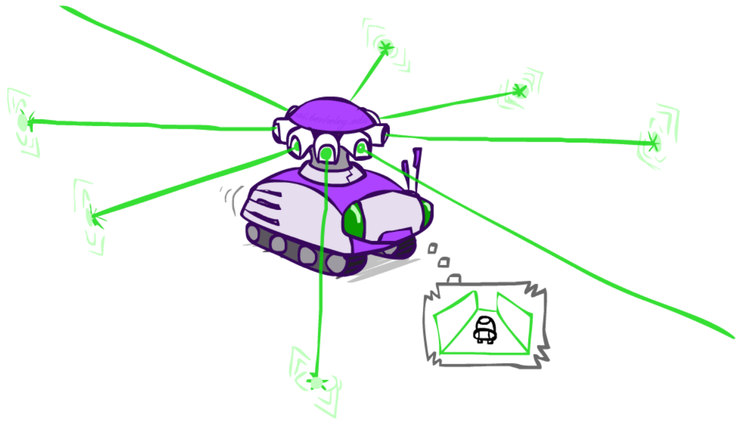
t=3

Example: Robot Localization

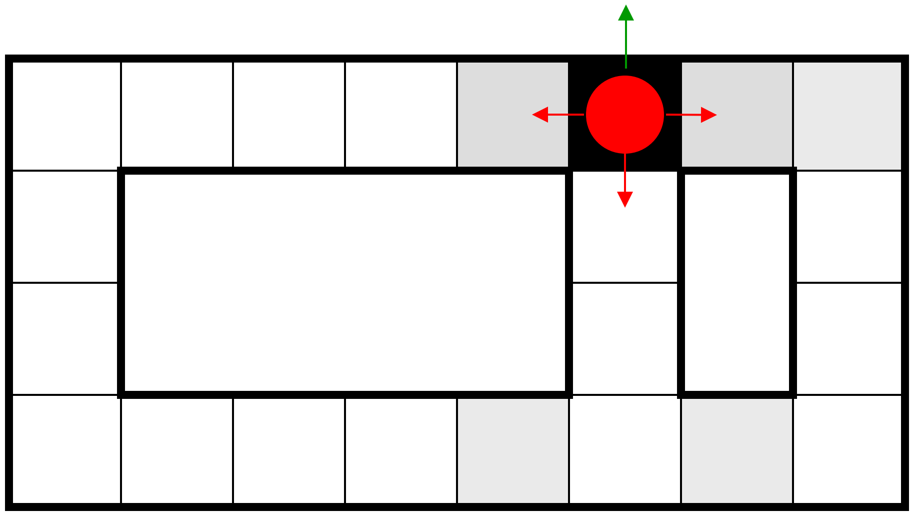


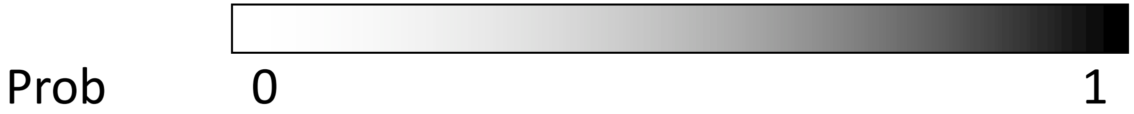
Prob 0 1

t=4

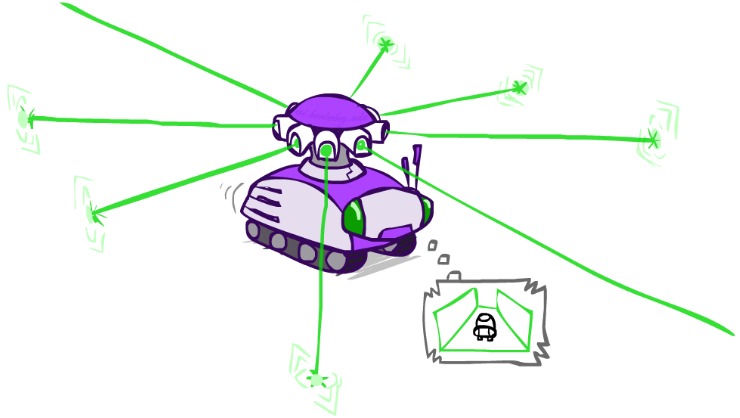


Example: Robot Localization





t=5

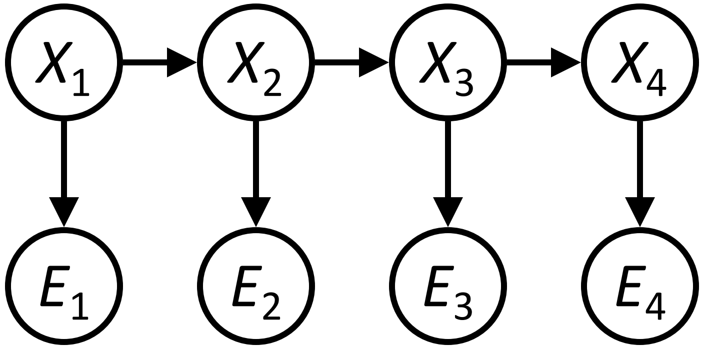


Inference: Find State Given Evidence

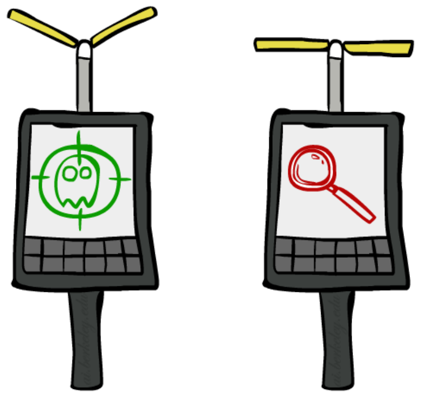
- We are given evidence at each time and want to know $P(X_t|e_{1:t})$
- Idea: start with $P(X_1)$ and derive $P(X_t \mid e_{1:t})$ in terms of $P(X_{t-1} \mid e_{1:t-1})$
- Two steps: Passage of time + Incorporate Evidence

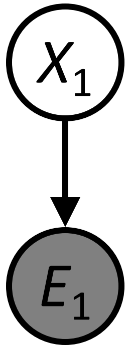
$P(X_{t+1} \mid e_{1:t})$

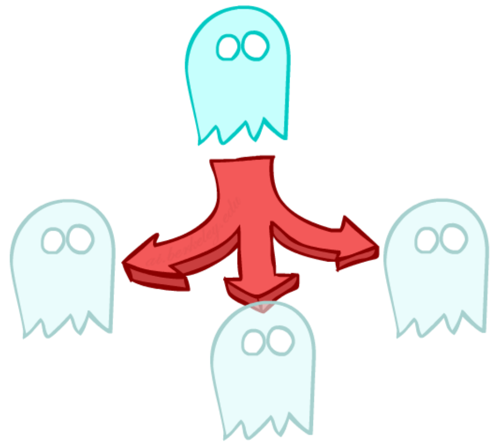
$P(X_t \mid e_{1:t})$ $P(X_{t+1} \mid e_{1:t+1})$

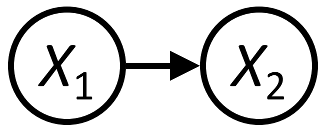


Inference: Base Cases









$$P(X_1|e_1)$$

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

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

$$P(X_2)$$

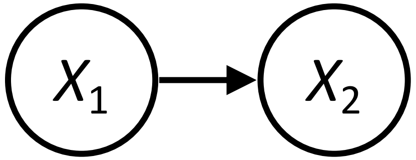
$$P(X_2) = \sum_{x_1} P(x_1, X_2)$$

$$P(X_2) = \sum_{x_1} P(X_2|x_1)P(x_1)$$

Passage of Time

- Assume we have current belief P(X | 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

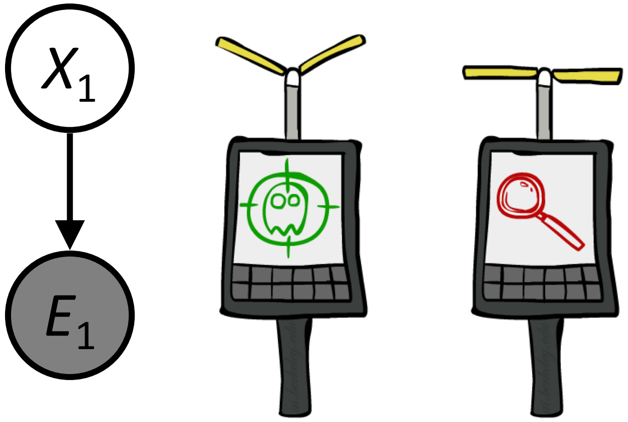
Observation

- Assume we have current belief P(X | previous evidence):

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

- Then, after evidence comes in:

$$\begin{aligned}
P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t}) \\
&\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t}) \\
&= P(e_{t+1}|e_{1:t}, X_{t+1})P(X_{t+1}|e_{1:t}) \\
&= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t})
\end{aligned}$$



  - Basic idea: beliefs “reweighted” 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)

<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	1.00	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

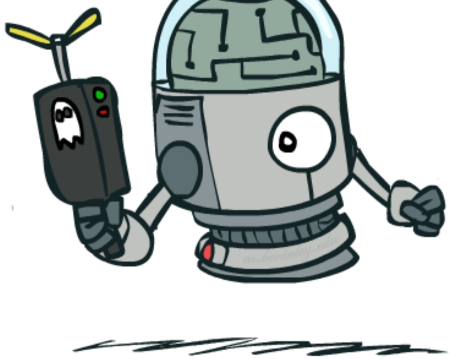
T = 1

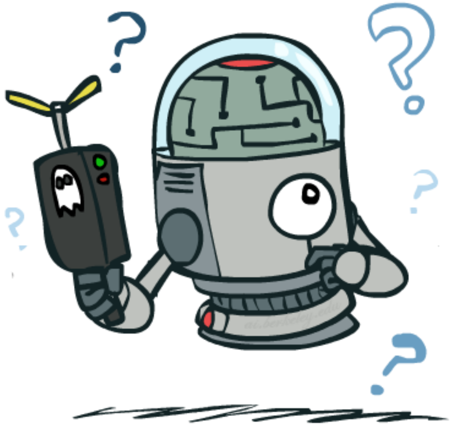
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

T = 2

0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

T = 5





Example: Observation

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

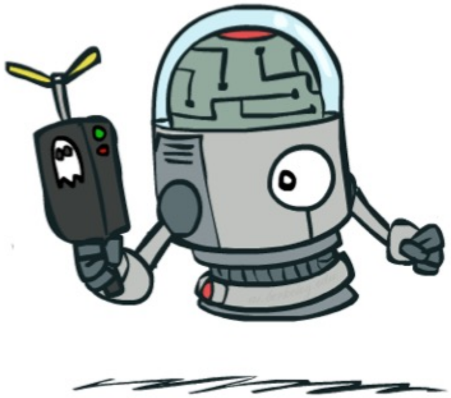
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

<0.01	<0.01	<0.01	<0.01	0.02	<0.01
<0.01	<0.01	<0.01	0.83	0.02	<0.01
<0.01	<0.01	0.11	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

After observation

$$B(X) \propto P(e|X)B'(X)$$



Example: $U_1 = +u$, $U_2 = +u$

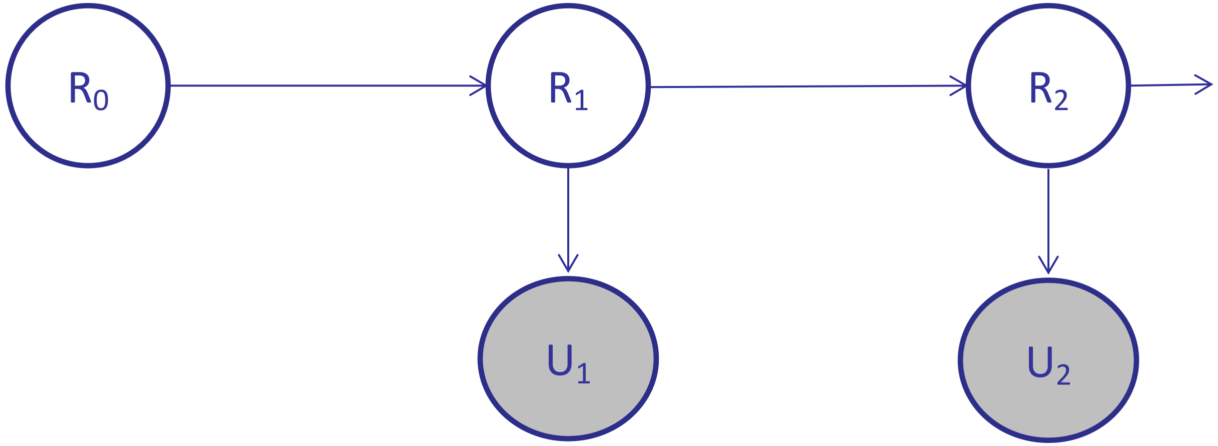
$P(R_1 = +r) = 0.5$
$P(R_1 = -r) = 0.5$

$P(R_2 = +r \mid +u_1) = 0.627$
$P(R_2 = -r \mid +u_1) = 0.373$

$P(R_0 = +r) = 0.5$
$P(R_0 = -r) = 0.5$

$P(R_1 = +r \mid +u_1) = 0.818$
$P(R_1 = -r \mid +u_1) = 0.182$

$P(R_2 = +r \mid +u_1, +u_2) = 0.883$
$P(R_2 = -r \mid +u_1, +u_2) = 0.117$



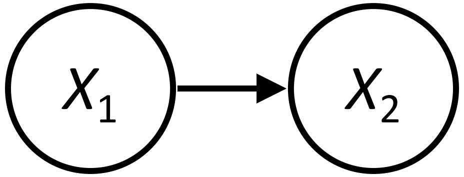
R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Online Belief Updates

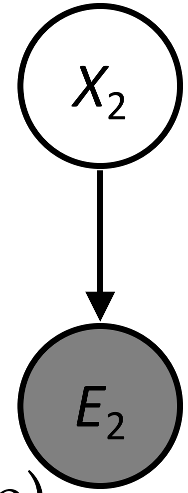
- Every time step, we start with current P(X | evidence)
- We update for time:

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



- We update for evidence:

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



- 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

$$P(x_t|e_{1:t}) \propto_{X_t} P(x_t, e_{1:t})$$

We can normalize as we go if we want to have P(x|e) at each time step, or just once at the end…

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

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

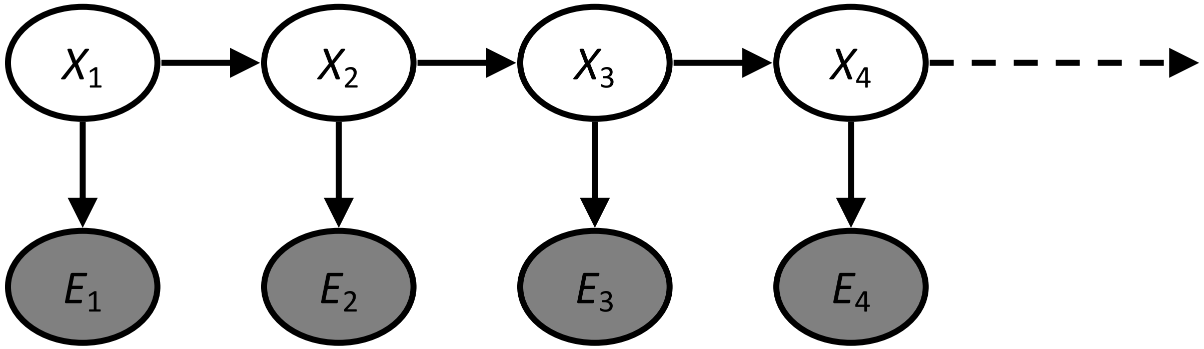
$$= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})$$

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: $P(E|X)$

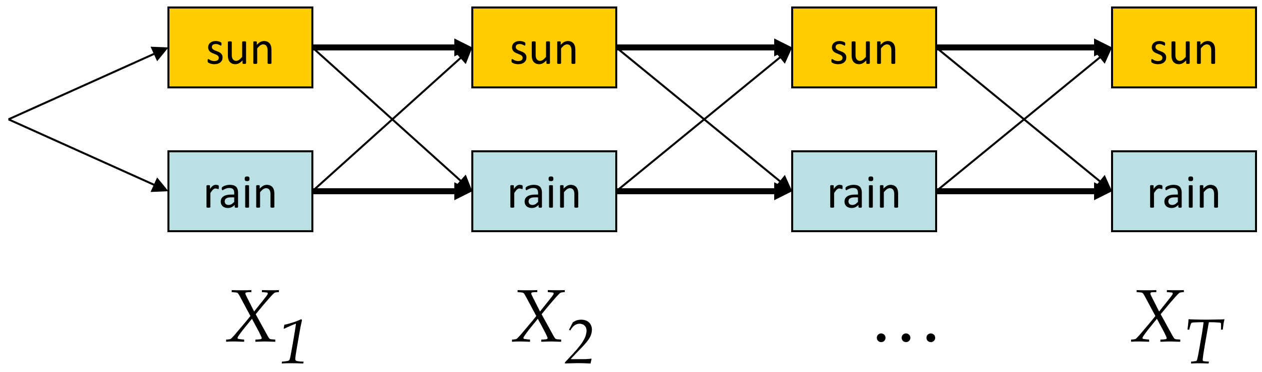


- New query: most likely explanation: $\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$

- New method: the Viterbi algorithm

Most likely explanation = most probable path

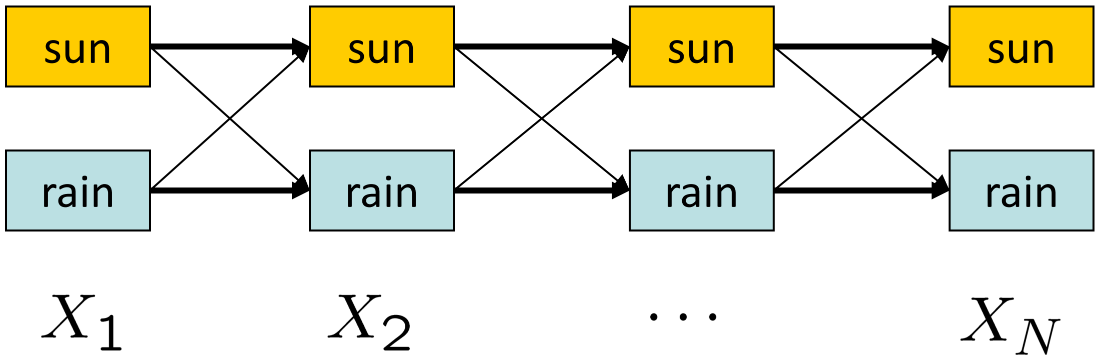
- **State trellis**: graph of states and transitions over time



$\text{argmax}_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
$= \text{argmax}_{x_{1:t}} P(x_{1:t}, e_{1:t})$
$= \text{argmax}_{x_{1:t}} \; P(x_{1:t}, e_{1:t})$
$= \text{argmax}_{x_{1:t}} P(x_0) \prod_t P(x_t \mid x_{t\text{-}1}) \, P(e_t \mid x_t)$

- 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

Forward / Viterbi Algorithms



Forward Algorithm (Sum)

For each state at time t, keep track of the ***total probability of all paths*** to it

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

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

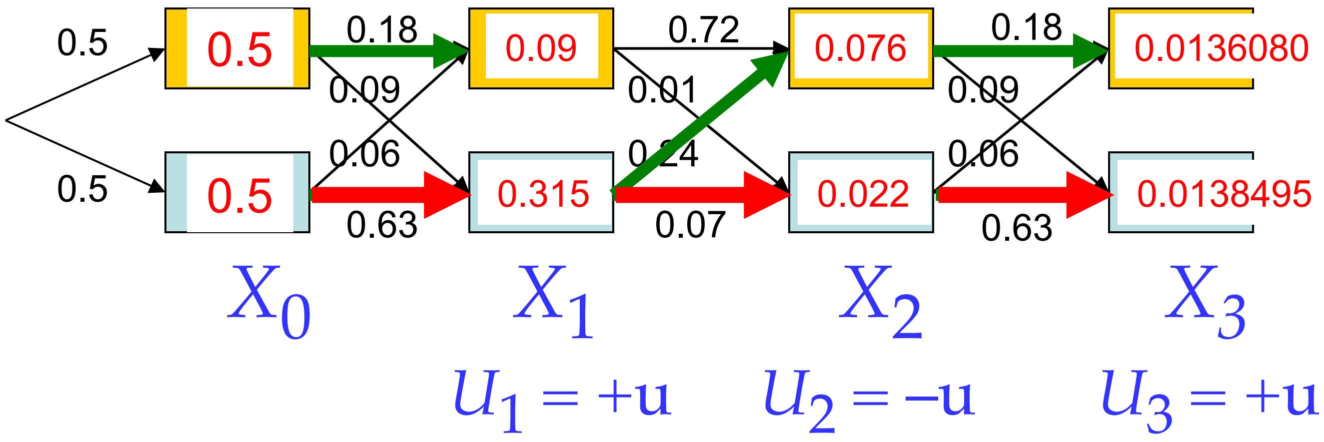
Viterbi Algorithm (Max)

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



R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.1
-r	-r	0.9

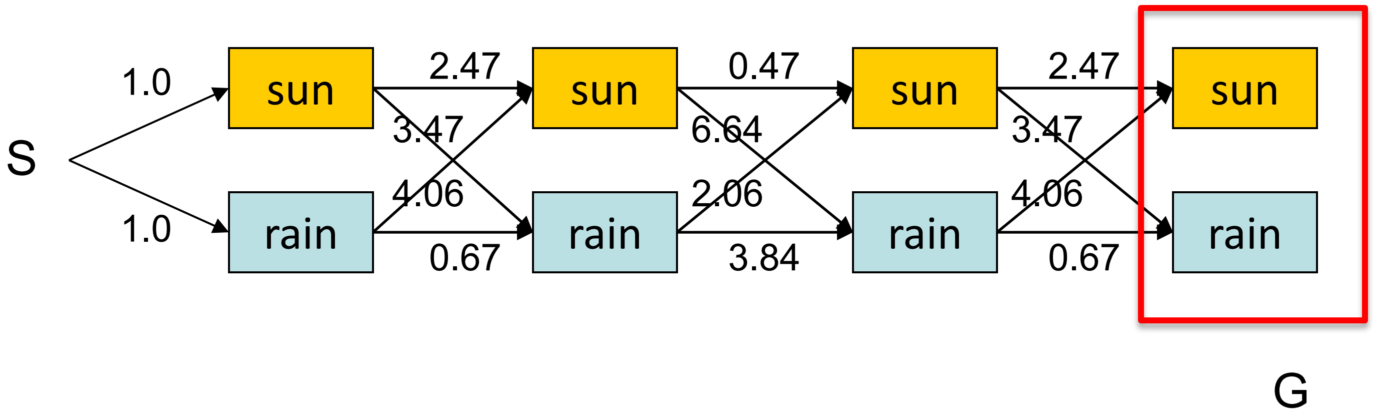
R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

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

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

Number of paths?
$O(|X|^T)$

Viterbi in negative log space



W_{t-1}	$P(W_t \mid W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

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

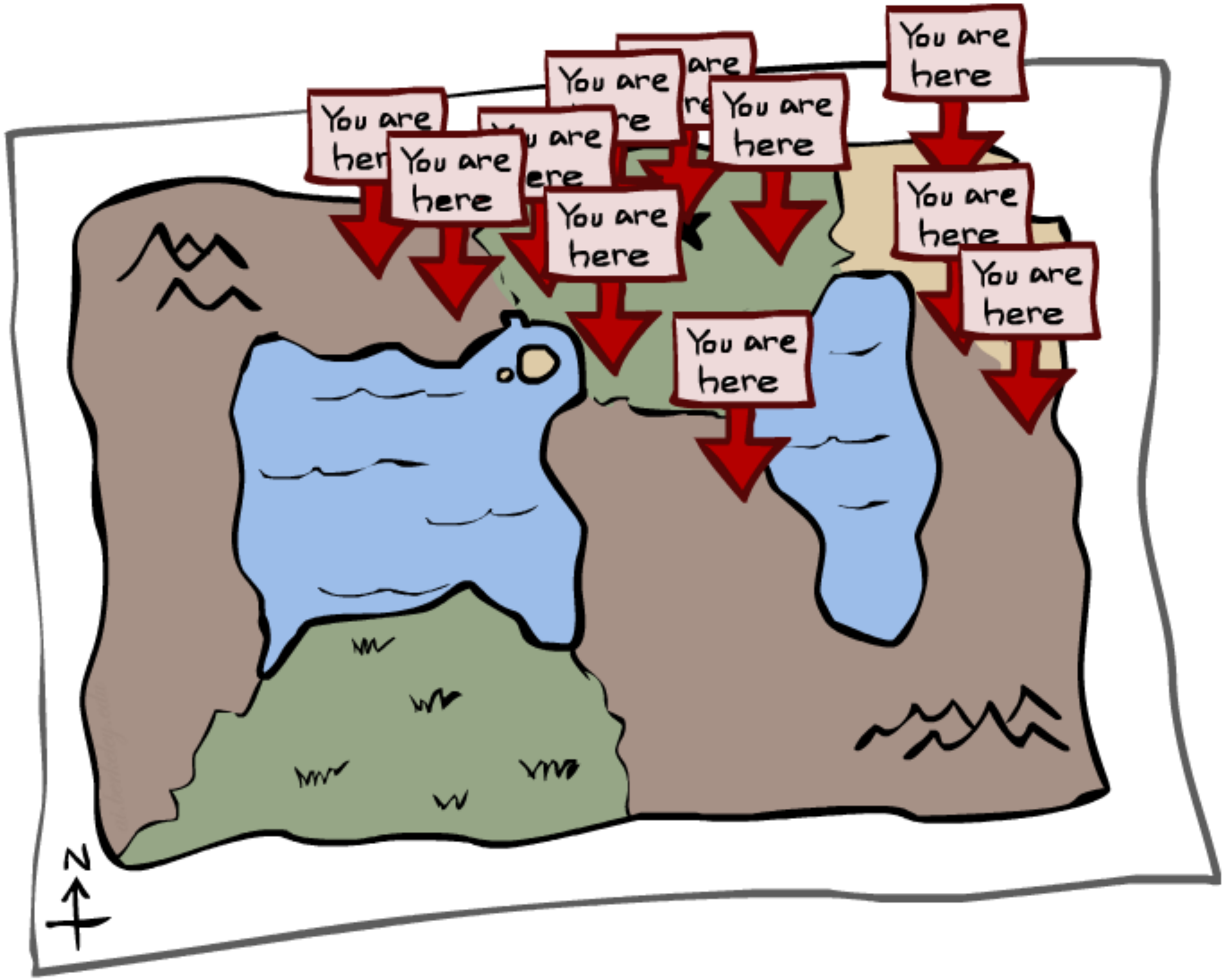
```
function VITERBI(O, S, Π, Y, A, B) : X
    for each state i = 1, 2, ..., K do
        T_1[i, 1] ← π_i · B_{iy_1}
        T_2[i, 1] ← 0
    end for
    for each observation j = 2, 3, ..., T do
        for each state i = 1, 2, ..., K do
            T_1[i, j] ← max_k (T_1[k, j-1] · A_{ki} · B_{iy_j})
            T_2[i, j] ← argmax_k (T_1[k, j-1] · A_{ki} · B_{iy_j})
        end for
    end for
    z_T ← argmax_k (T_1[k, T])
    x_T ← s_{z_T}
    for j = T, T-1, ..., 2 do
        z_{j-1} ← T_2[z_j, j]
        x_{j-1} ← 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 $x_j = s_i$

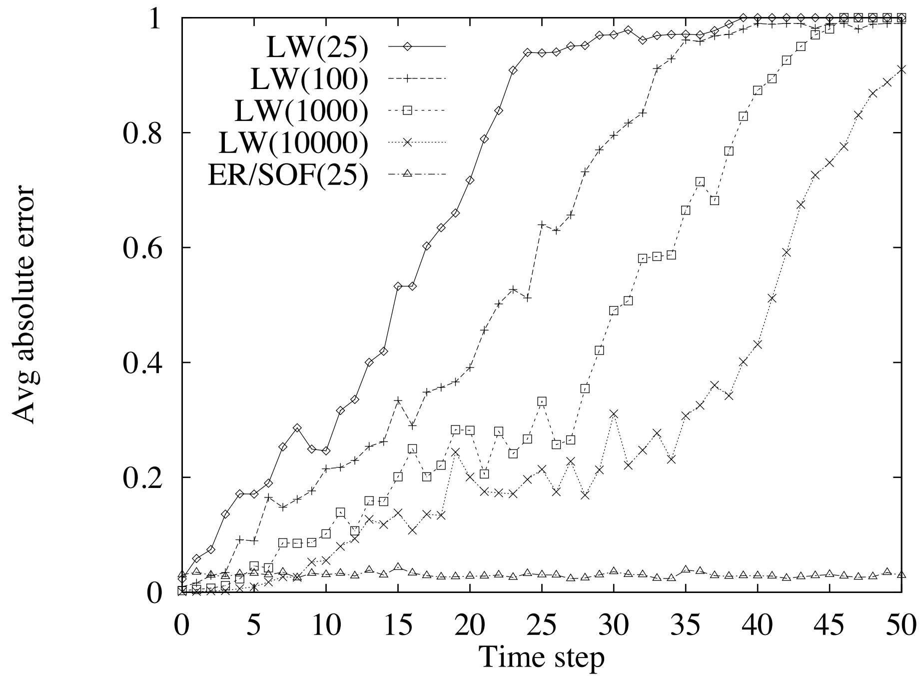
Matrix $T_2[i, j]$ stores x_{j-1} of most likely path so far with $x_j = s_i$

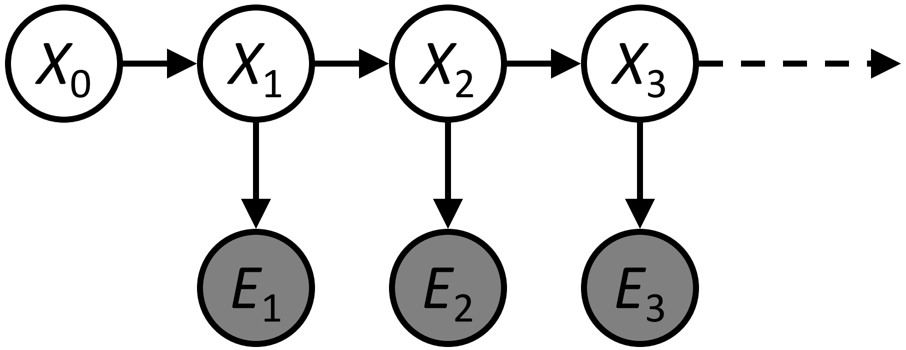
Particle Filtering



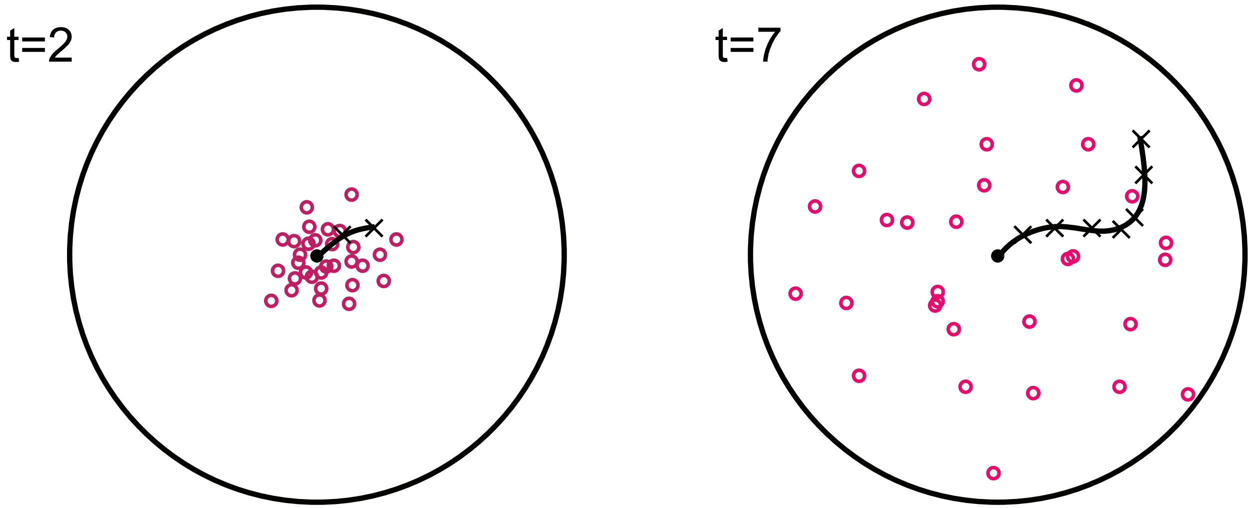
Approximate Inference on HMMs

- 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





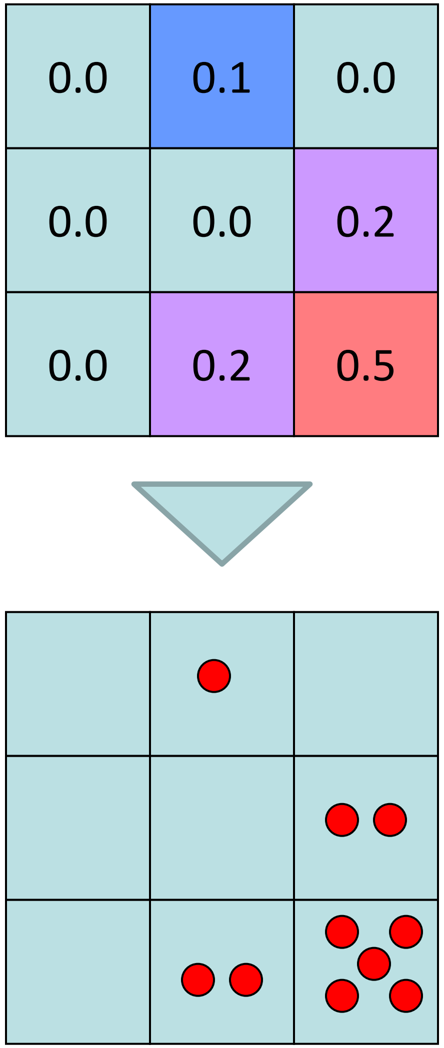
We need a new idea!



- 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

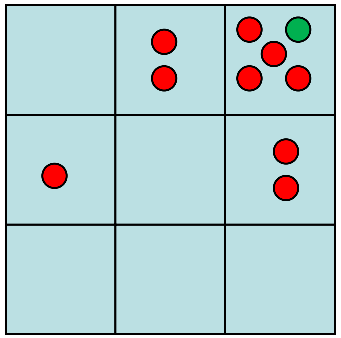
Particle Filtering

- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store $P(X \mid e_{1:T})$
- 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



Representation: Particles

- Our representation of P(X) is now a list of N particles (samples)
  - Generally, N << |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



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

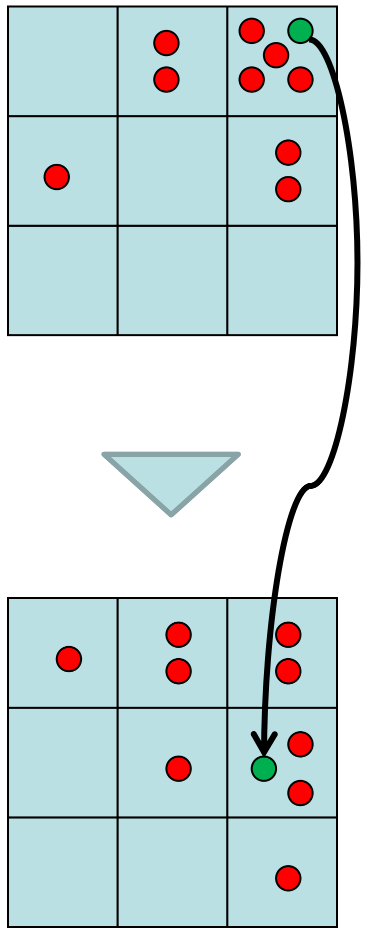
Particle Filtering: Elapse Time

- 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)

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



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

Particle Filtering: Incorporate Observation

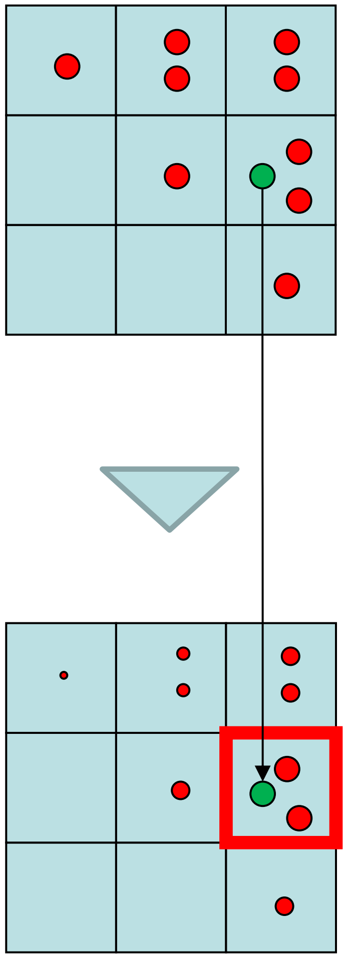
- After observing Evidence e_{t+1}:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, downweight samples based on the evidence

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

$$B(X) \propto P(e|X)B'(X)$$

  - As before, the probabilities don't sum to one, since all have been downweighted (in fact they now sum to (N times) an approximation of P(e))

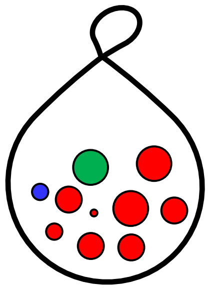
Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)



Particles:
(3,2) w=.9
(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

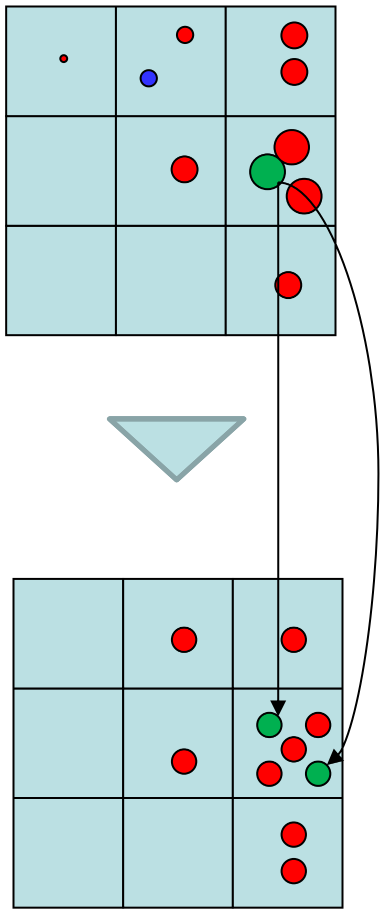
Particle Filtering: Resample

- 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



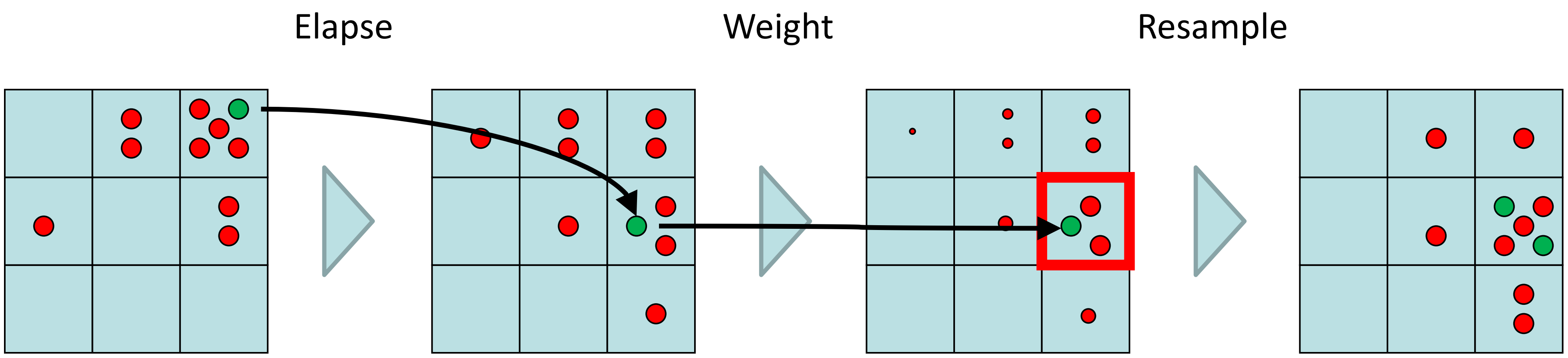
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



Recap: Particle Filtering

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



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

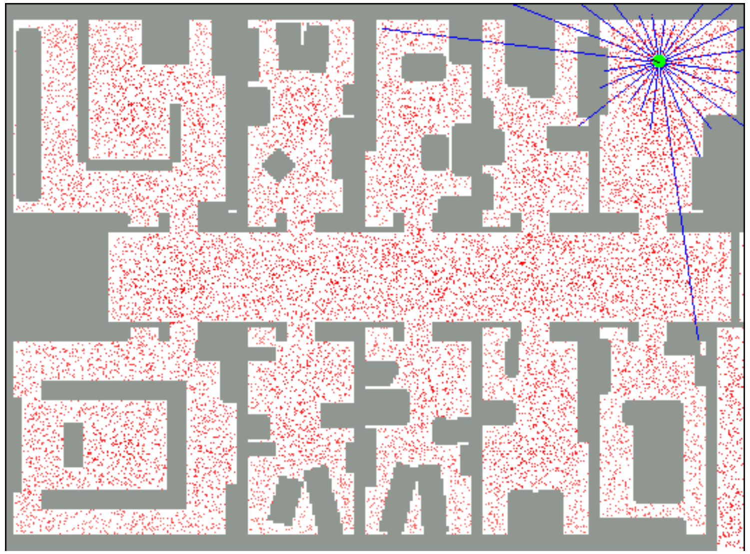
Video of Demo – Moderate Number of Particles

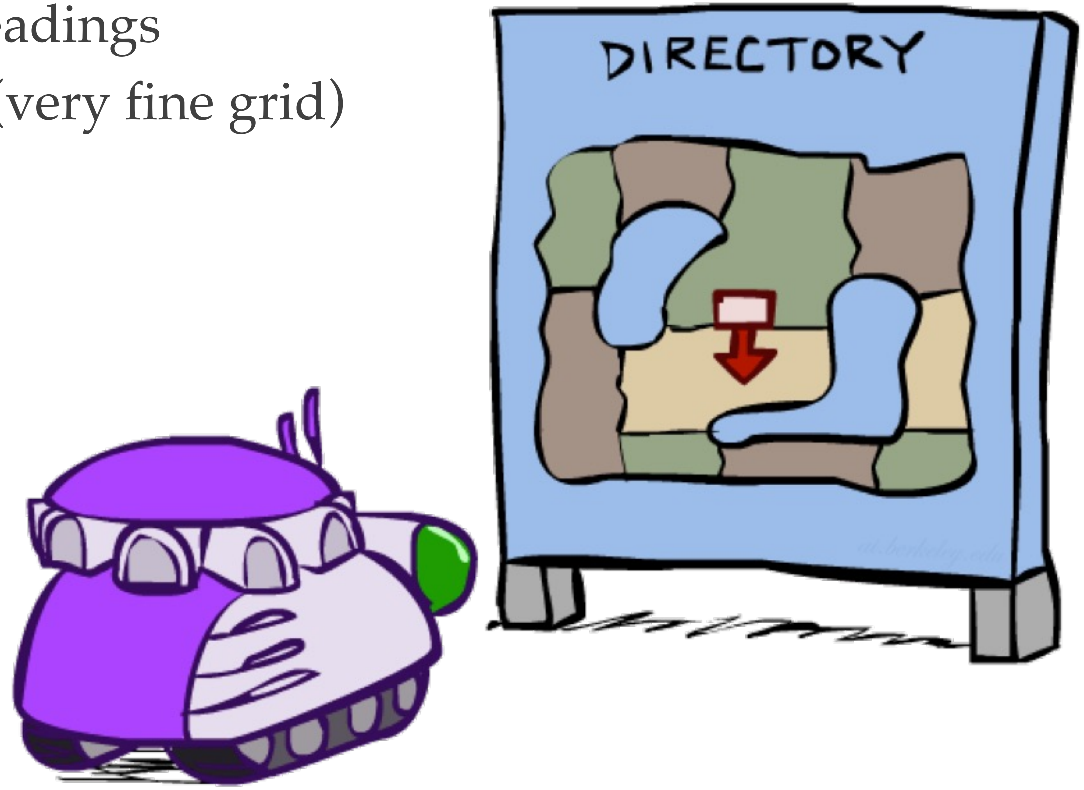
Video of Demo – One Particle

Video of Demo – Huge Number of Particles

Robot Localization

- 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



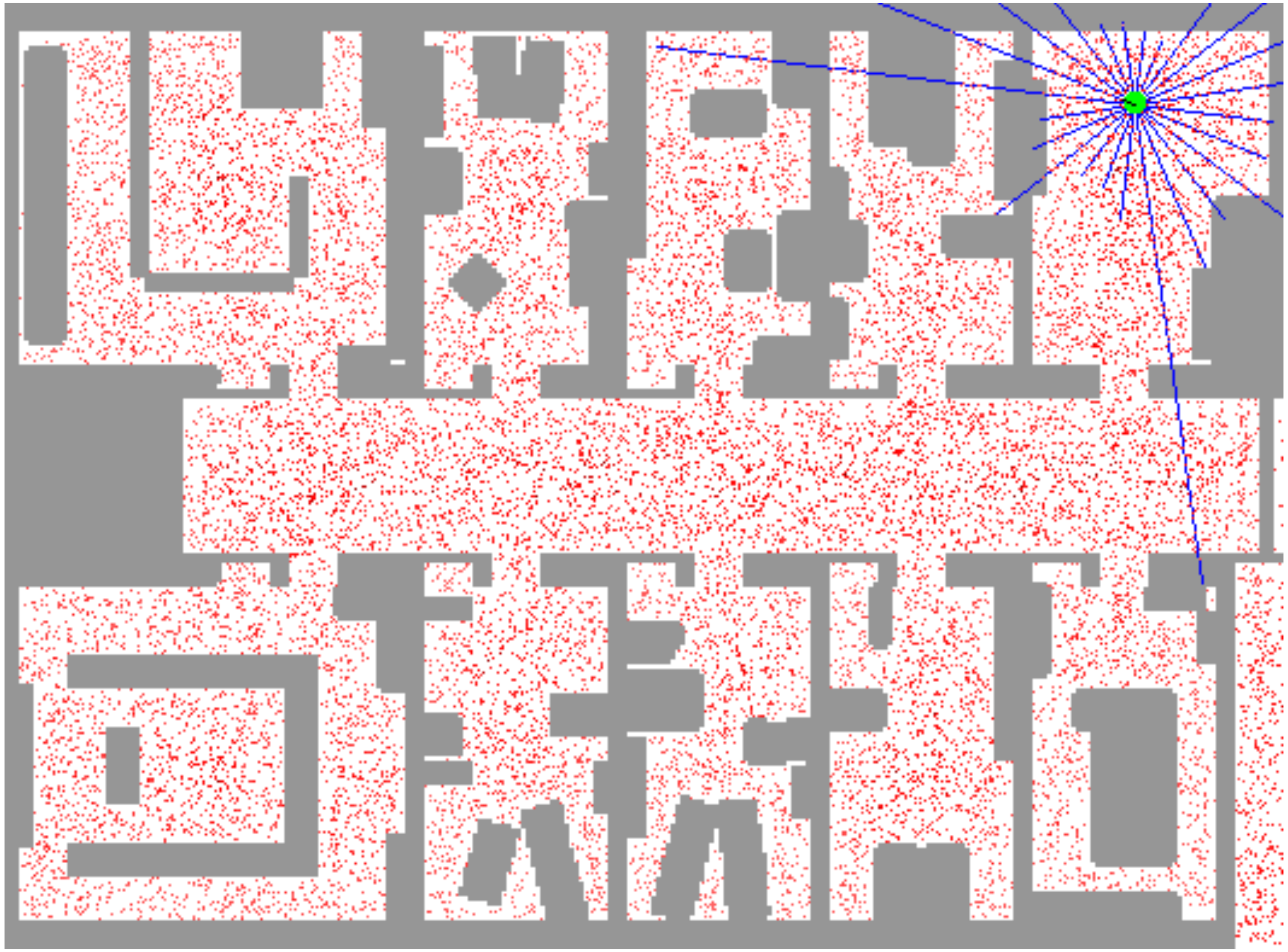


Particle Filter Localization (Sonar)

[Dieter Fox, et al.]

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

Particle Filter Localization (Laser)

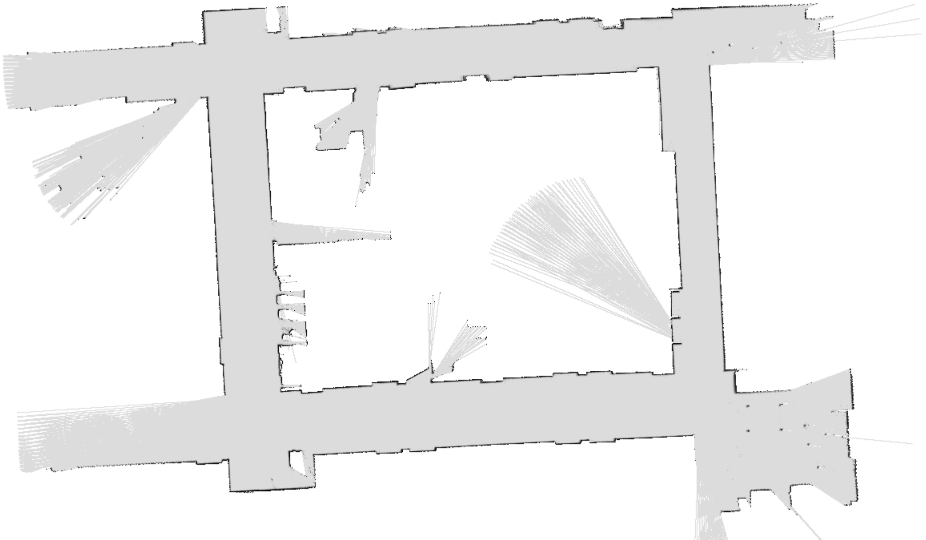


[Dieter Fox, et al.]

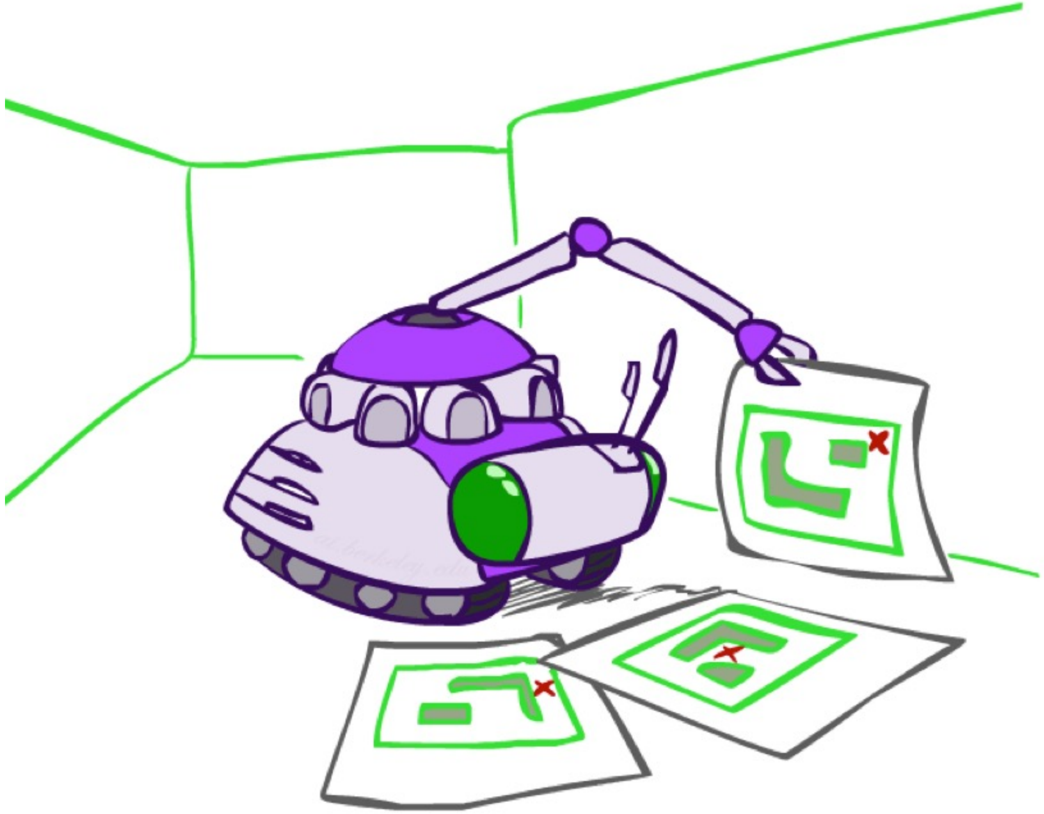
[Video: global-floor.gif]

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

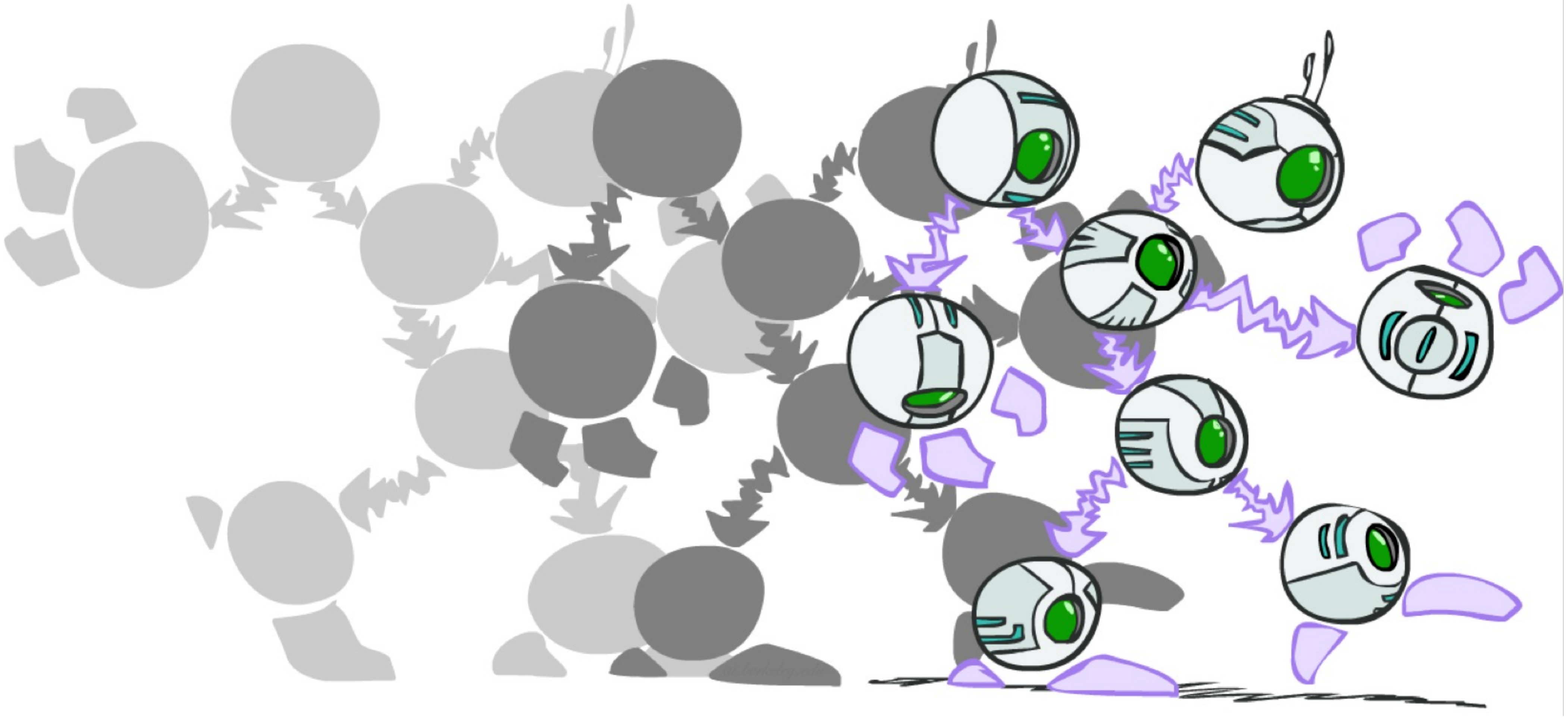


DP-SLAM, Ron Parr



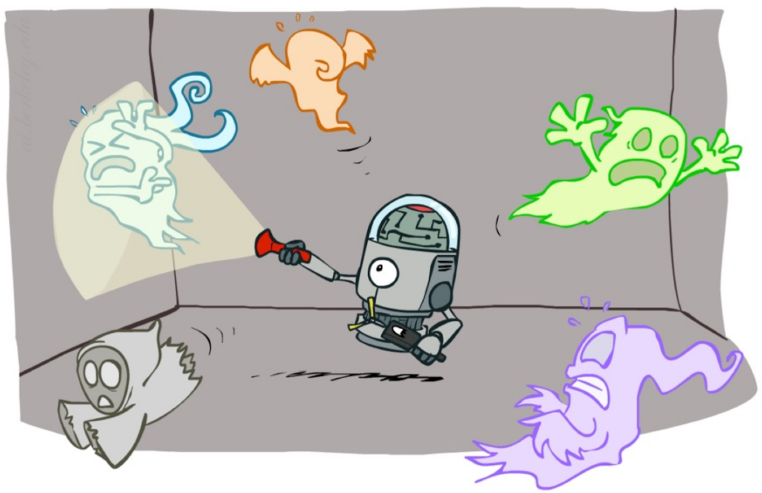
[Demo: PARTICLES-SLAM-mapping1-new.avi]

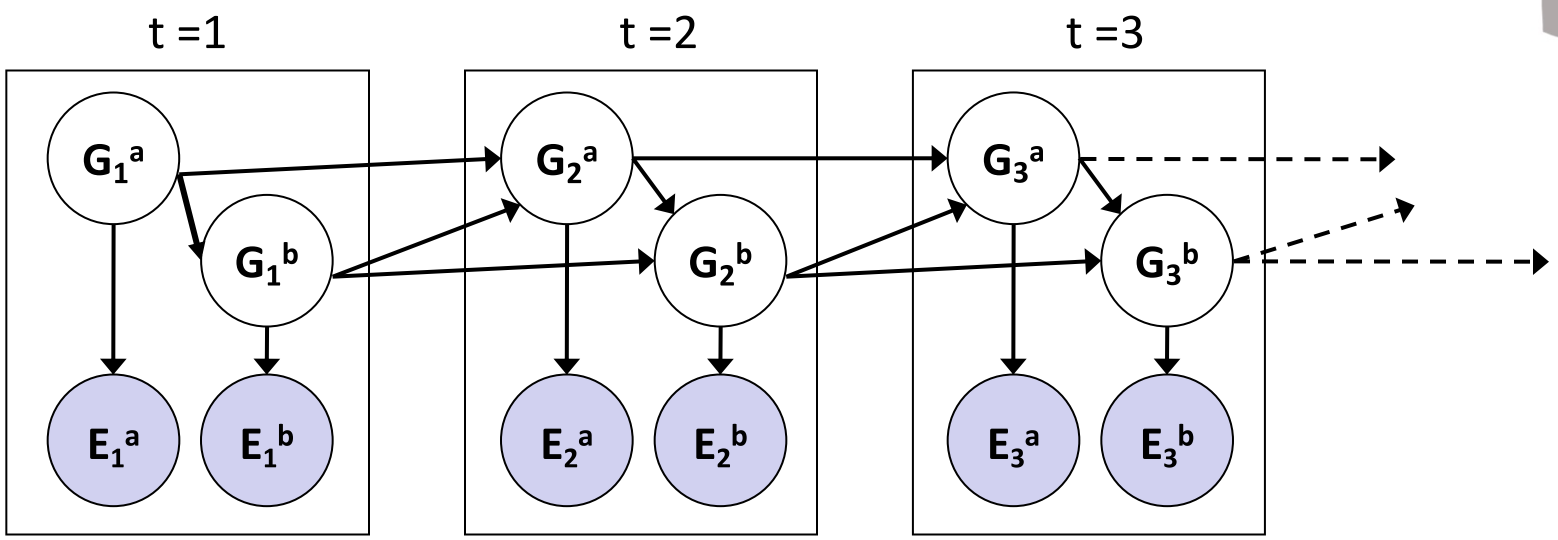
Dynamic Bayes Nets



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*

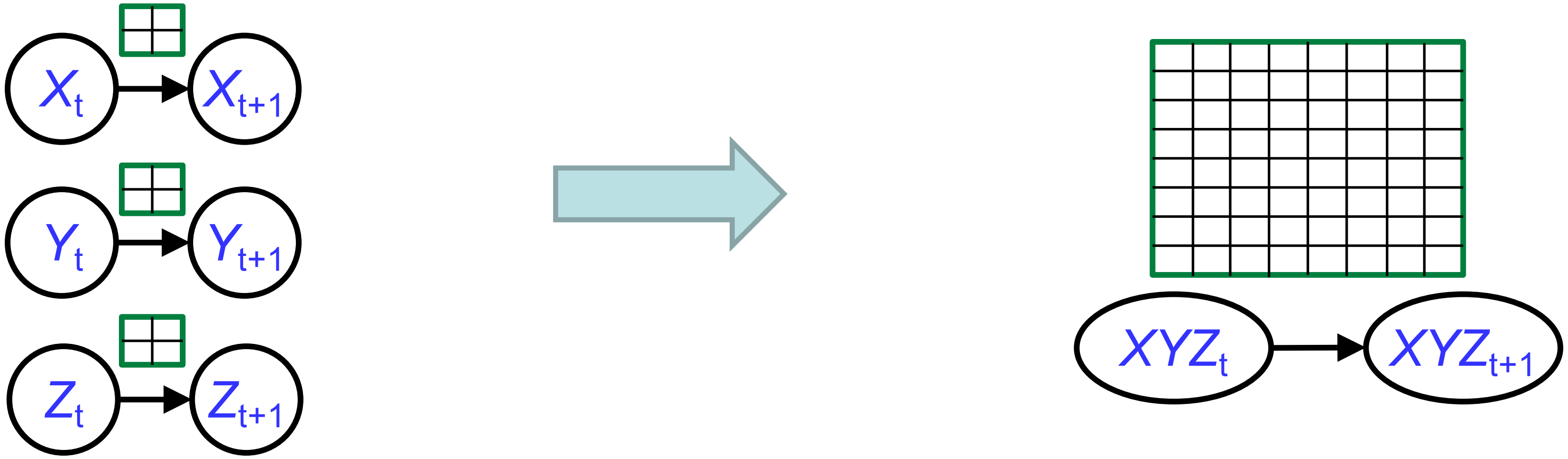




- Dynamic Bayes nets are a generalization of HMMs

DBNs and HMMs

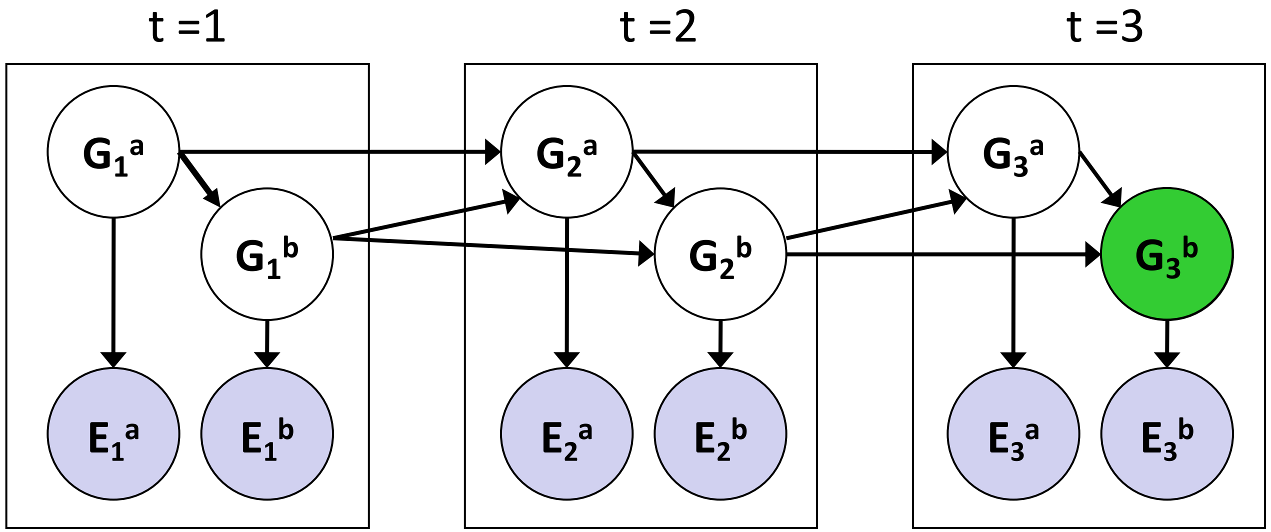
- Every HMM is a single-variable DBN
- Every discrete DBN is an HMM
  - HMM state is Cartesian product of DBN state variables



- 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 $2^{20} \times 2^{20} =\sim 10^{12}$ parameters

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Procedure: "unroll" the network for T time steps, then eliminate variables until $P(X_T \mid e_{1:T})$ 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: $\mathbf{G_1}^a = (3,3)$ $\mathbf{G_1}^b = (5,3)$
- **Elapse time**: Sample a successor for each particle
  - Example successor: $\mathbf{G_2}^a = (2,3)$ $\mathbf{G_2}^b = (6,3)$
- **Observe**: Weight each *entire* sample by the likelihood of the evidence conditioned on the sample
  - Likelihood: $P(\mathbf{E_1}^a \mid \mathbf{G_1}^a) * P(\mathbf{E_1}^b \mid \mathbf{G_1}^b)$
- **Resample:** Select prior samples (tuples of values) in proportion to their likelihood