

Announcements

- P4 Bayes' Nets / Treasure Hunting has been released.
 - It's due Monday April 4th at 11:59pm.
- HW7 has been released.
 - It's due Tuesday April 5th, at 11:59pm. It's about twice as long as a typical homework. We encourage you to get started early!
- Regrade request window for the Midterm
 - Monday through Wednesday this week.

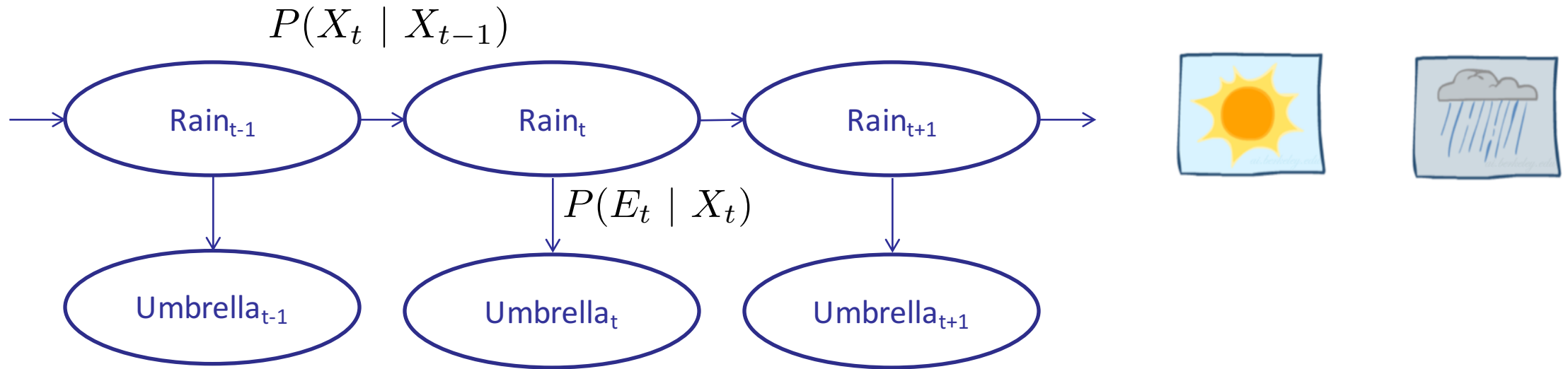
CS 188: Artificial Intelligence

Hidden Markov Models -- Filters



Instructors: Pieter Abbeel & Anca Dragan --- University of California, Berkeley

Example: Weather HMM



- An HMM is defined by:

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

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

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

Video of Demo Ghostbusters – Circular Dynamics -- HMM

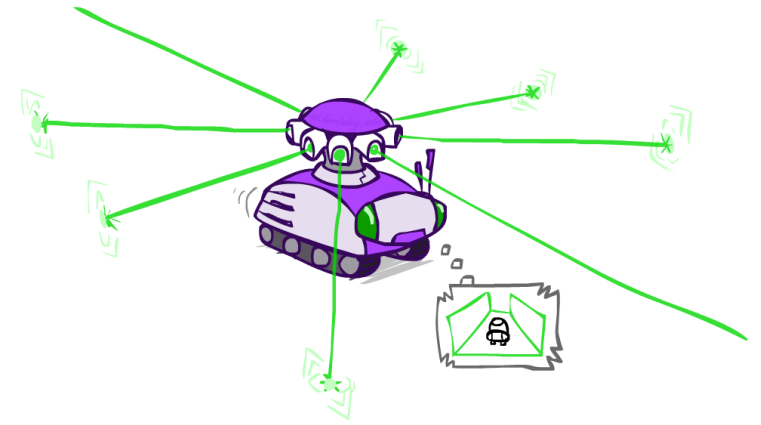
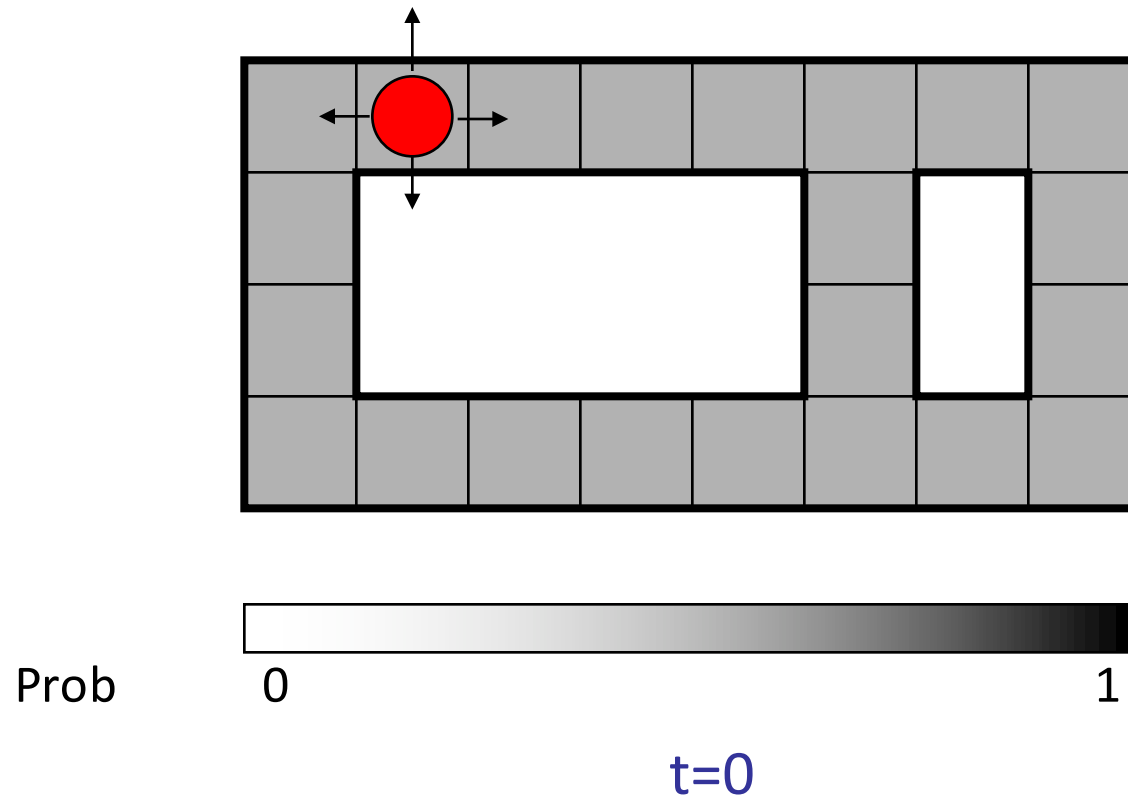


Filtering / Monitoring

- Filtering, or monitoring, is the task of tracking the distribution $B_t(X) = P_t(X_t \mid e_1, \dots, e_t)$ (the belief state) over time
- We start with $B_1(X)$ in an initial setting, usually uniform
- As time passes, or we get observations, we update $B(X)$
- The Kalman filter was invented in the 60's and first implemented as a method of trajectory estimation for the Apollo program

Example: Robot Localization

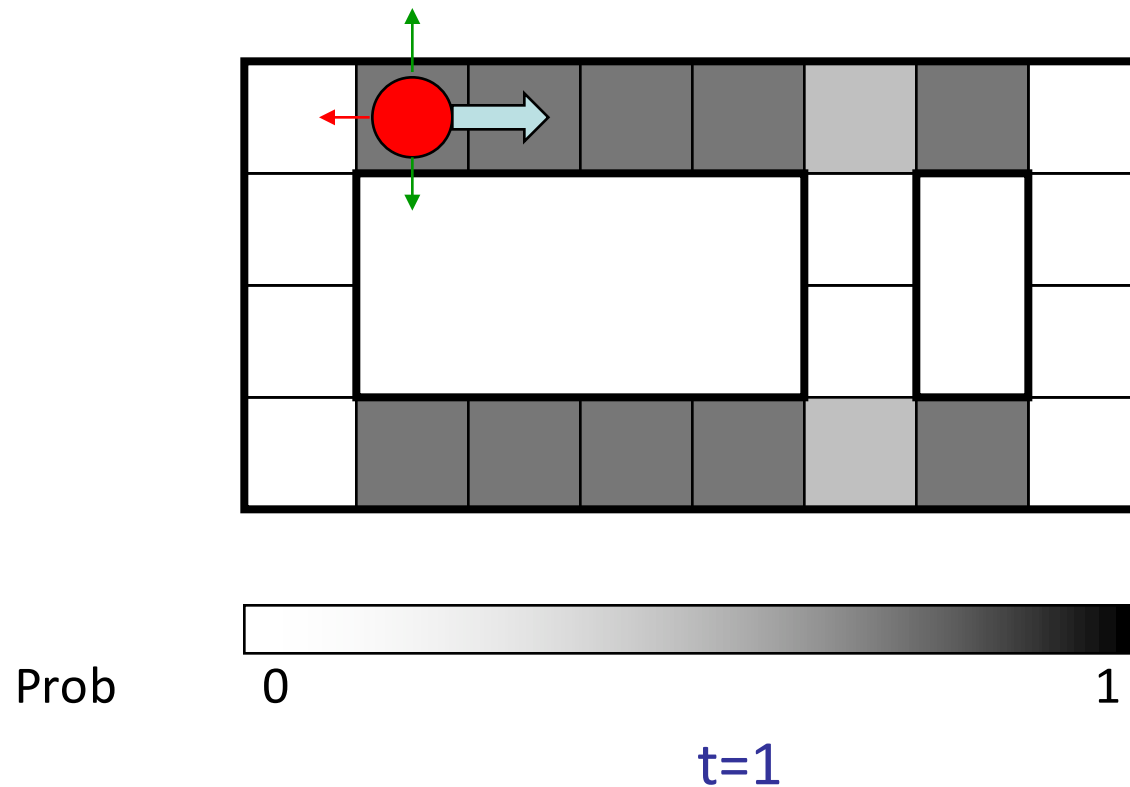
*Example from
Michael Pfeiffer*



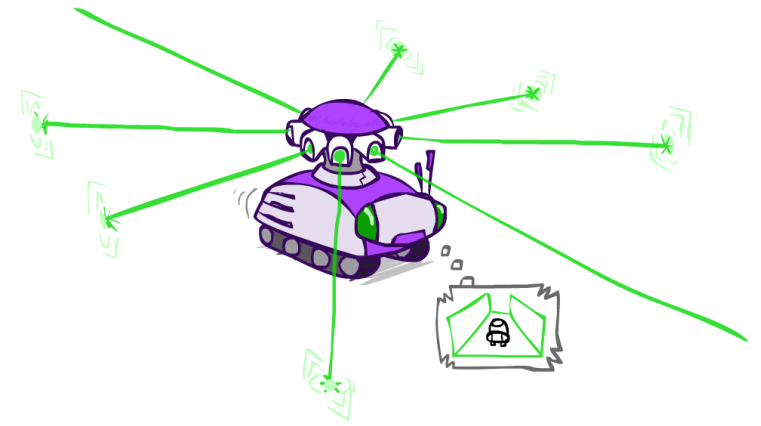
Sensor model: can read in which directions there is a wall,
never more than 1 mistake

Motion model: may not execute action with small prob.

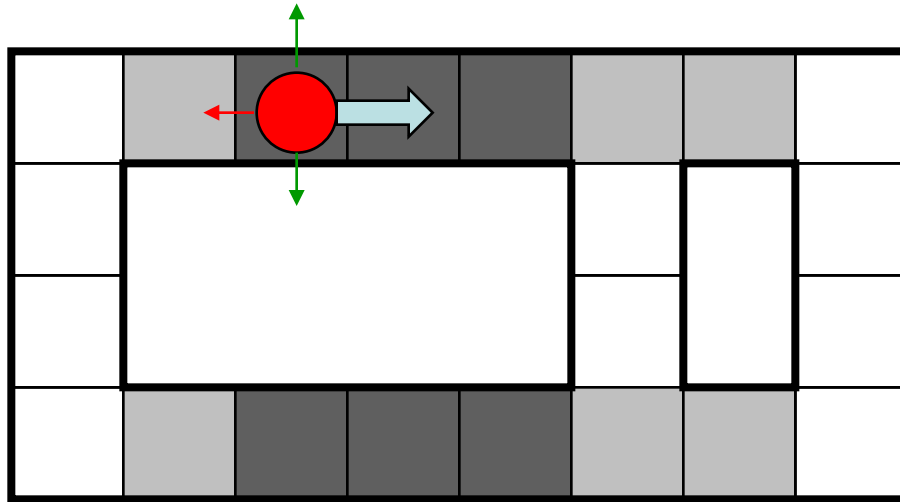
Example: Robot Localization



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



Example: Robot Localization

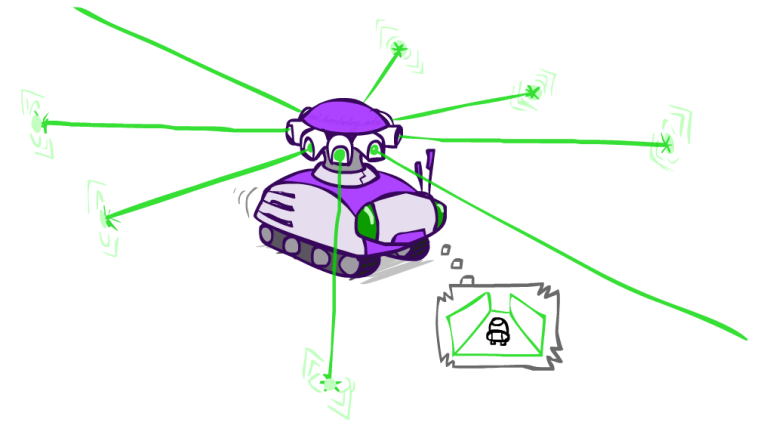


Prob

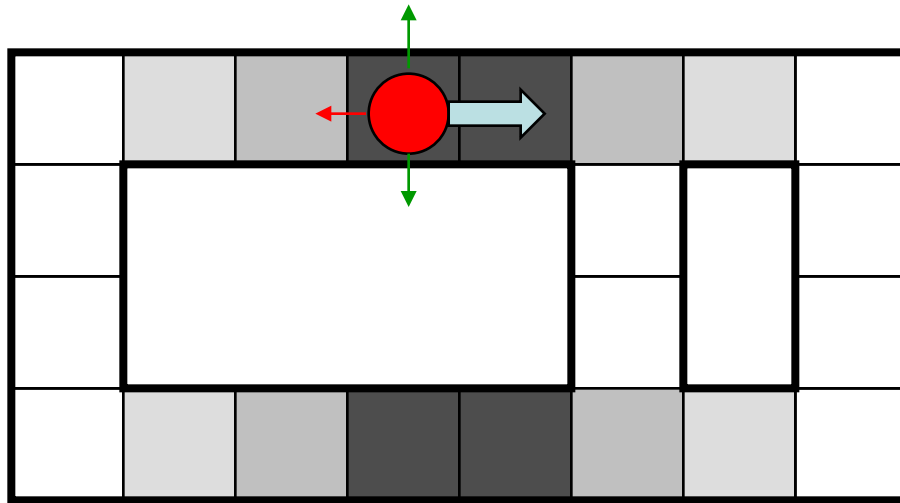
0

1

t=2



Example: Robot Localization

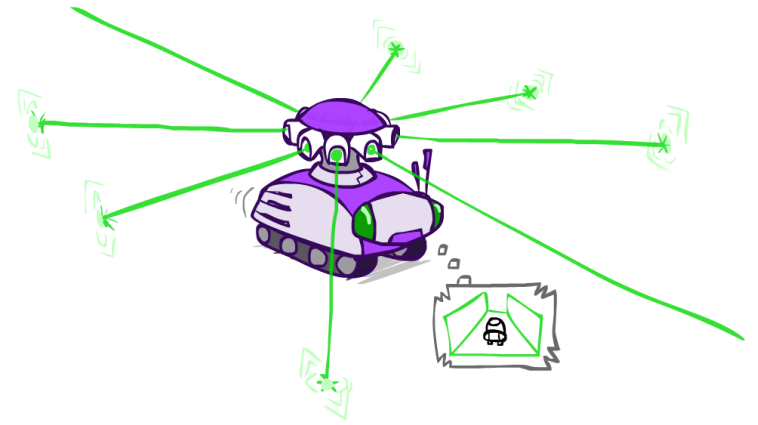


Prob

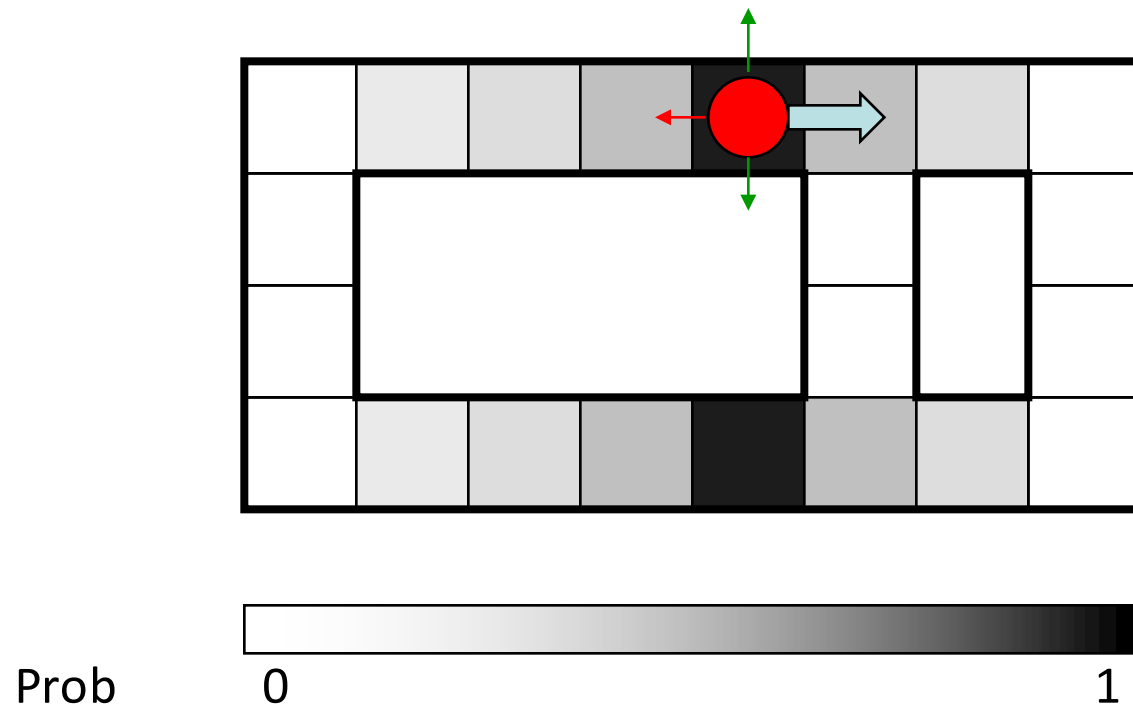
0

1

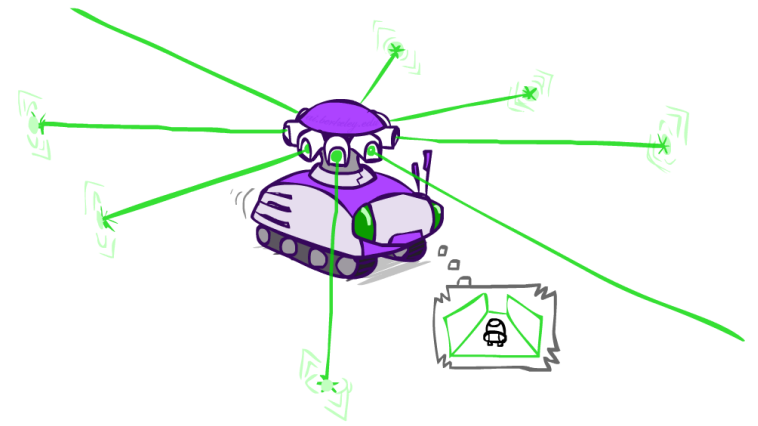
$t=3$



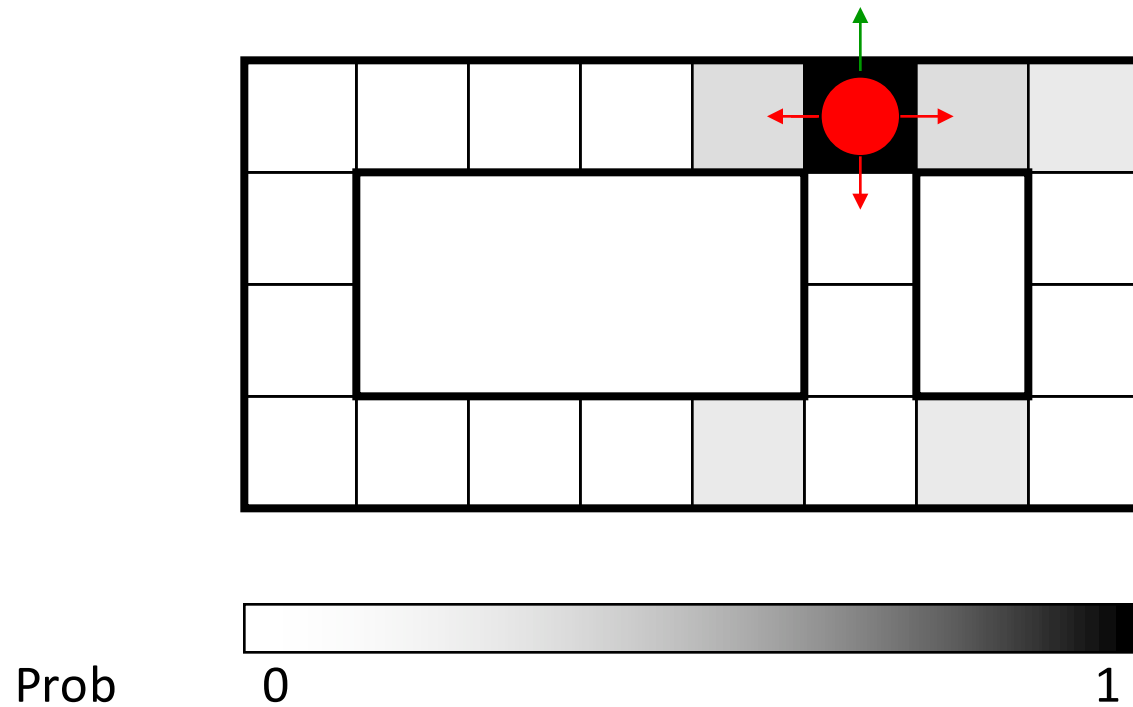
Example: Robot Localization



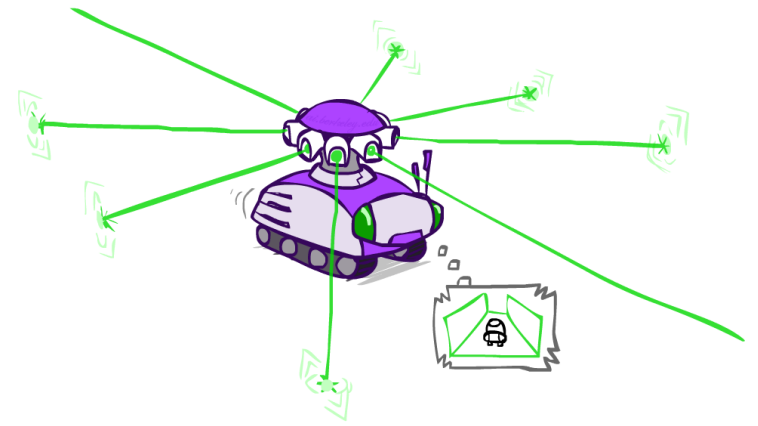
$t=4$



Example: Robot Localization



$t=5$



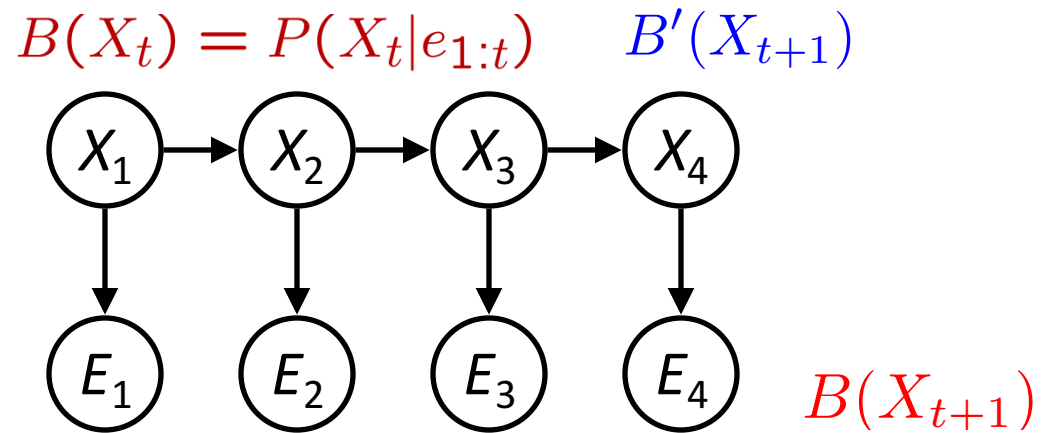
Inference: Find State Given Evidence

- We are given evidence at each time and want to know

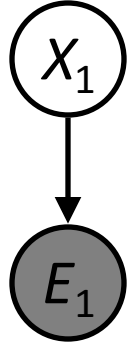
$$B_t(X) = P(X_t | e_{1:t})$$

- Idea: start with $P(X_1)$ and derive B_t in terms of B_{t-1}
 - equivalently, derive B_{t+1} in terms of B_t

Two Steps: Passage of Time + Observation



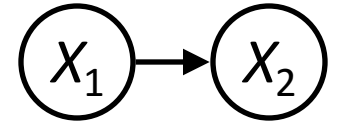
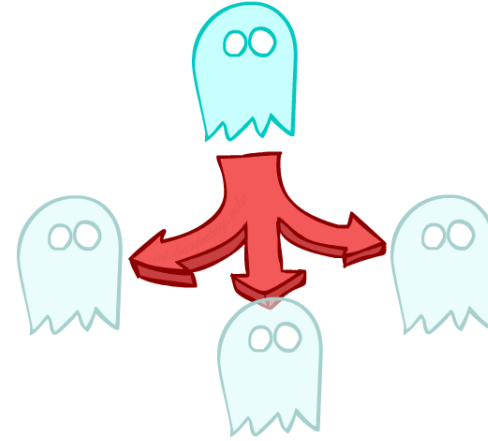
Inference: Base Cases



$$P(X_1|e_1)$$

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

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



$$P(X_2)$$

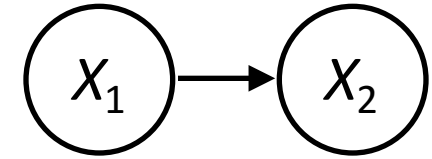
$$P(X_2) = \sum_{x_1} P(x_1, X_2)$$

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

Passage of Time

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

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



- Then, after one time step passes:

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

- 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 t the belief is about, and what evidence it includes

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

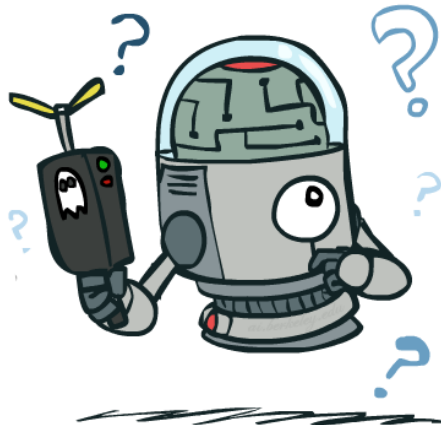
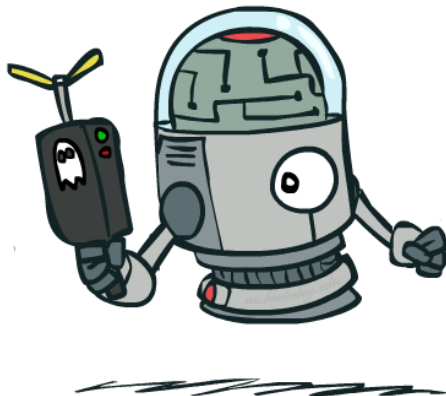
T = 1

<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



Observation

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

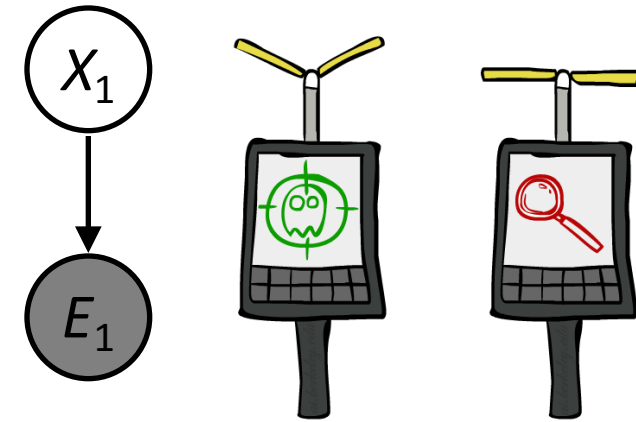
$$B'(X_{t+1}) = P(X_{t+1} | e_{1:t})$$

- Then, after evidence comes in:

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

- Or, compactly:

$$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1} | X_{t+1}) B'(X_{t+1})$$



- Basic idea: beliefs “reweighted” by likelihood of evidence
- Unlike passage of time, we have to renormalize

Example: Observation

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

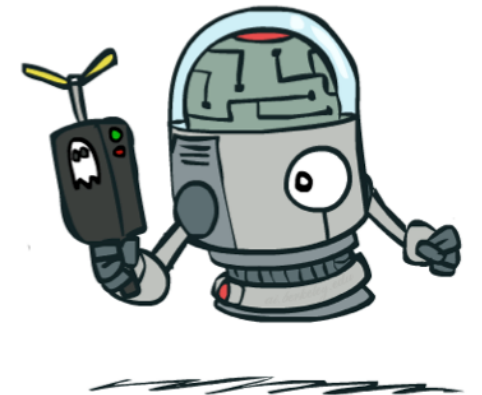
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

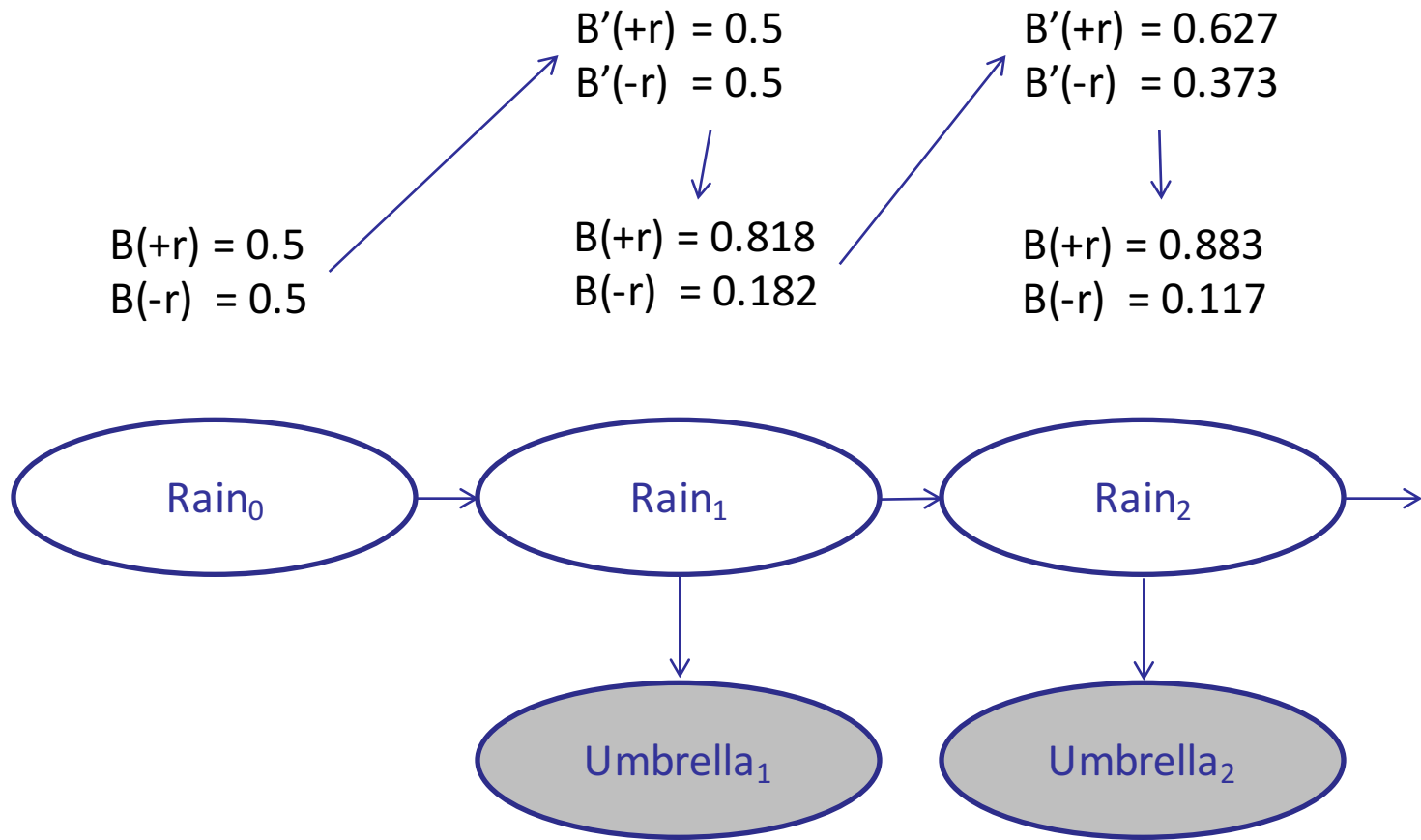
<0.01	<0.01	<0.01	<0.01	0.02	<0.01
<0.01	<0.01	<0.01	0.83	0.02	<0.01
<0.01	<0.01	0.11	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

After observation

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



Example: Weather HMM



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

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

Online Belief Updates

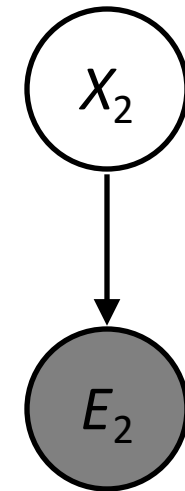
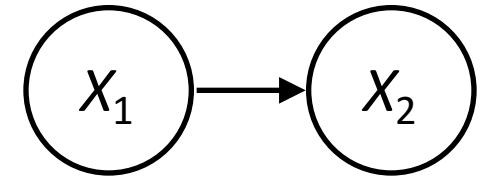
- Every time step, we start with current $P(X \mid \text{evidence})$
- We update for time:

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

- 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

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

- We can derive the following updates

$$\begin{aligned} P(x_t|e_{1:t}) &\propto_{X_t} P(x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\ &= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t) \\ &= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1}) \end{aligned}$$

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

Pacman – Sonar

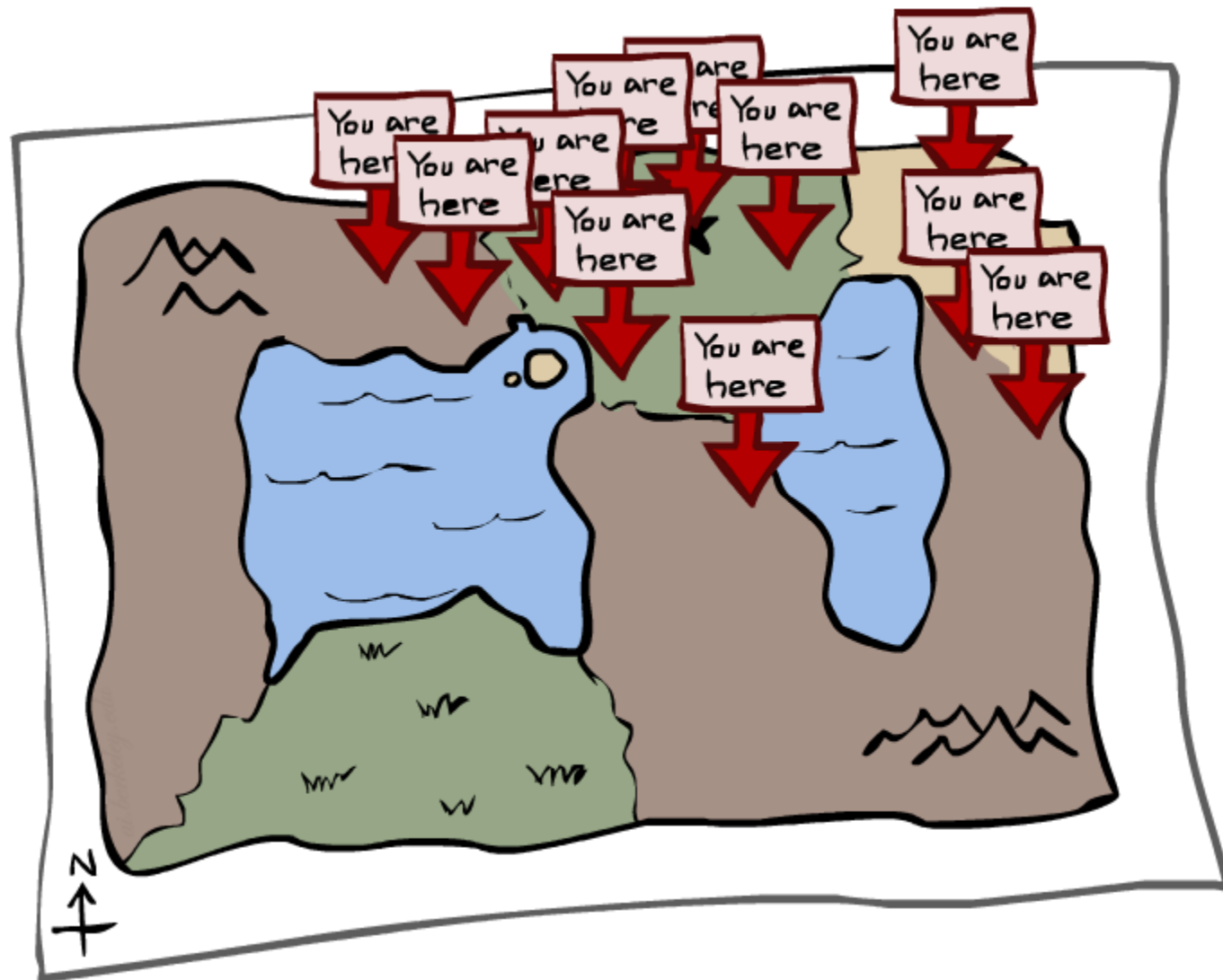


[Demo: Pacman – Sonar – No Beliefs(L14D1)]

Video of Demo Pacman – Sonar (with beliefs)



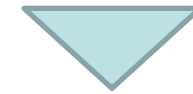
Particle Filtering



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

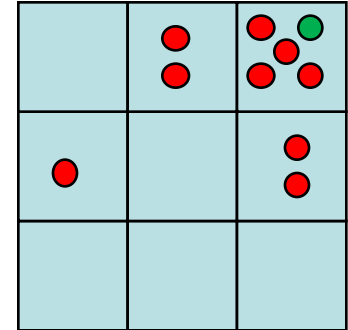
0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5



	●	
		● ●
	● ●	● ● ● ●

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



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

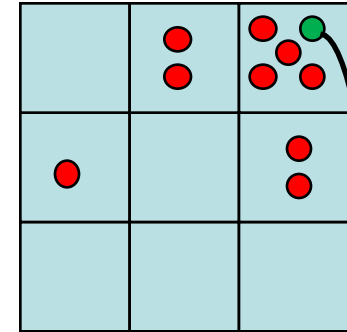
- Each particle is moved by sampling its next position from the transition model

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

- This is like prior sampling – 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)

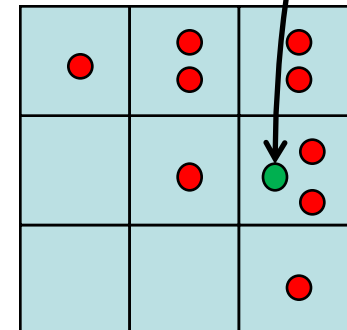
Particles:

(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: Observe

- Slightly trickier:

- Don't sample observation, fix it
- Similar to likelihood weighting, downweight samples based on the evidence

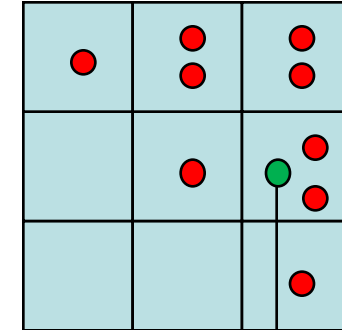
$$w(x) = P(e|x)$$

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

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

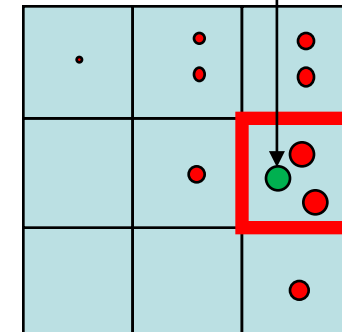
Particles:

(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

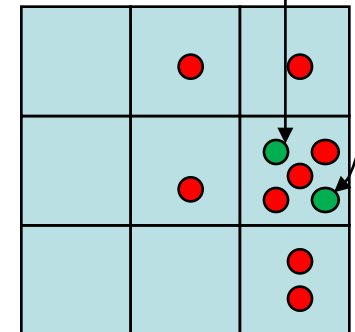
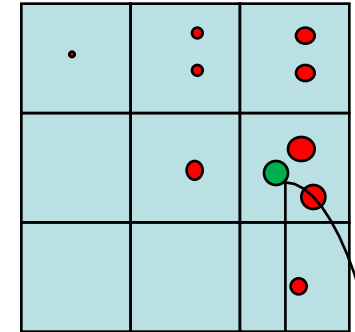
- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This is equivalent to renormalizing the distribution
- Now the update is complete for this time step, continue with the next one

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

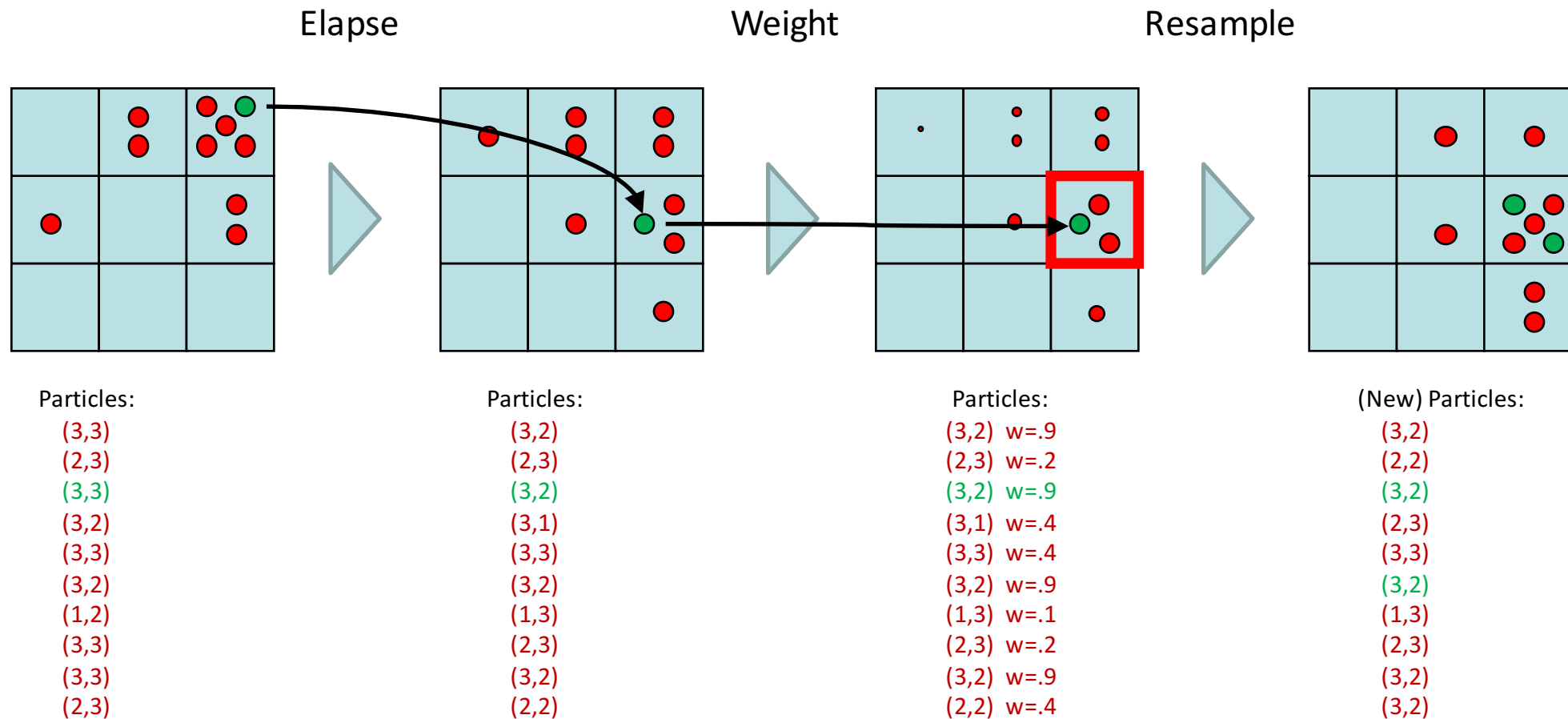
(New) Particles:

(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



Video of Demo – Moderate Number of Particles



Video of Demo – One Particle

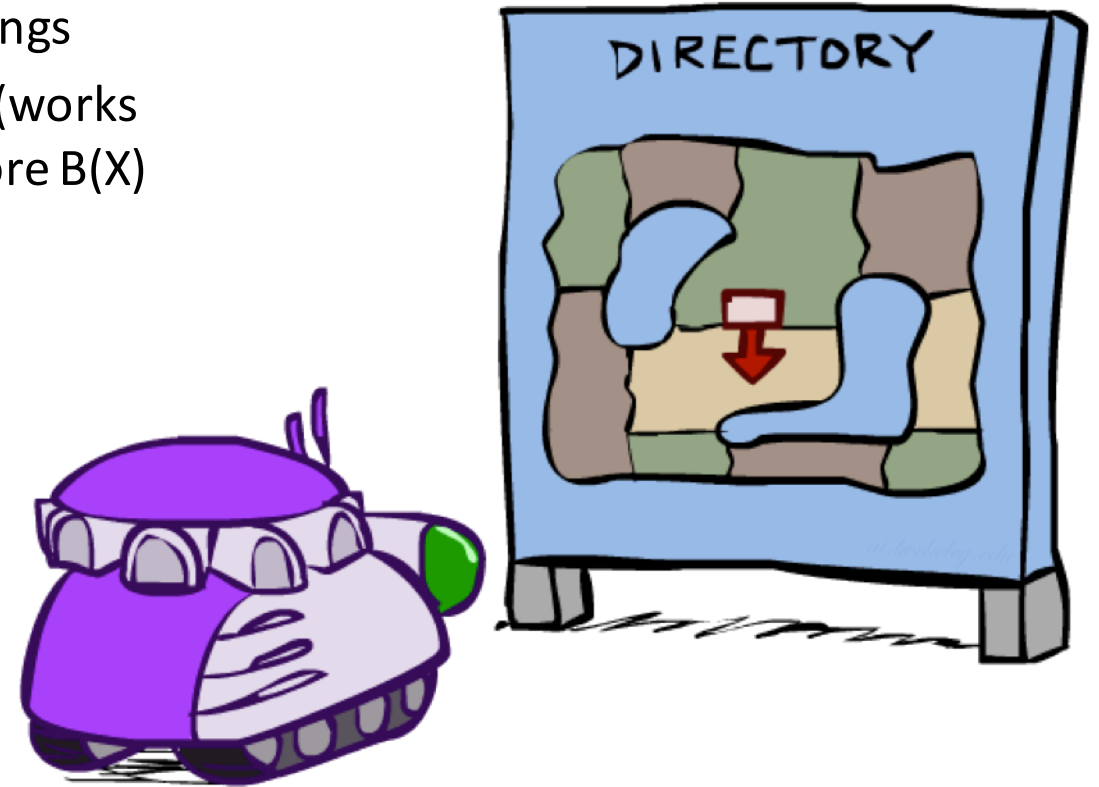
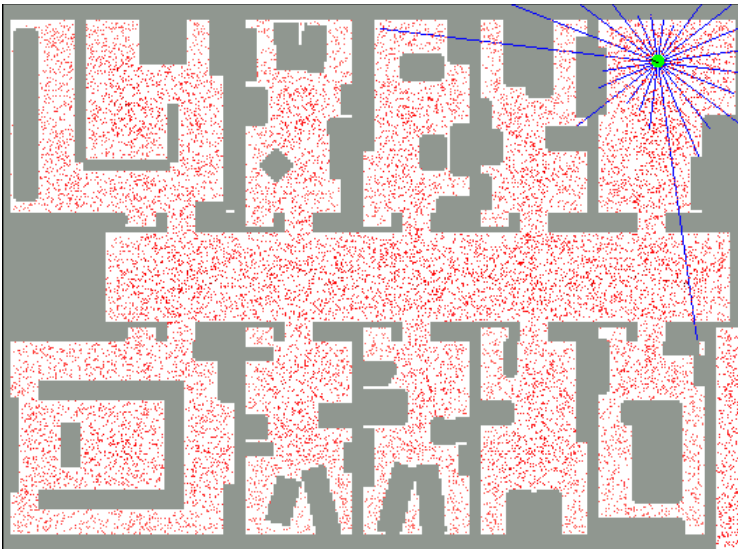


Video of Demo – Huge Number of Particles



Robot Localization

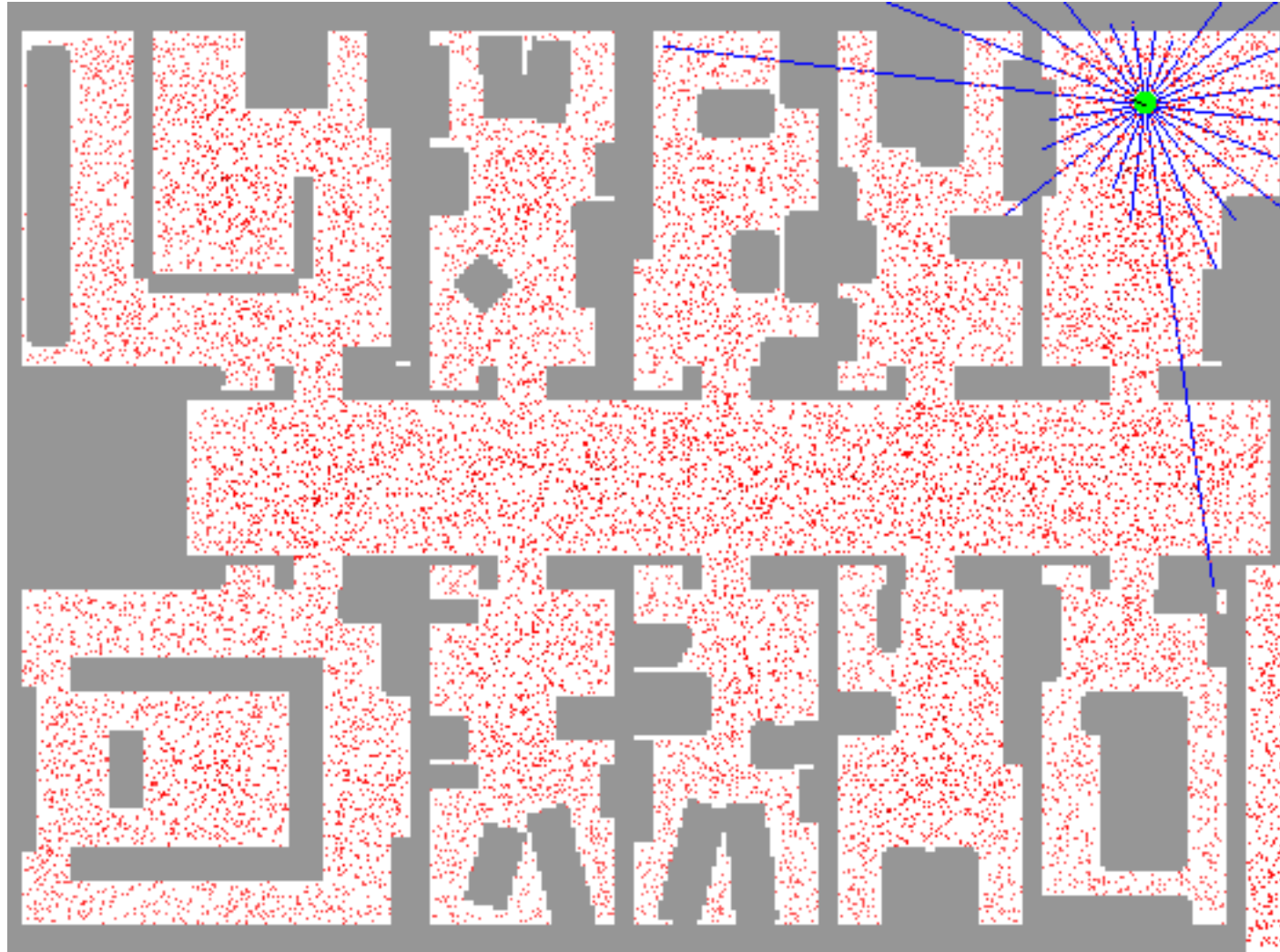
- In robot localization:
 - We know the map, but not the robot's position
 - Observations may be vectors of range finder readings
 - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store $B(X)$
 - Particle filtering is a main technique



Particle Filter Localization (Sonar)

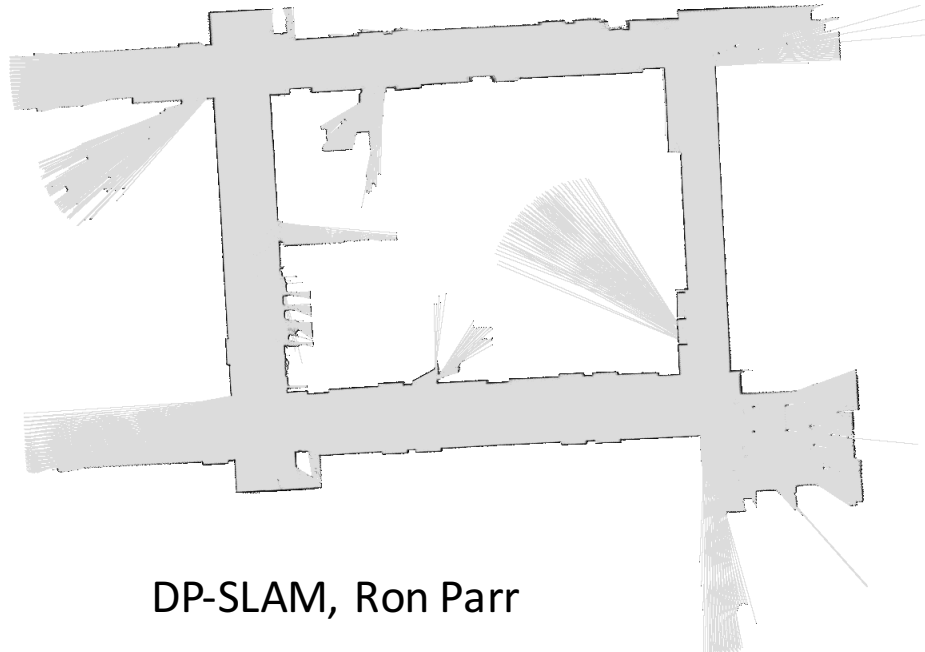


Particle Filter Localization (Laser)

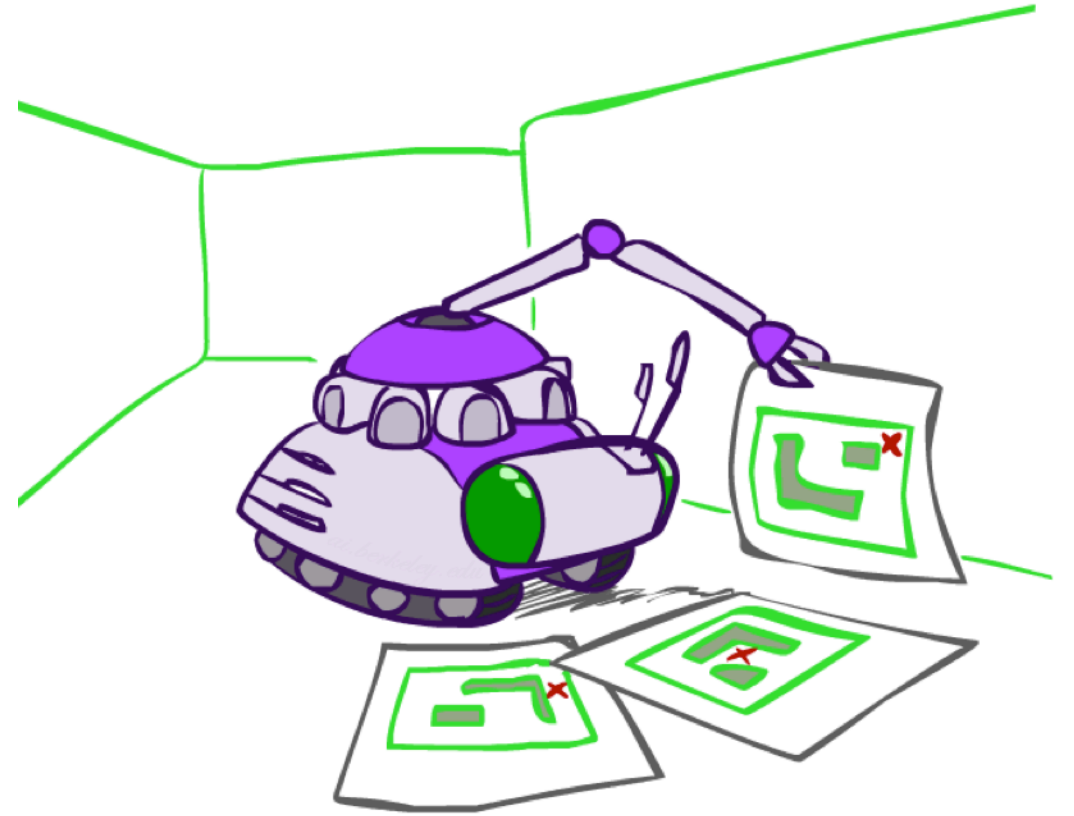


Robot Mapping

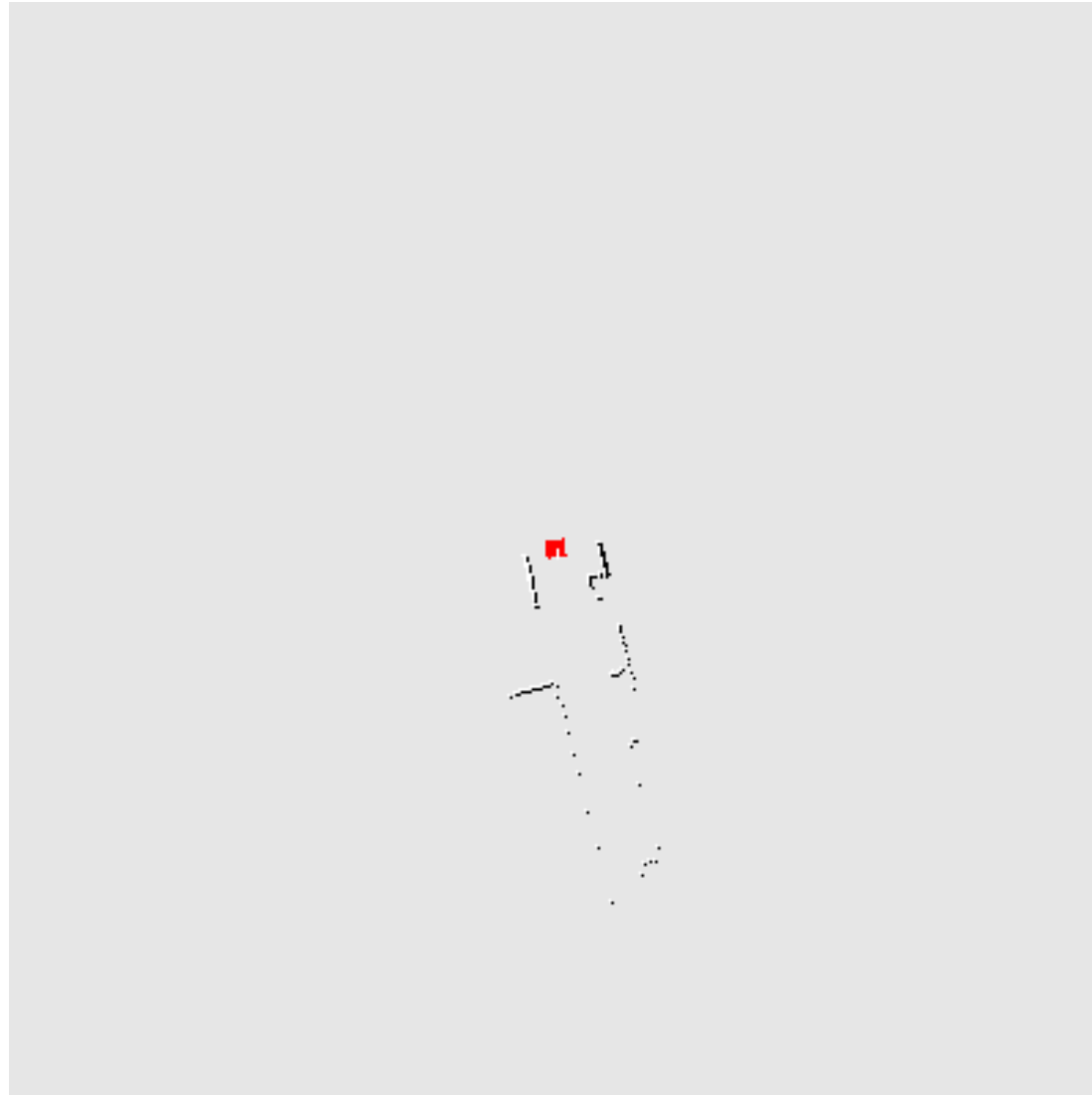
- SLAM: Simultaneous Localization And Mapping
 - We do not know the map or our location
 - State consists of position AND map!
 - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods



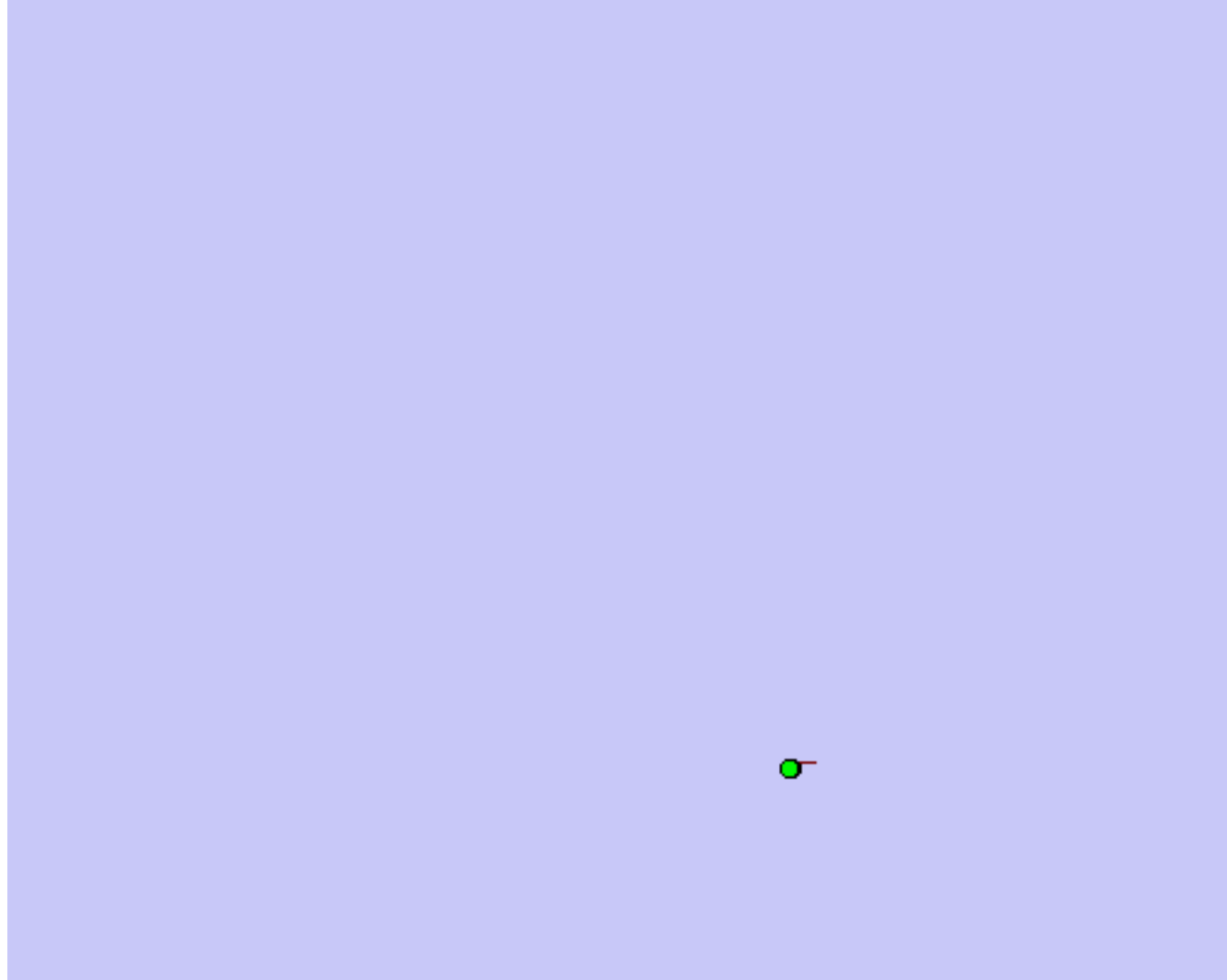
DP-SLAM, Ron Parr



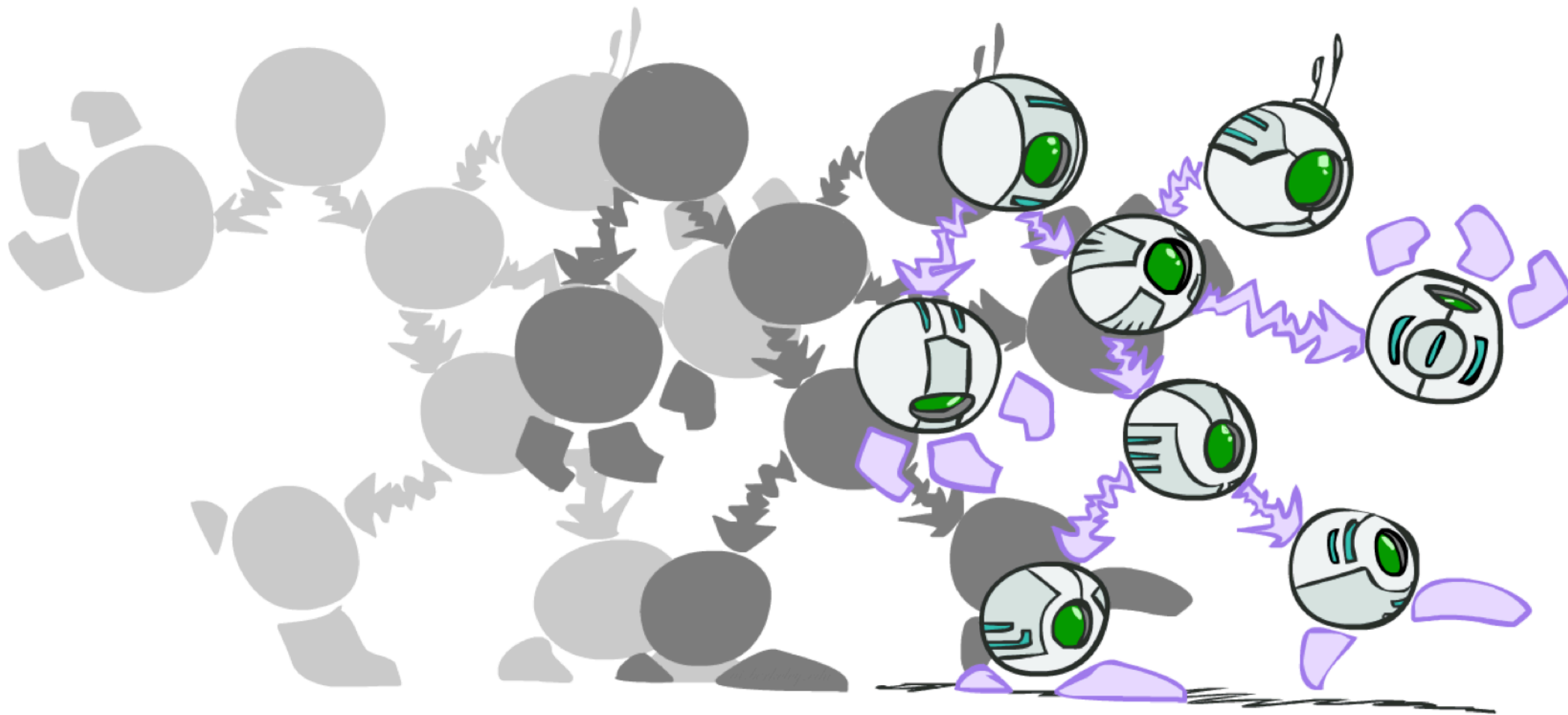
Particle Filter SLAM – Video 1



Particle Filter SLAM – Video 2

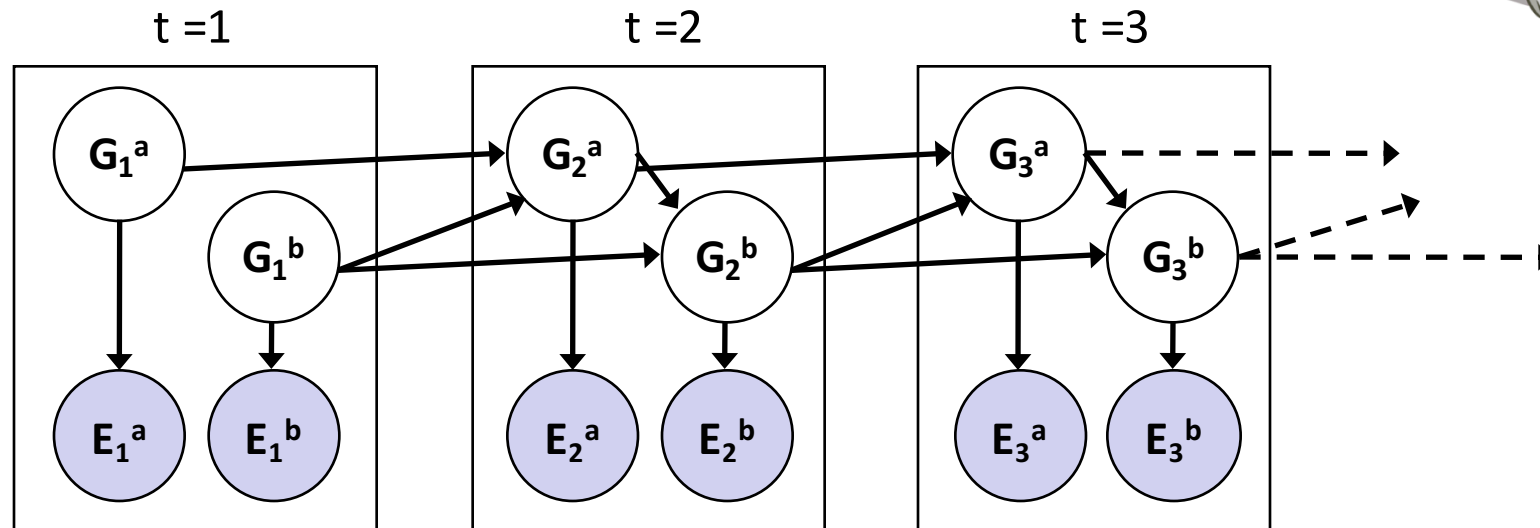
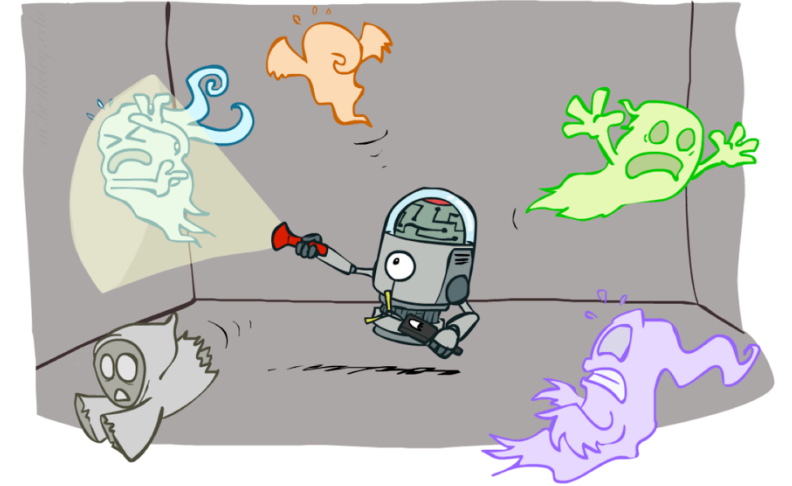


Dynamic Bayes Nets



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

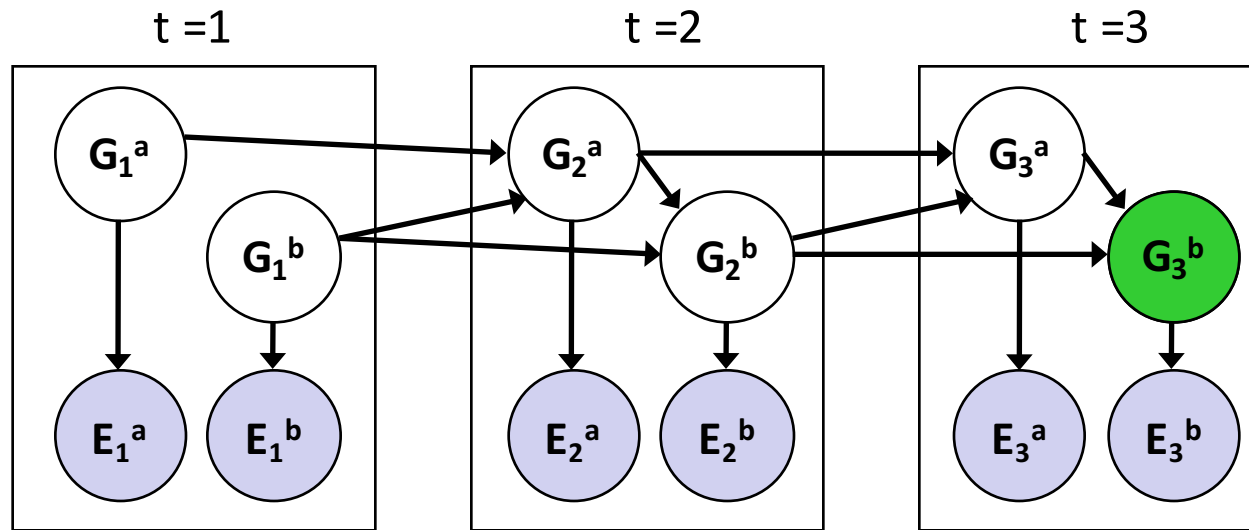
[Demo: pacman sonar ghost DBN model (L15D6)]

Video of Demo Pacman Sonar Ghost DBN Model



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: $\mathbf{G}_1^a = (3,3)$ $\mathbf{G}_1^b = (5,3)$
- **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: $P(\mathbf{E}_1^a | \mathbf{G}_1^a) * P(\mathbf{E}_1^b | \mathbf{G}_1^b)$
- **Resample:** Select prior samples (tuples of values) in proportion to their likelihood

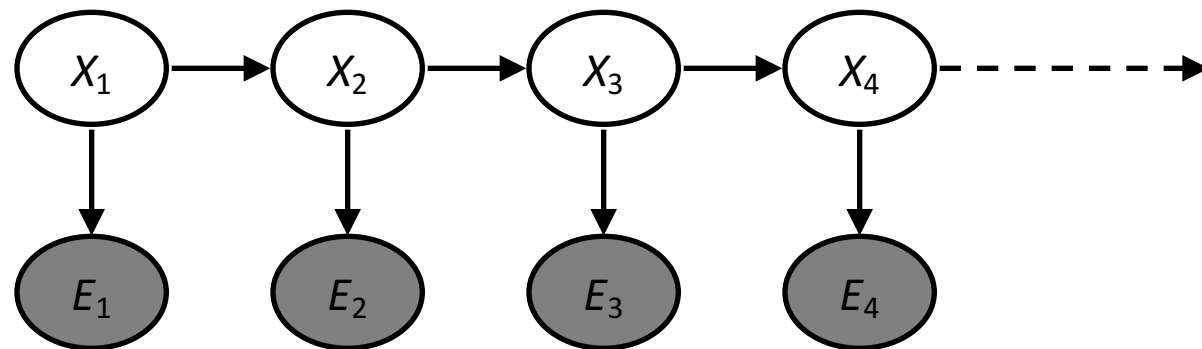
Most Likely Explanation



HMMs: MLE Queries

- HMMs defined by

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



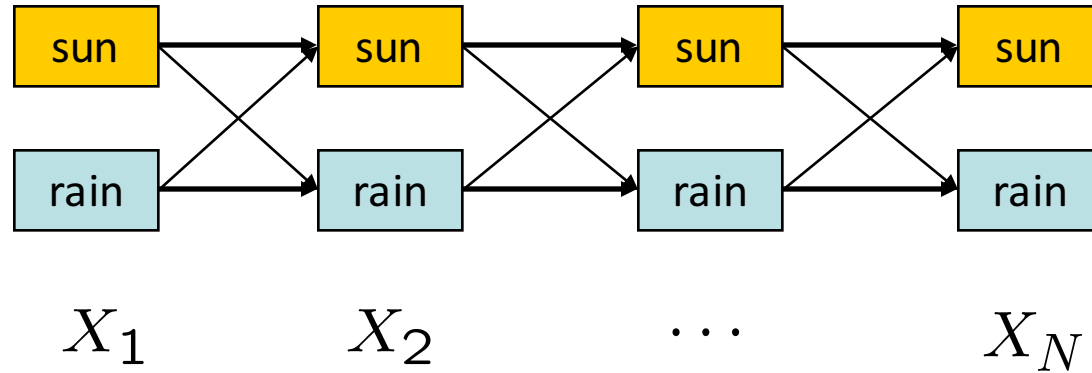
- New query: most likely explanation:

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

- New method: the Viterbi algorithm

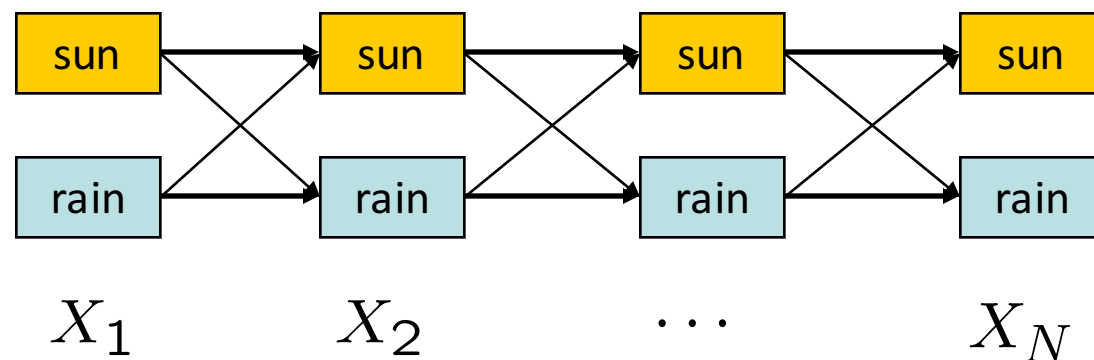
State Trellis

- 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)

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

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

Viterbi Algorithm (Max)

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

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