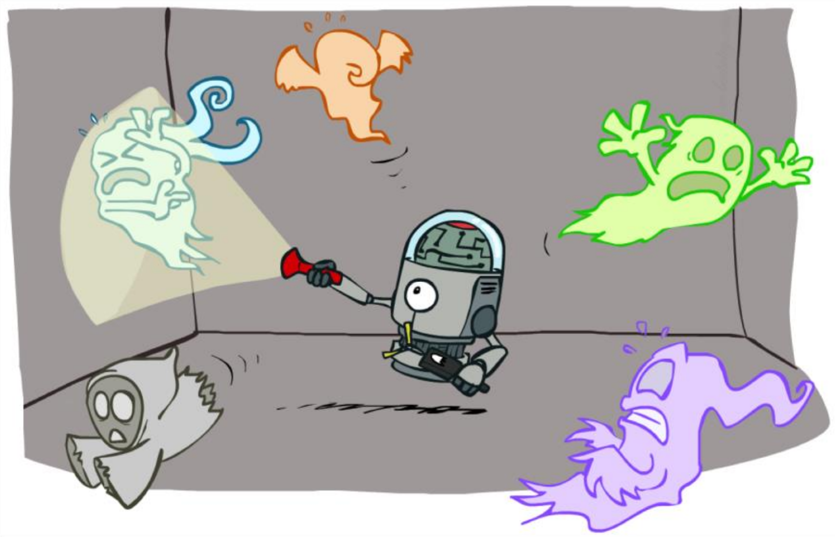
CSE 573 PMP: Artificial Intelligence

Hanna Hajishirzi

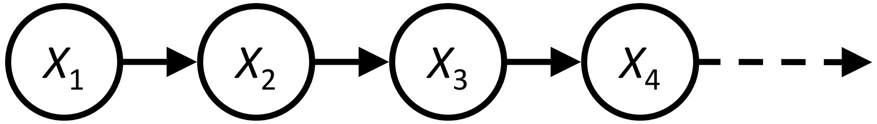
HMMs Inference, Particle Filters



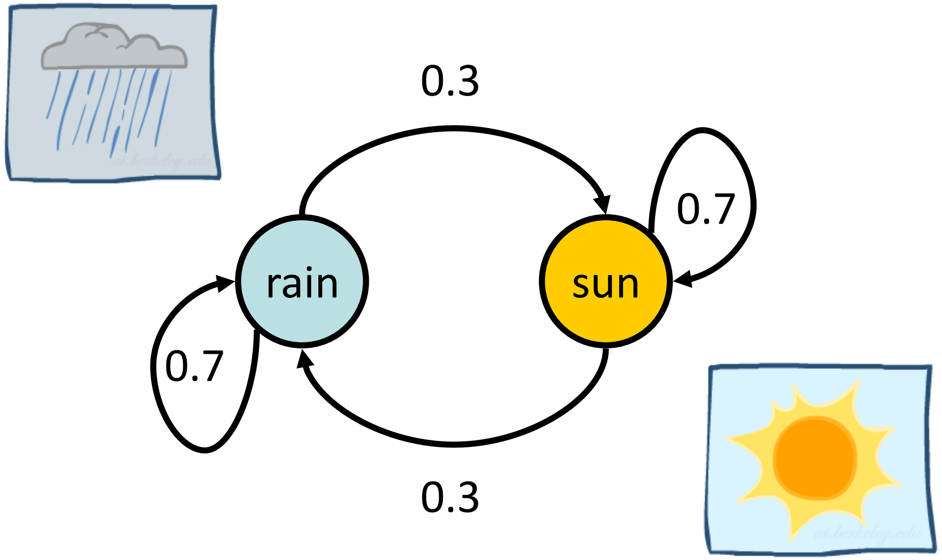
slides adapted from
Dan Klein, Pieter Abbeel ai.berkeley.edu
And Dan Weld, Luke Zettelmoyer

Recap: Reasoning Over Time

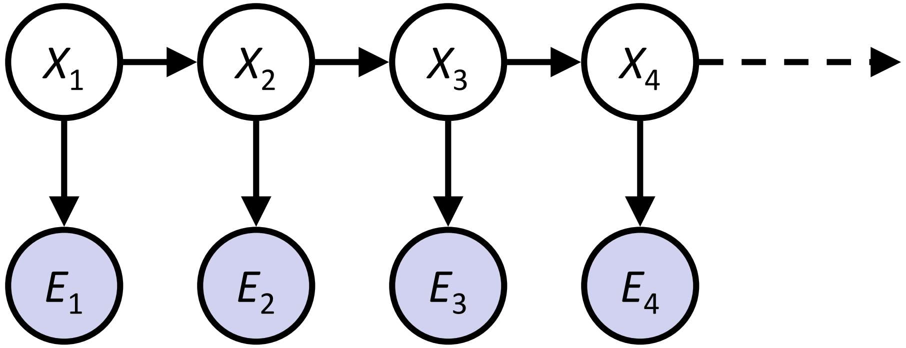
- Markov models



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



- Hidden Markov models



$$P(E|X)$$

X	E	P
rain	umbrella	0.9
rain	no umbrella	0.1
sun	umbrella	0.2
sun	no umbrella	0.8

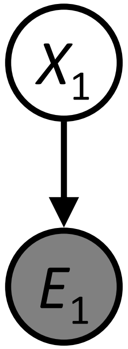
Inference: Find State Given Evidence

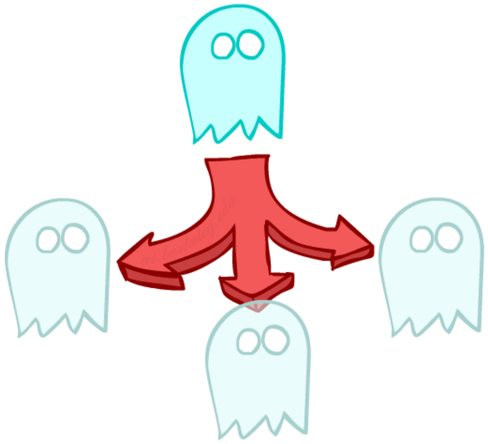
- We are given evidence at each time and want to know

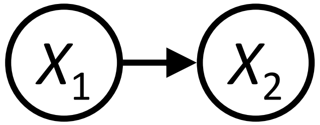
$$B_t(X) = P(X_t | e_{1:t})$$

- 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

Inference: Base Cases



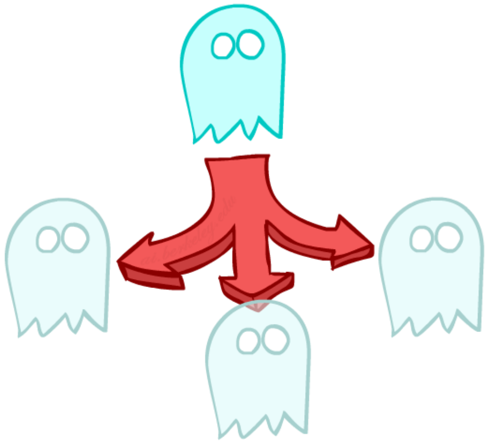


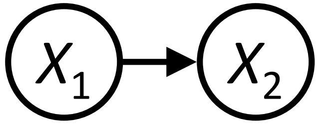


$P(X_1|e_1)$

$P(X_2)$

Inference: Base Cases





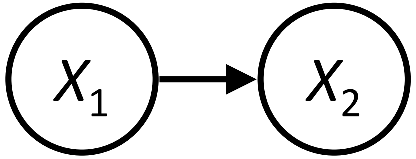
$$P(X_2)$$

$$P(x_2) = \sum_{x_1} P(x_1, x_2)$$
$$= \sum_{x_1} P(x_1)P(x_2|x_1)$$

Passage of Time

- Assume we have current belief P(X | evidence to date)

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



- Then, after one time step passes:

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

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

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

- 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

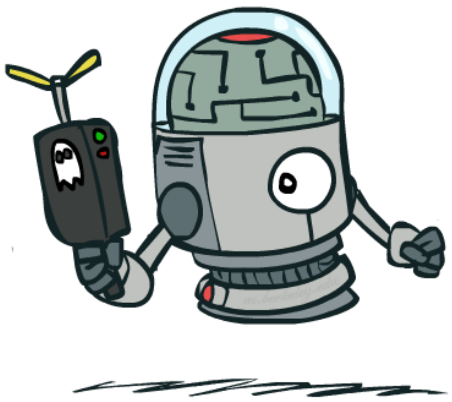
T = 1

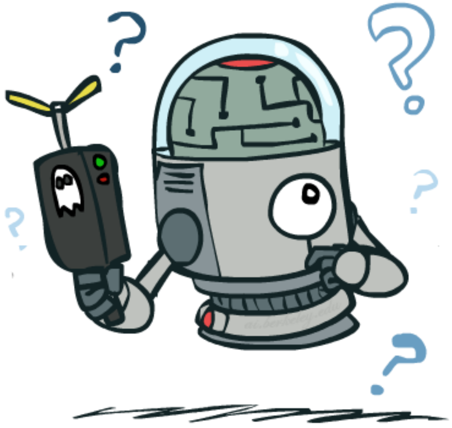
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

T = 2

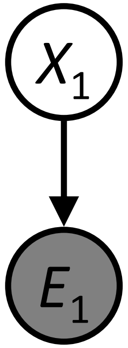
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

T = 5





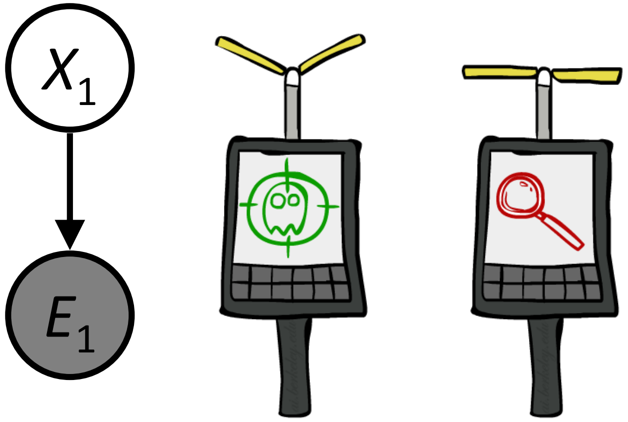
Inference: Base Cases



$$P(X_1|e_1)$$

$$\begin{aligned} P(x_1|e_1) &= P(x_1, e_1)/P(e_1) \\ &\propto_{X_1} P(x_1, e_1) \\ &= P(x_1)P(e_1|x_1) \end{aligned}$$

Observation



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

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

- Then, after evidence comes in:

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

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

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

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

- 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"

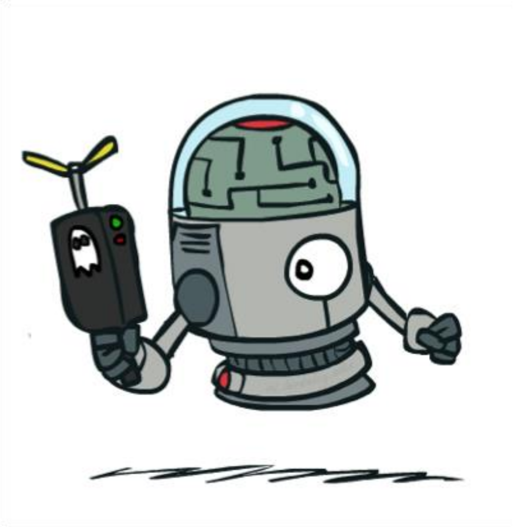
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

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

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



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

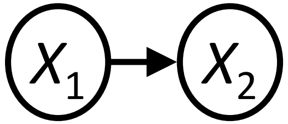
$$
\begin{aligned}
P(x_t|e_{1:t}) &\propto_X P(x_t, e_{1:t}) \\
&= \sum_{x_{t-1}} P(x_{t-1}, x_t, e_{1:t}) \\
&= \sum_{x_{t-1}} P(x_{t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t) \\
&= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})
\end{aligned}
$$

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

Filtering: $P(X_t \mid \text{evidence}_{1:t})$

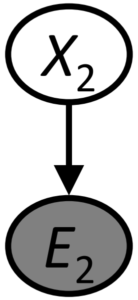
Elapse time: compute $P(X_t \mid e_{1:t-1})$

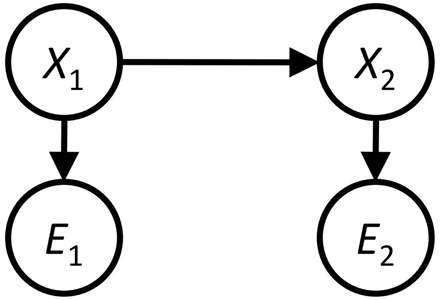
$$P(x_t|e_{1:t-1}) = \sum_{x_{t-1}} P(x_{t-1}|e_{1:t-1}) \cdot P(x_t|x_{t-1})$$



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: <P(rain), P(sun)>

$P(X_1)$	<0.5, 0.5>	*Prior on X_1*
$P(X_1 \mid E_1 = umbrella)$	<0.82, 0.18>	*Observe*
$P(X_2 \mid E_1 = umbrella)$	<0.63, 0.37>	*Elapse time*
$P(X_2 \mid E_1 = umb, E_2 = umb)$	<0.88, 0.12>	*Observe*

Example: Weather HMM

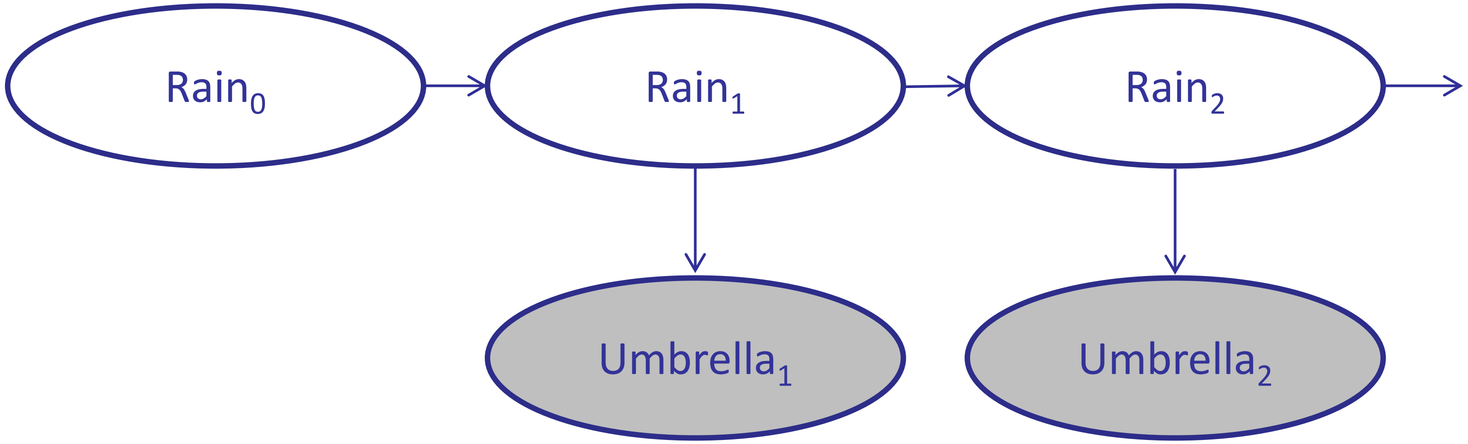
B(+r) = 0.5
B(-r) = 0.5

B'(+r) = 0.5
B'(-r) = 0.5

B(+r) = 0.818
B(-r) = 0.182

B'(+r) = 0.627
B'(-r) = 0.373

B(+r) = 0.883
B(-r) = 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

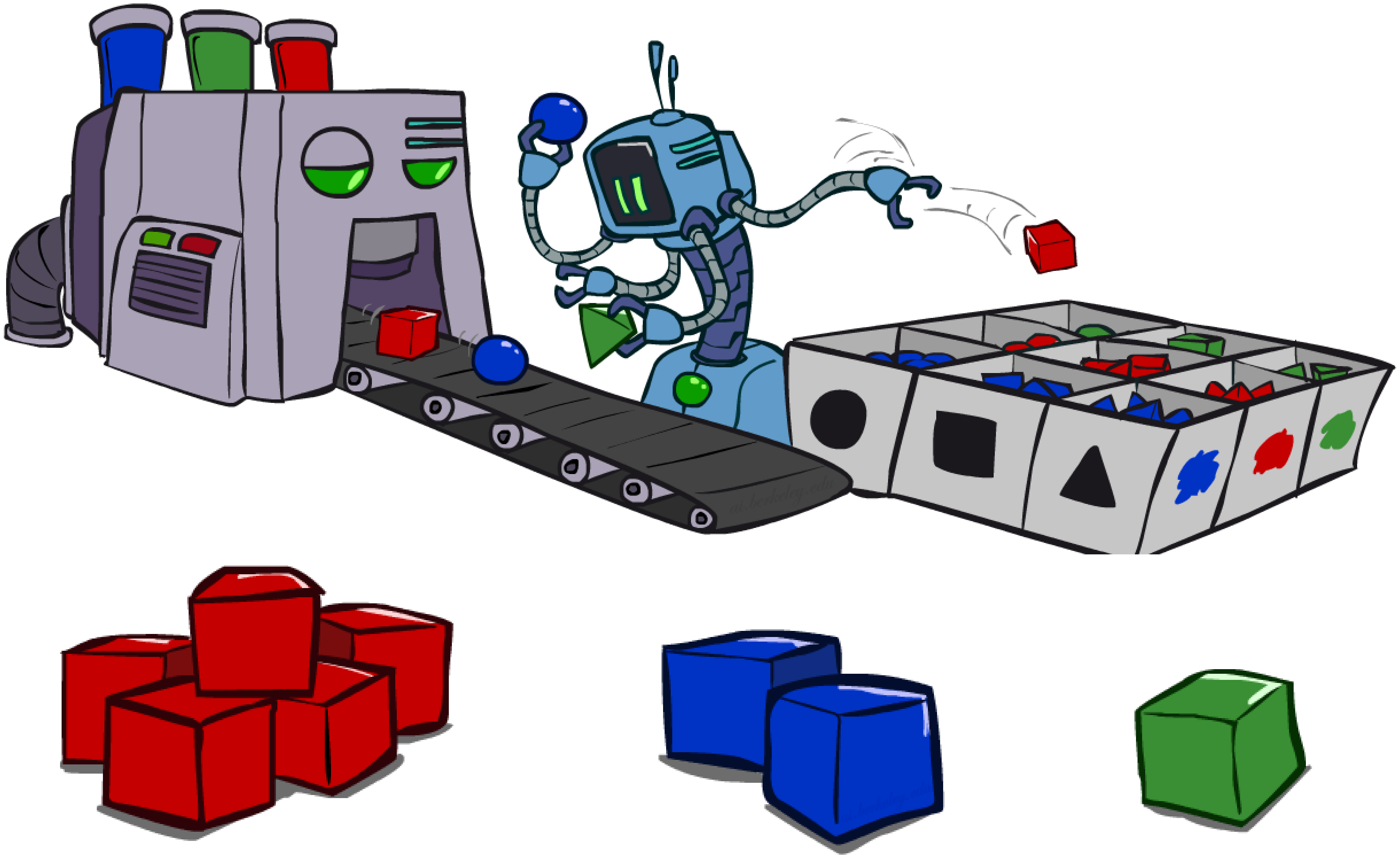
Pacman – Sonar (P4)

Approximate Inference

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

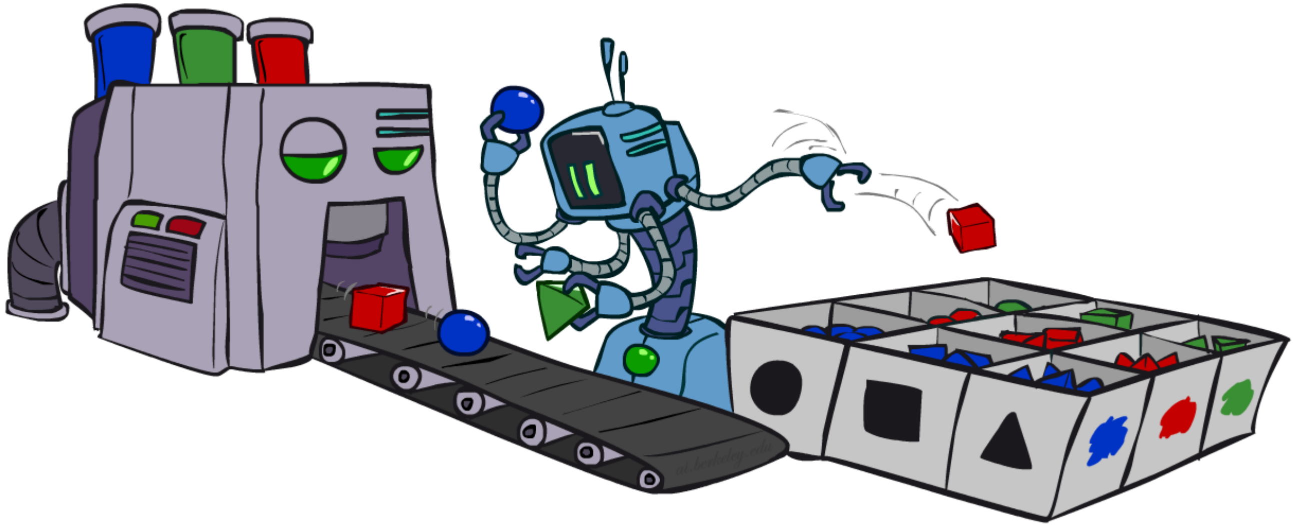
- Solution: approximate inference by sampling
- How robot localization works in practice

Approximate Inference: Sampling



Sampling

- Sampling is a lot like repeated simulation
  - Predicting the weather, basketball games, …
- Basic idea
  - Draw N samples from a sampling distribution S
  - Compute an approximate probability
- Why sample?
  - Learning: get samples from a distribution you don't know
  - Inference: getting a sample is faster than computing the right answer



Sampling

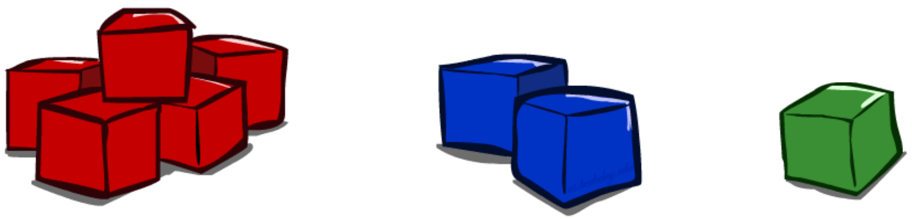
- Sampling from given distribution
  - Step 1: Get sample *u* from uniform distribution over [0, 1)
    - E.g. random() in python
  - Step 2: Convert this sample *u* into an outcome for the given distribution by having each target outcome associated with a sub-interval of [0,1) with sub-interval size equal to probability of the outcome

- Example

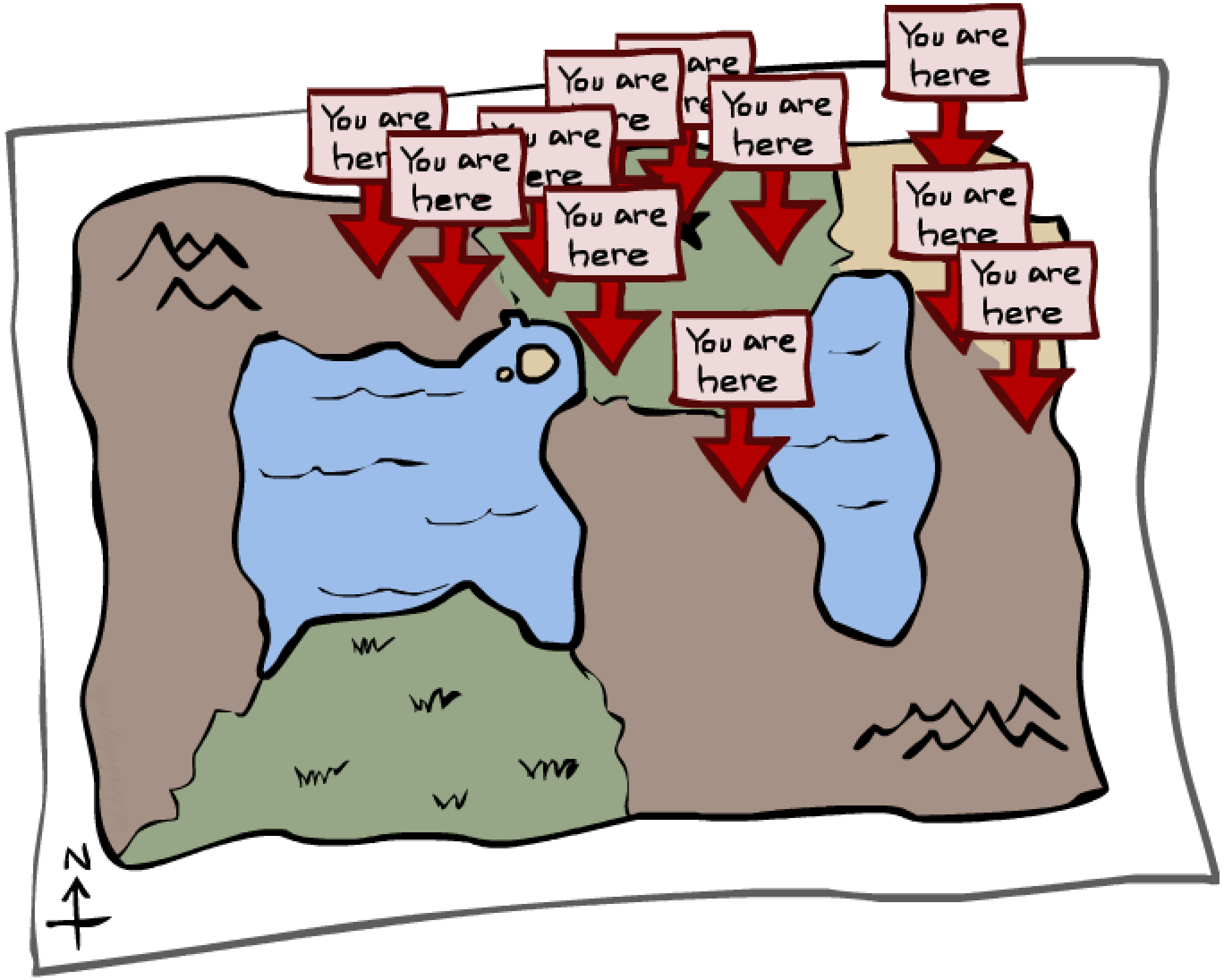
C	P(C)
red	0.6
green	0.1
blue	0.3

$$0 \leq u < 0.6, \rightarrow C = red$$
$$0.6 \leq u < 0.7, \rightarrow C = green$$
$$0.7 \leq u < 1, \rightarrow C = blue$$

  - If random() returns *u* = 0.83, then our sample is *C* = blue
  - E.g, after sampling 8 times:

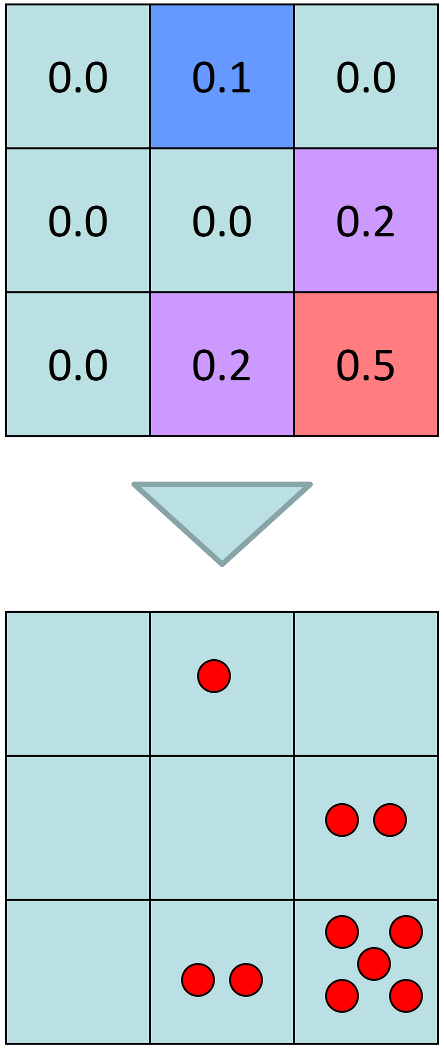


Particle Filtering



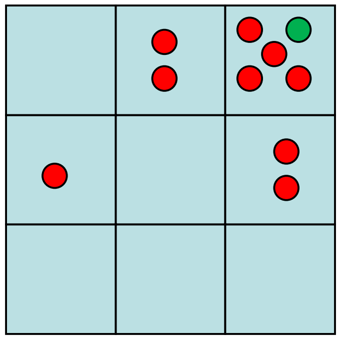
Particle Filtering

- 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



Representation: Particles

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

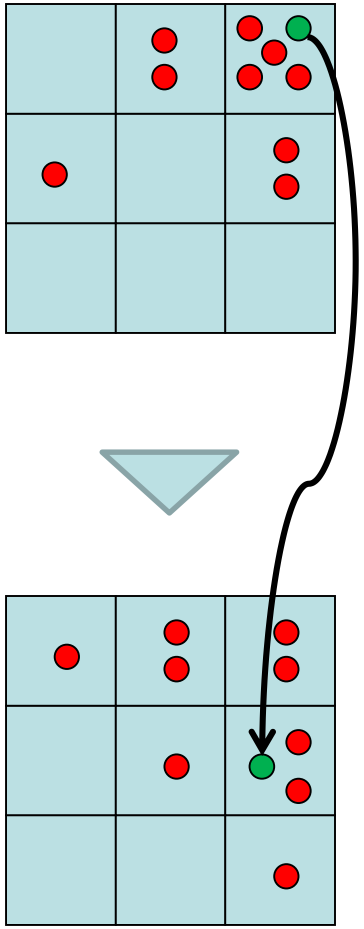
- Each particle is moved by sampling its next position from the transition model

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

  - 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

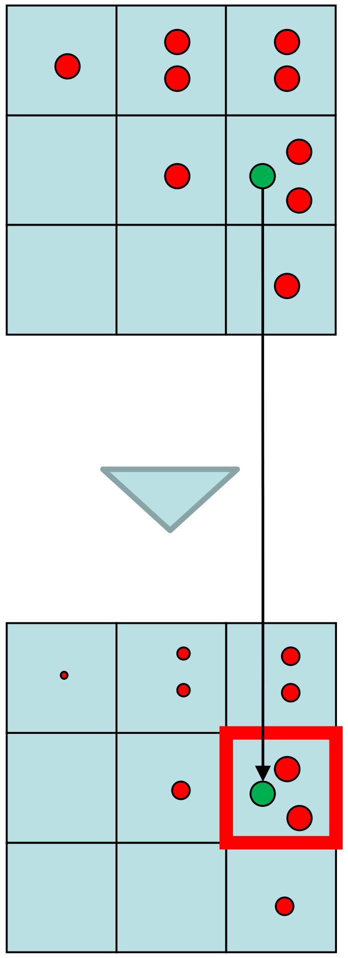
- Slightly trickier:
  - Don’t sample observation, fix it
  - Downweight samples based on the evidence

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

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

  - 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)

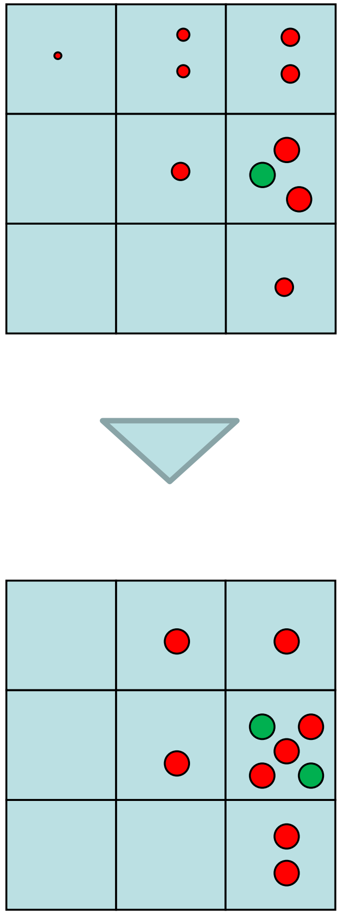


Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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

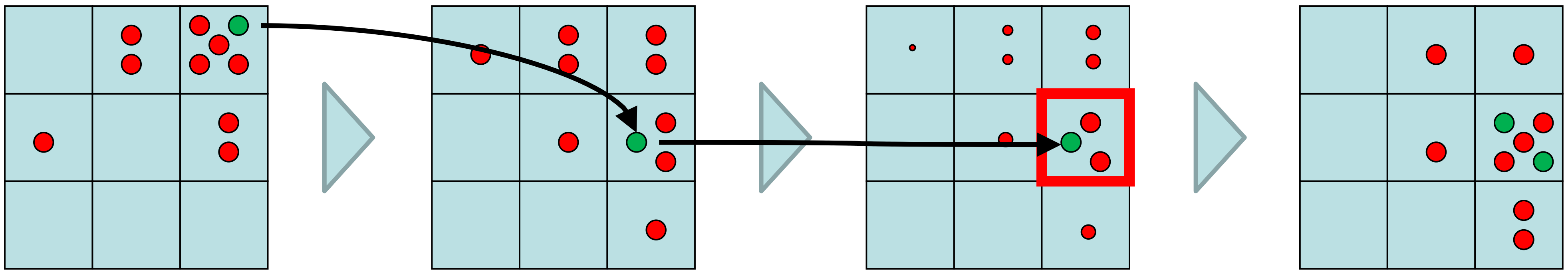


(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)

Recap: Particle Filtering

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

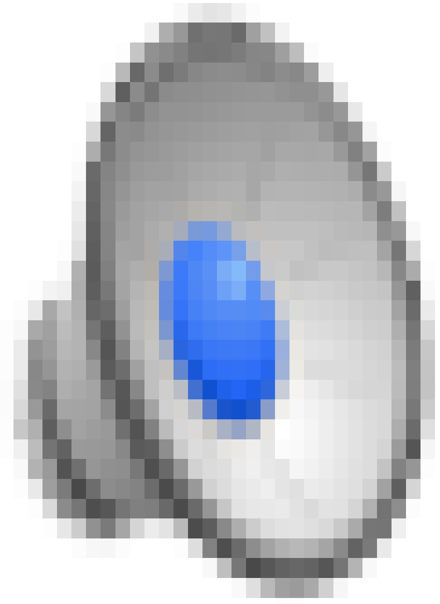
Elapse | Weight | Resample



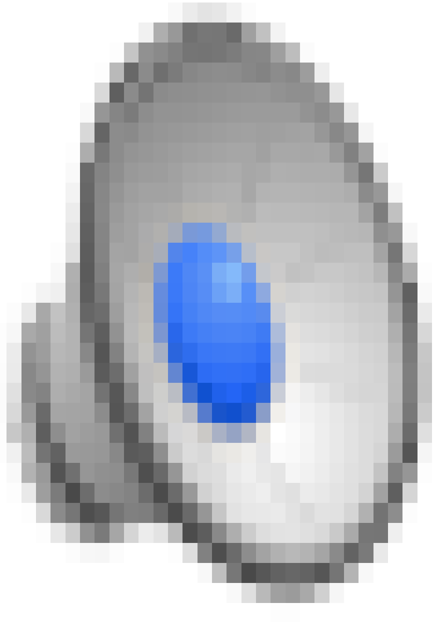
Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9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

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

Video of Demo – Moderate Number of Particles

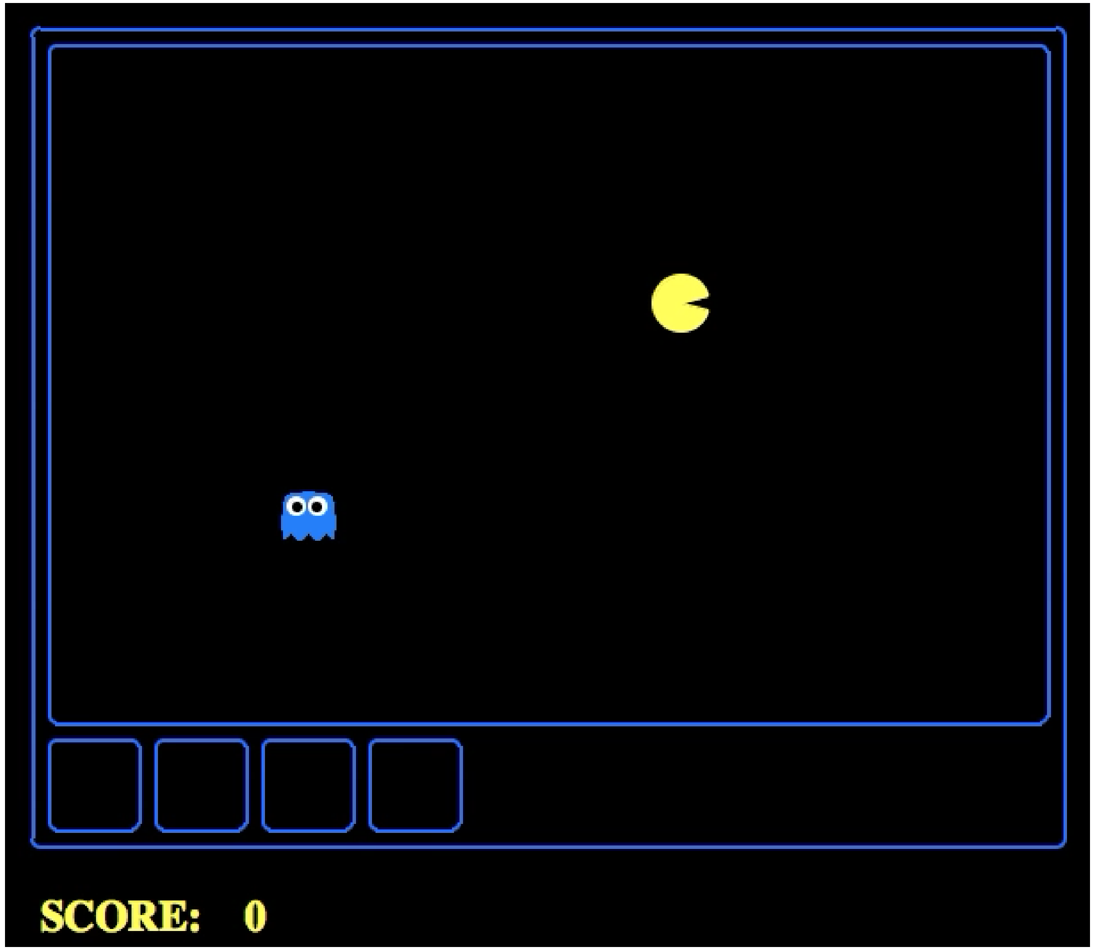


Video of Demo – Huge Number of Particles



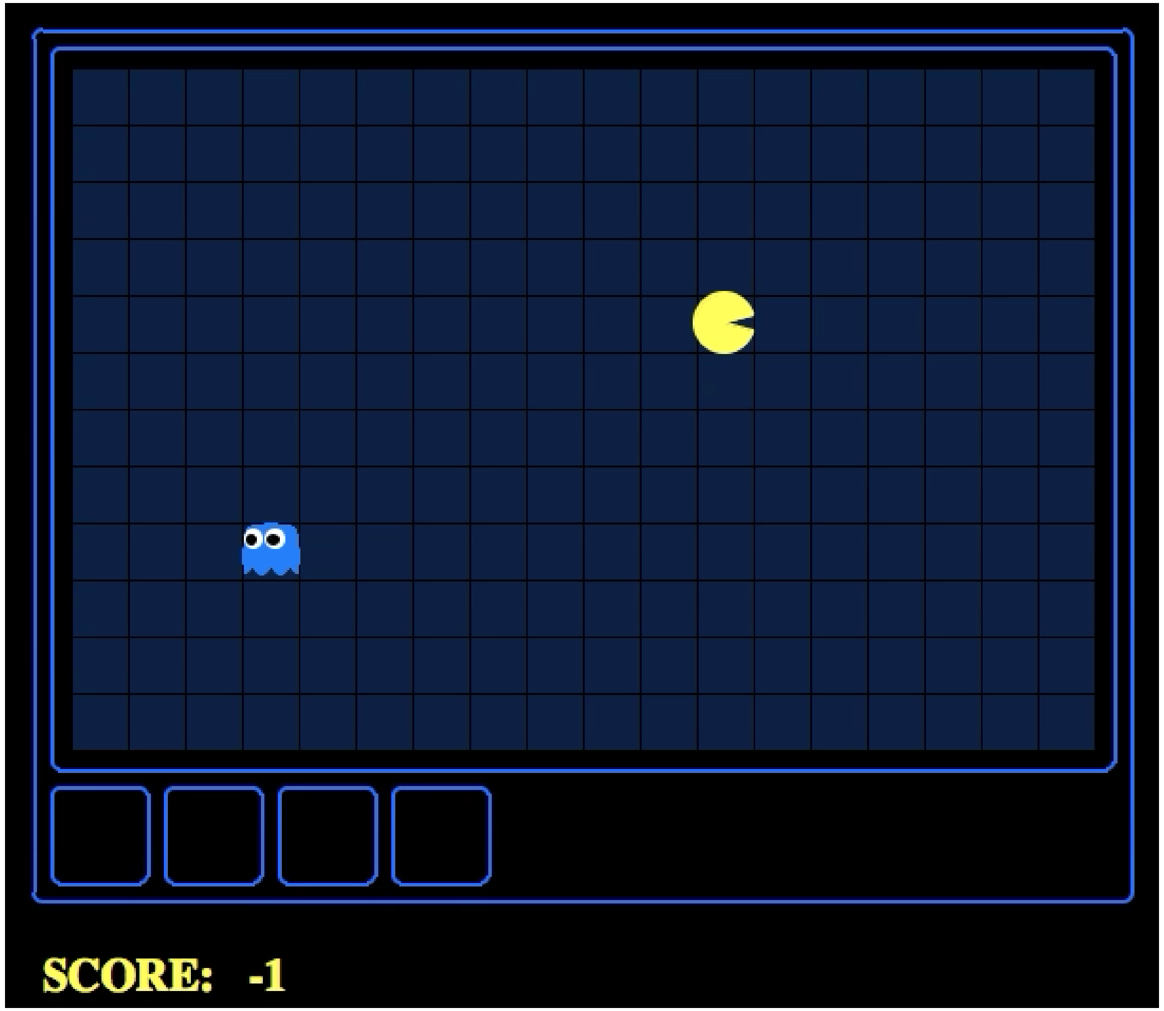
Which Algorithm?

Particle filter, uniform initial beliefs, 25 particles



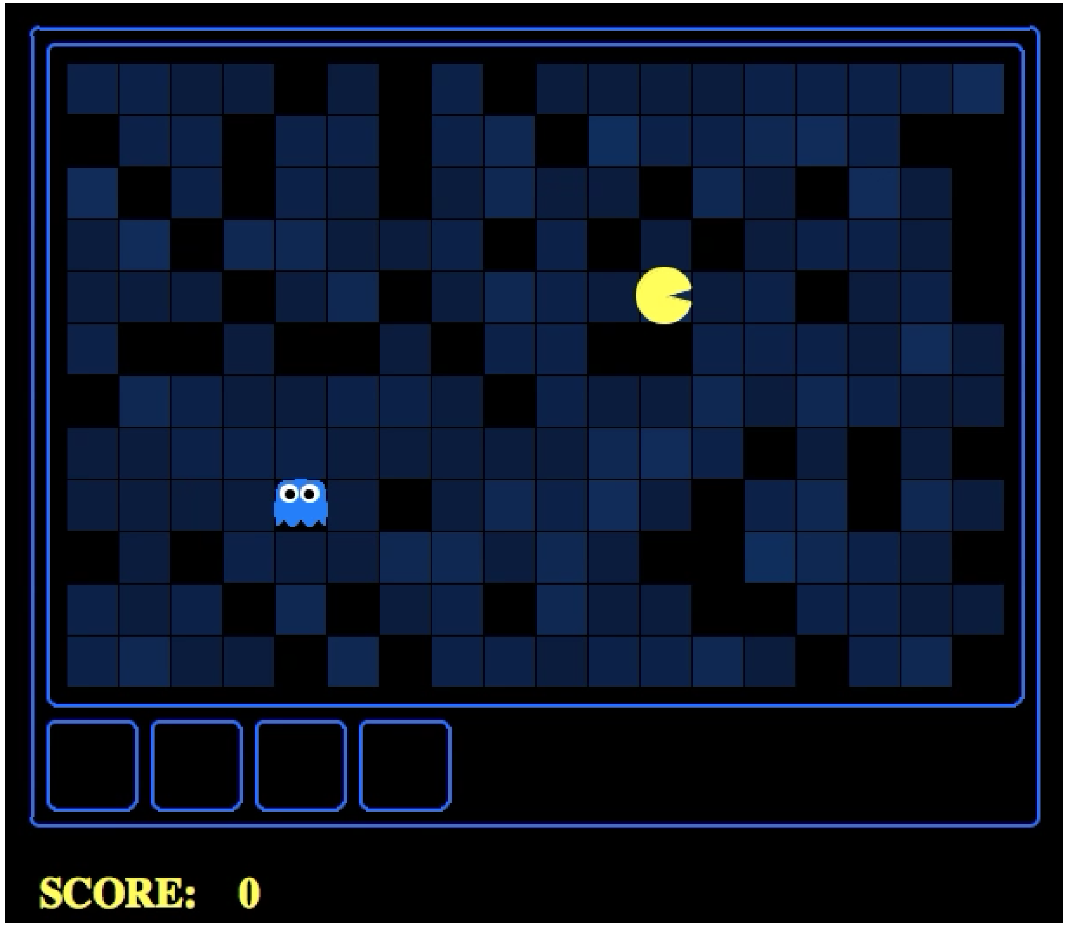
Which Algorithm?

Exact filter, uniform initial beliefs



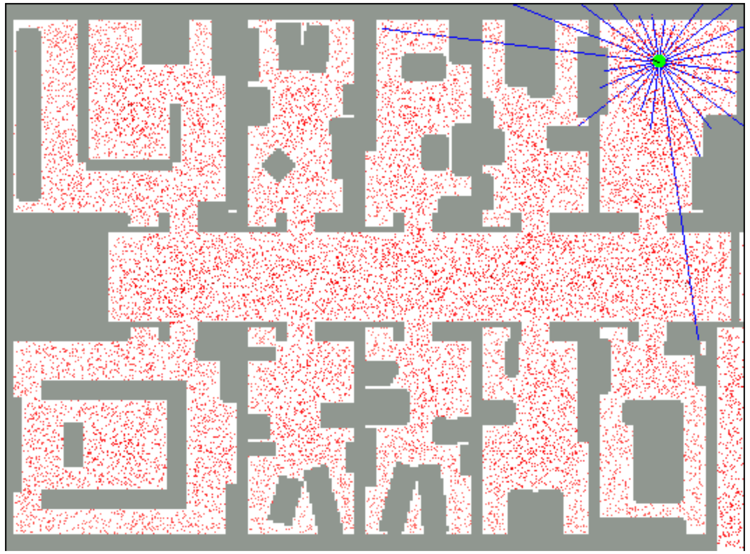
Which Algorithm?

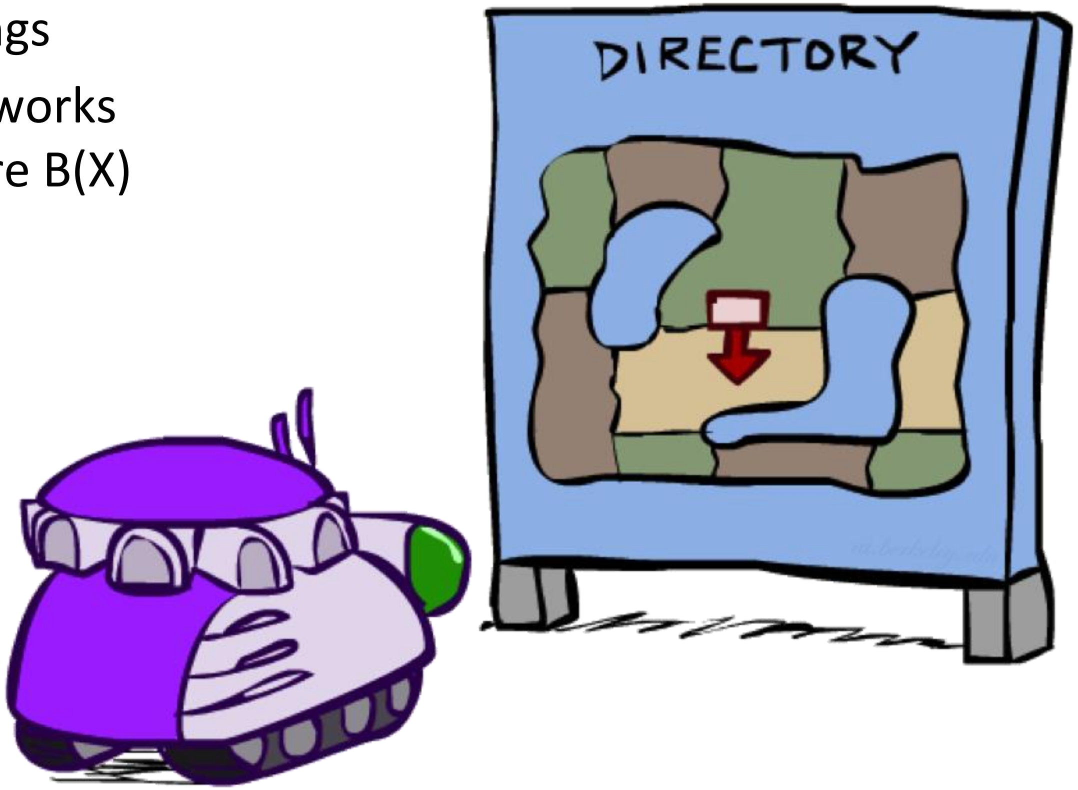
Particle filter, uniform initial beliefs, 300 particles



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)

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

Particle Filter Localization (Laser)