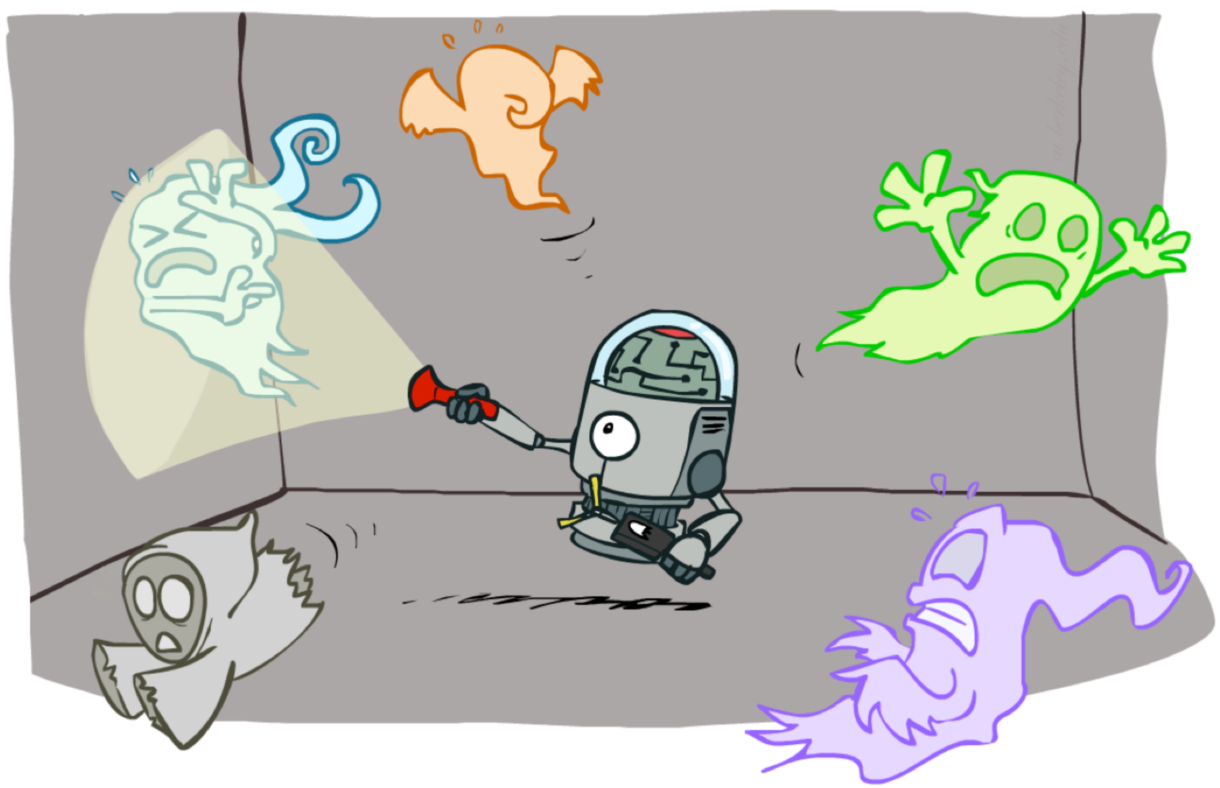
Ve492: Introduction to Artificial Intelligence
Hidden Markov Models II



Paul Weng

UM-SJTU Joint Institute

Slides adapted from http://ai.berkeley.edu, AIMA, UM, CMU

Today

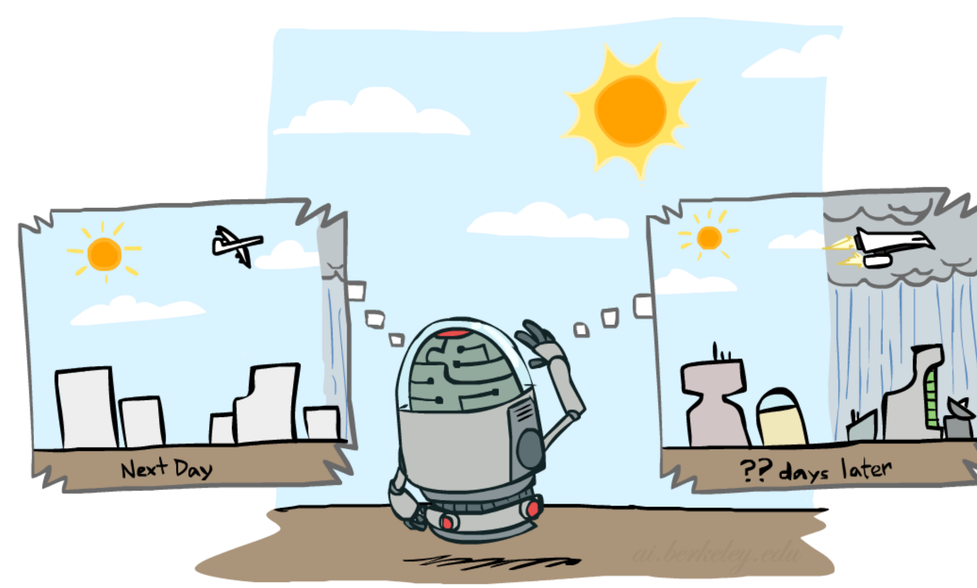
- Quick Review of (H)MM
- Particle Filtering
- Viterbi Algorithm for Most Likely Explanation
- Dynamic Bayesian Network

Markov Chain

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

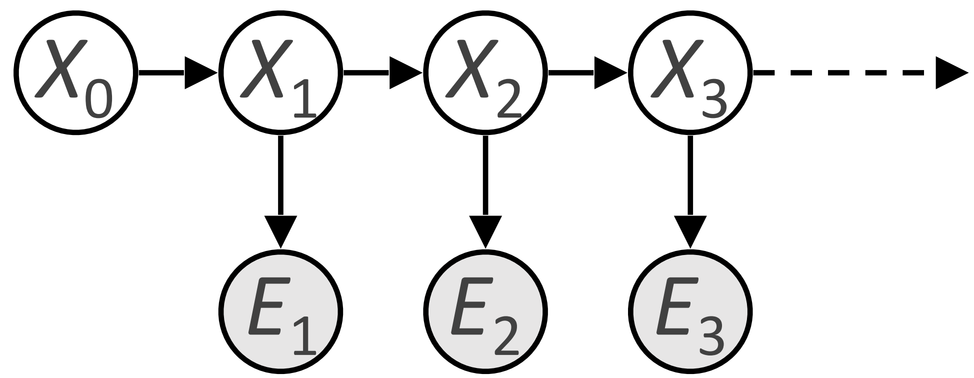
- You know transition probabilities, $P(X_t \mid X_{t-1})$, and $P(X_4)$.
- Write an equation to compute $P(X_5)$.

$$P(X_5) = \sum_{x_4} P(x_4, X_5)$$
$$= \sum_{x_4} P(X_5 \mid x_4) P(x_4)$$



- More generally: $P_{t+1} = T^{\intercal} P_t$ where $P_t = P(X_t)$
- Stationary distribution: $P_\infty = T^{\intercal} P_\infty$

HMM, Filtering, Forward Algorithm



Filtering: What is the current state, given all evidence?

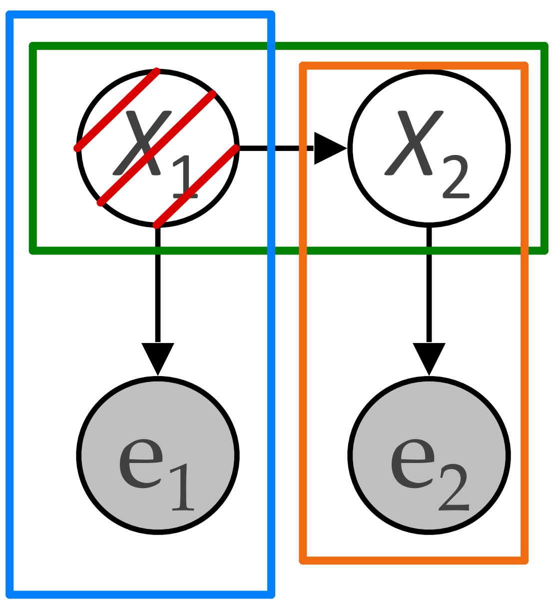
$$P(X_{t+1} | e_{1:t+1}) = \underbrace{\alpha}_{\text{Normalize}} \; \underbrace{P(e_{t+1} | X_{t+1})}_{\text{Update}} \; \underbrace{\sum_{x_t} P(X_{t+1} | x_t) \, P(x_t \mid e_{1:t})}_{\text{Predict}}$$

$$\boldsymbol{f}_{1:t+1} = \textsc{Forward}(\boldsymbol{f}_{1:t} \, , e_{t+1})$$

Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

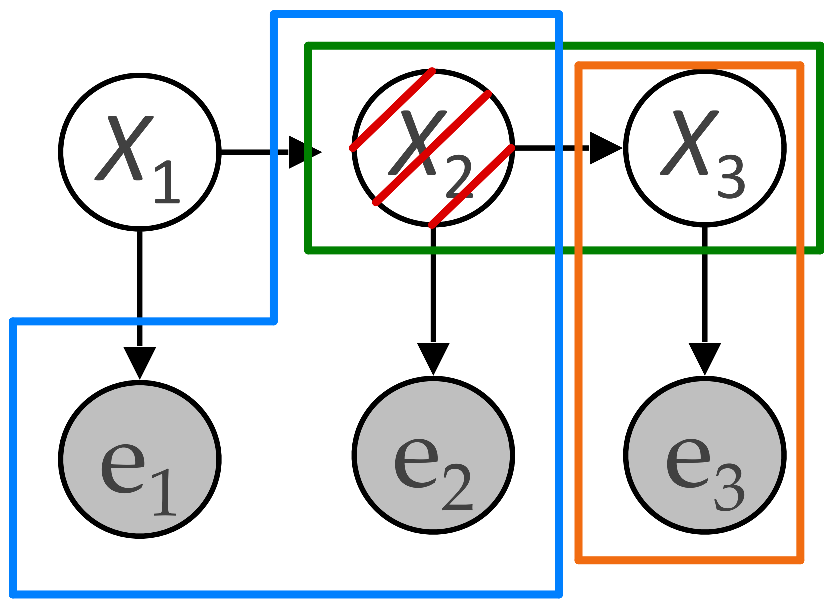
Marching **forward** through the HMM network



Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

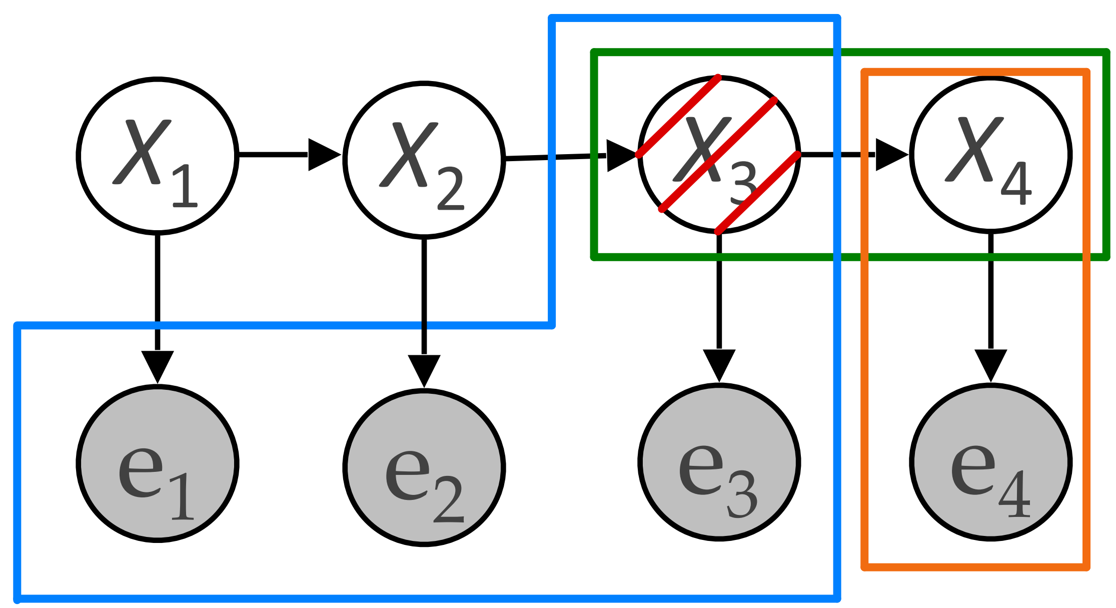
Marching **forward** through the HMM network



Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

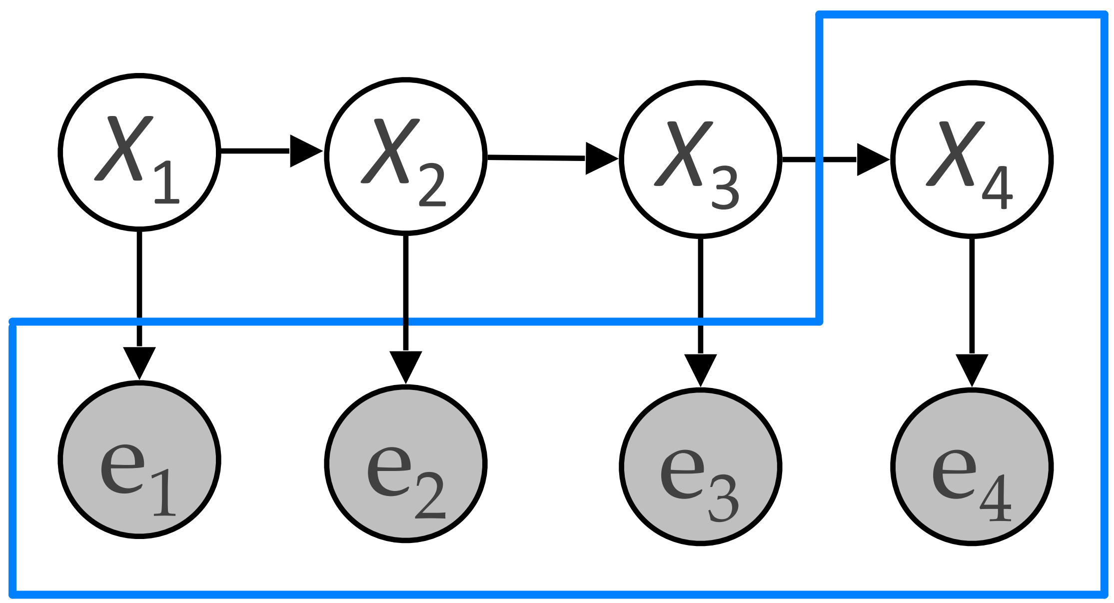
Marching **forward** through the HMM network



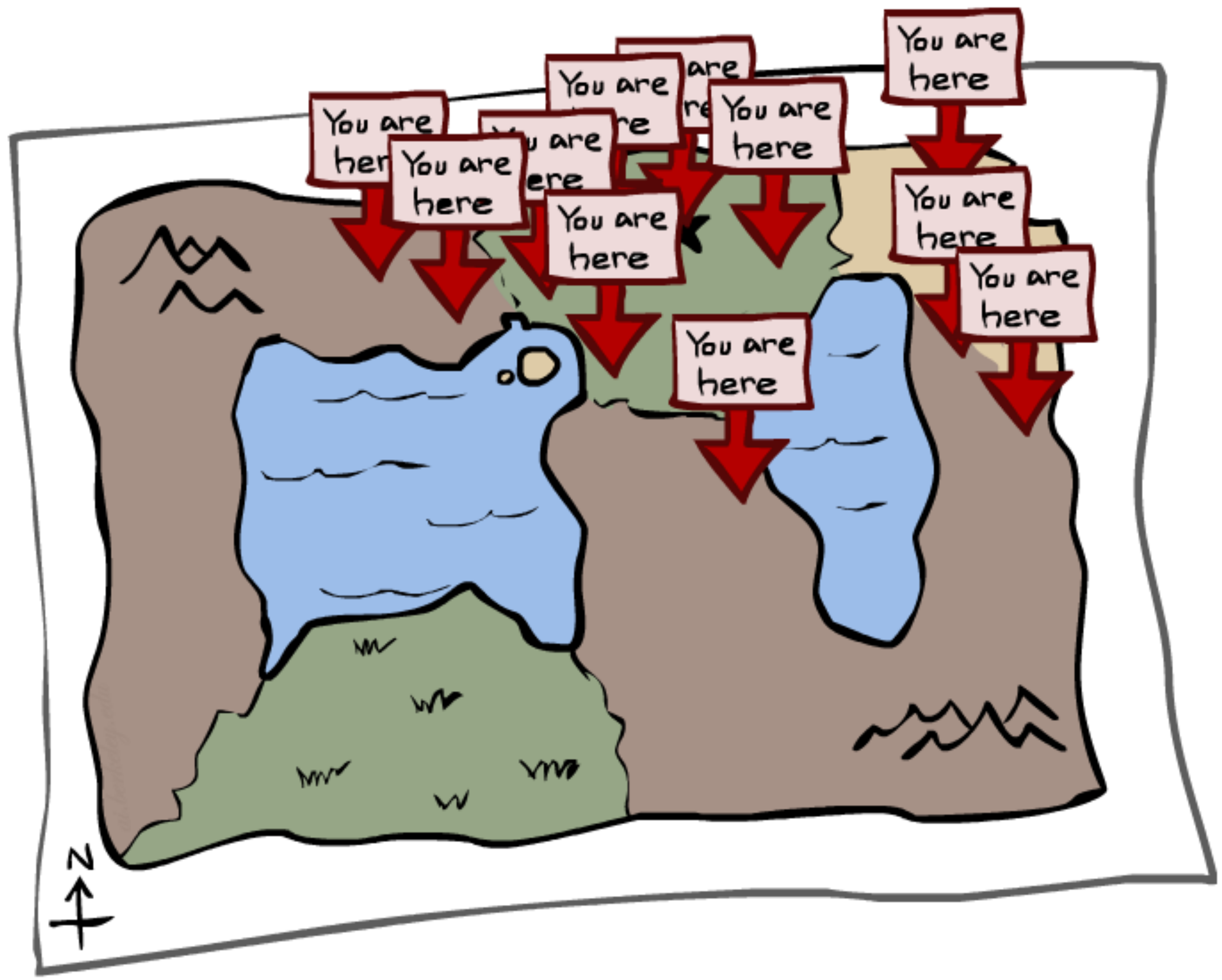
Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

Marching **forward** through the HMM network



Particle Filtering



Quiz: Algorithms for Filtering

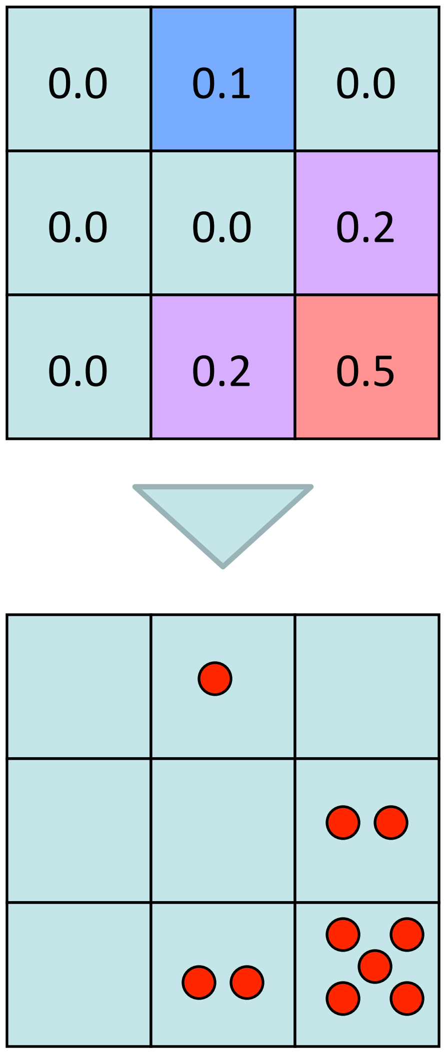
- Which algorithms can be applied to the filtering task?
  - Variable elimination
  - Prior sampling
  - Rejection sampling
  - Likelihood weighting
  - Gibbs sampling

Limitations of Current Algorithms

- Exact inference infeasible if state space is large or continuous
  - e.g., when $|X|$ is more than 10^6 or so (tracking 3 ghosts in a 10x20 world)
  - e.g., robot localization
- Previous sampling methods do not exploit the temporal structure of an HMM

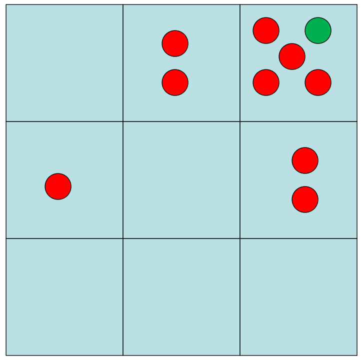
Particle Filtering

- PF = approximate inference for filtering task
- Principle of Particle Filtering
  - Track samples of X_t, 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



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
- In PF, a set of samples approximates $f_{1:t}(X_t) = P(X_t \mid e_{1:t})$



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

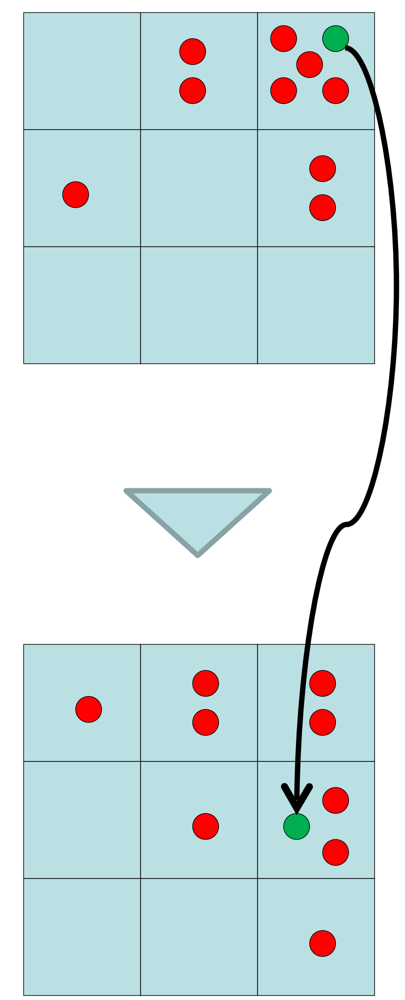
- This first step captures the passage of time
- Each particle is moved by sampling its next position from the transition model

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

  - 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
- New particles approximates: $\sum_{x_t} P(X_{t+1} | x_t) f_{1:t}(x_t)$
- **Consistency**: If enough samples, close to exact values before and after

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

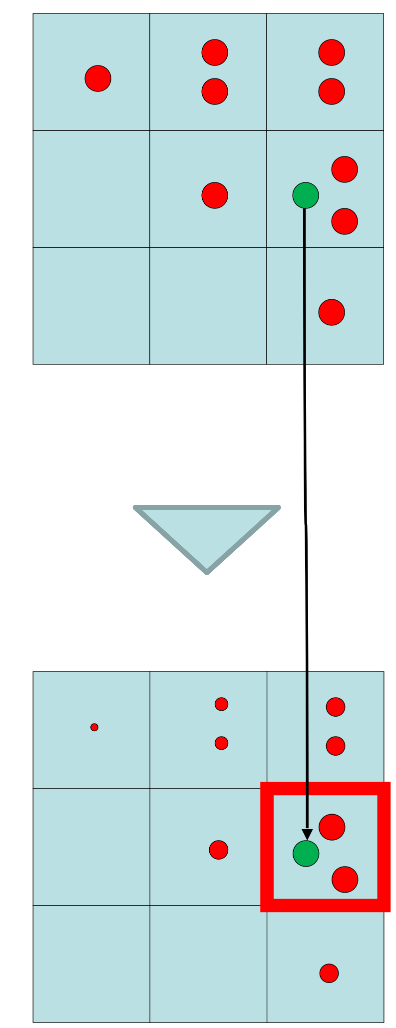
- Slightly trickier:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, weight samples based on the evidence

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

$$f_{1:t+1}(X) \propto w(X) \sum_{x_t} P(X|x_t) f_{1:t}(x_t)$$

  - As before, the probabilities don't sum to one, since all have been weighted
  - What do they sum to?

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

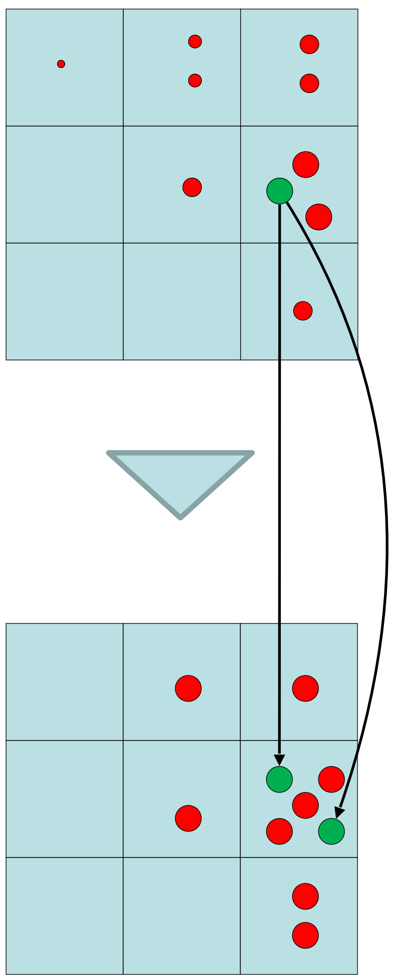


Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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

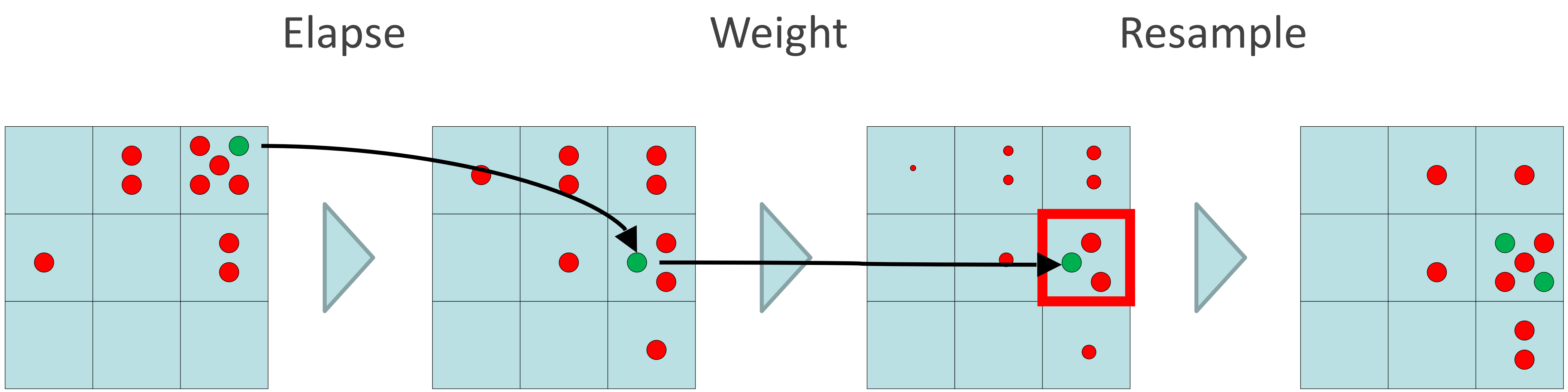
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4



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

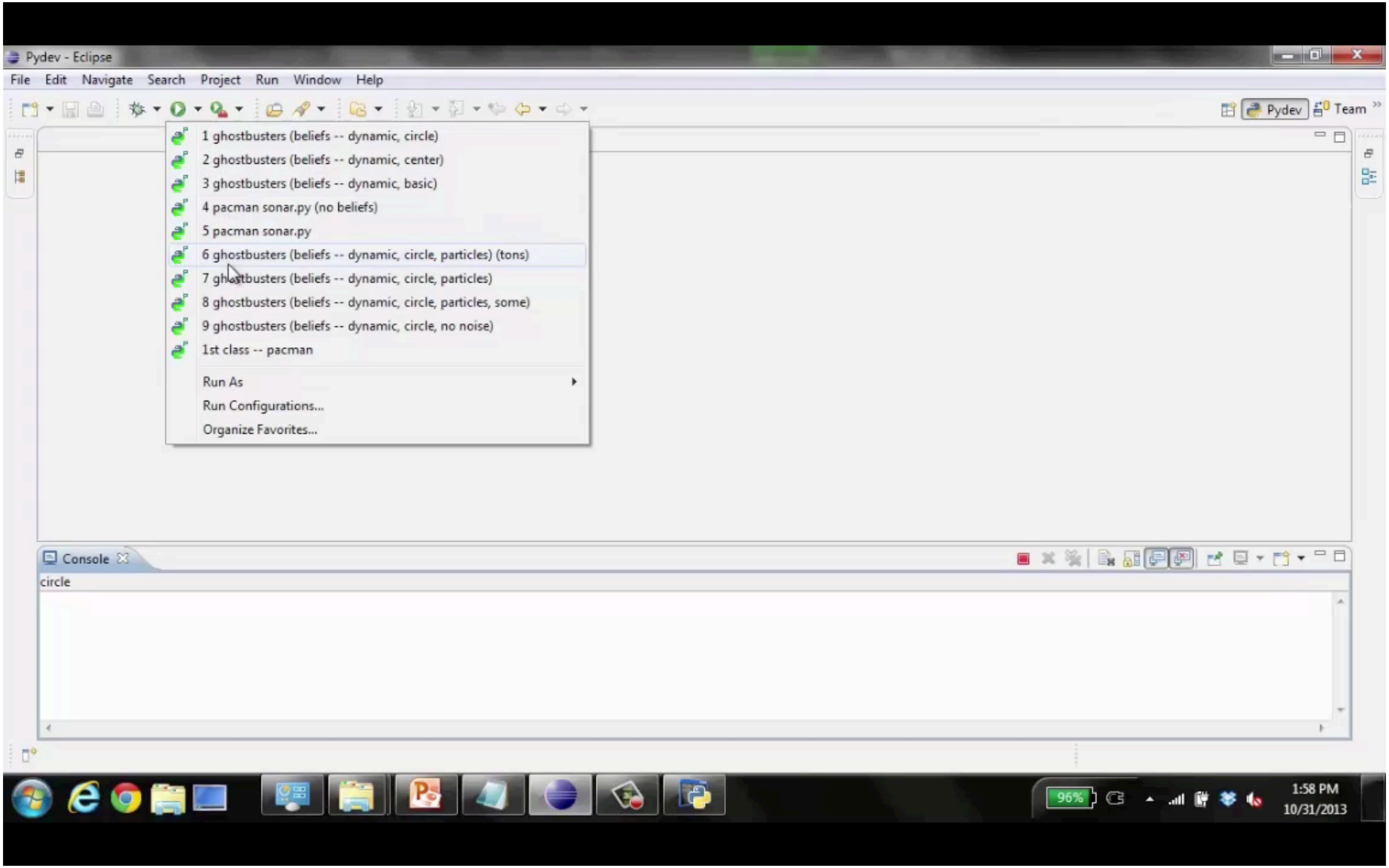
Summary: Particle Filtering

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

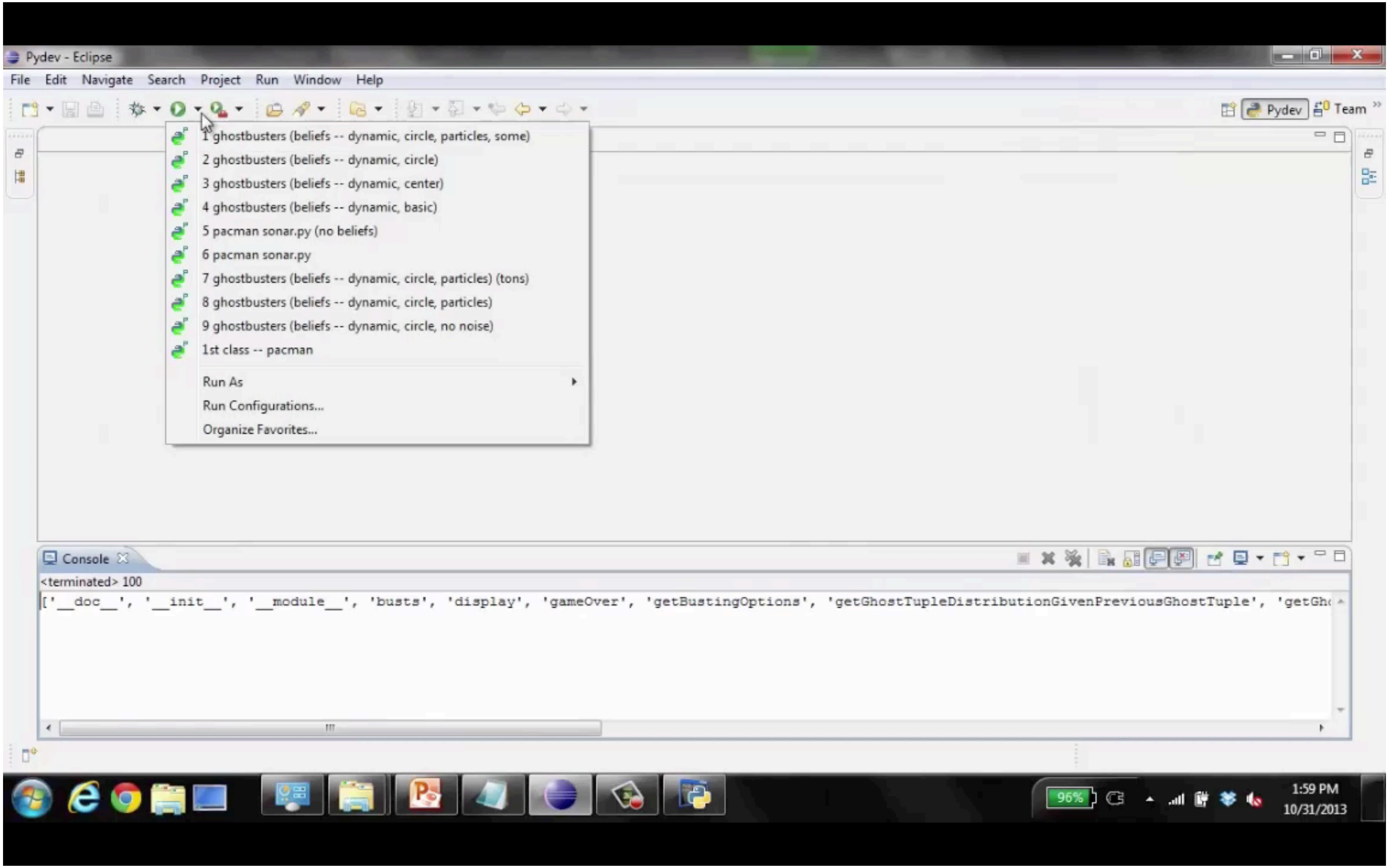


Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9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

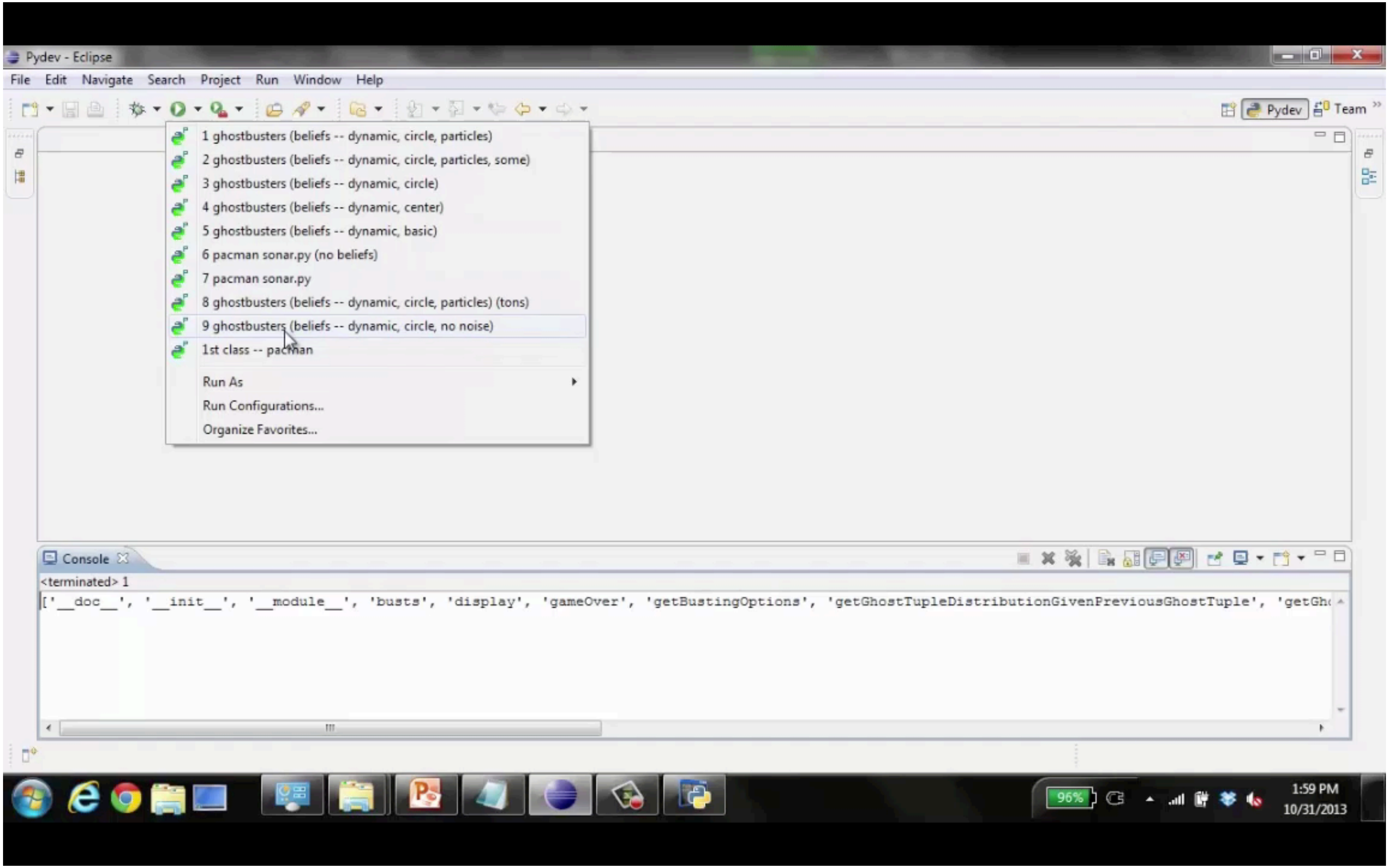
Video of Demo – Moderate Number of Particles



Video of Demo – One Particle



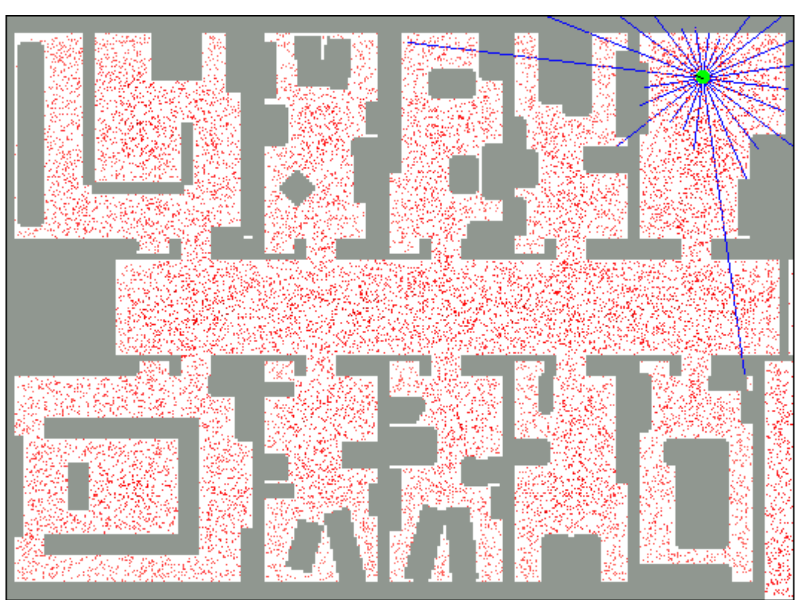
Video of Demo – Huge Number of Particles

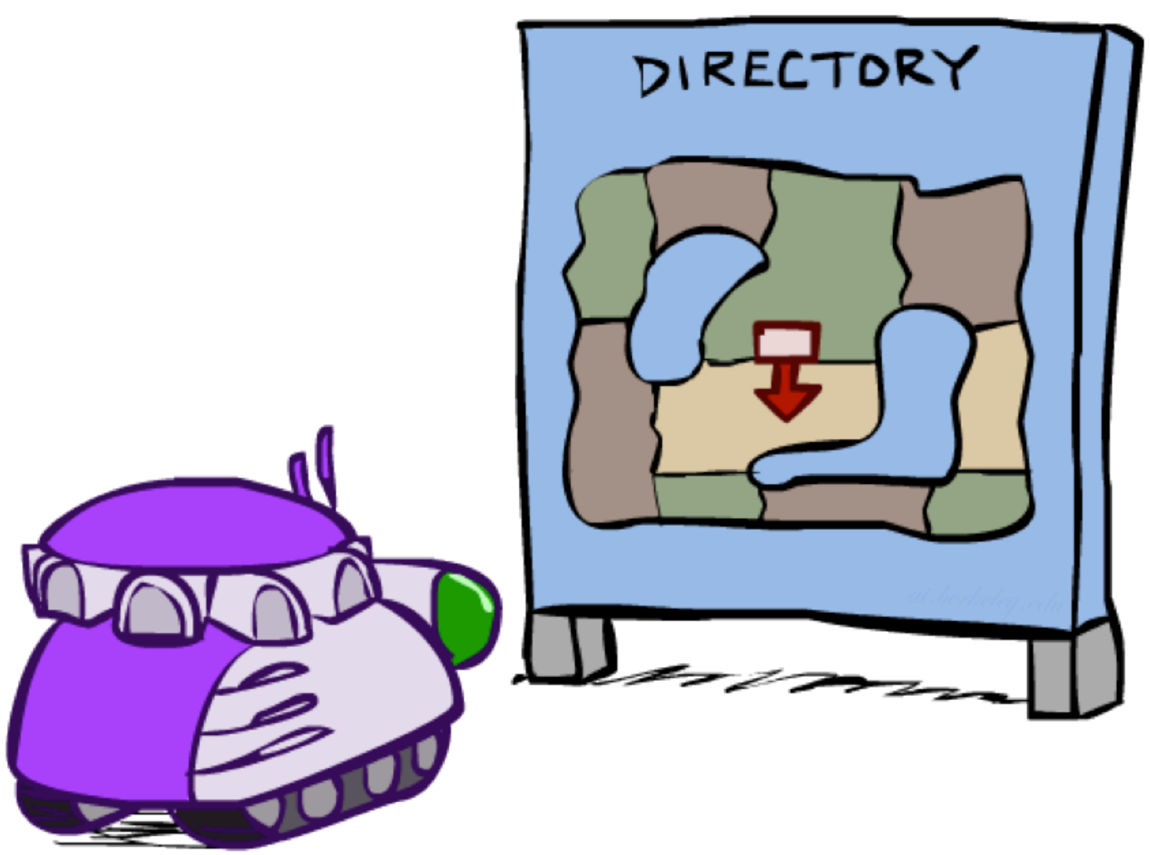


Robot Localization

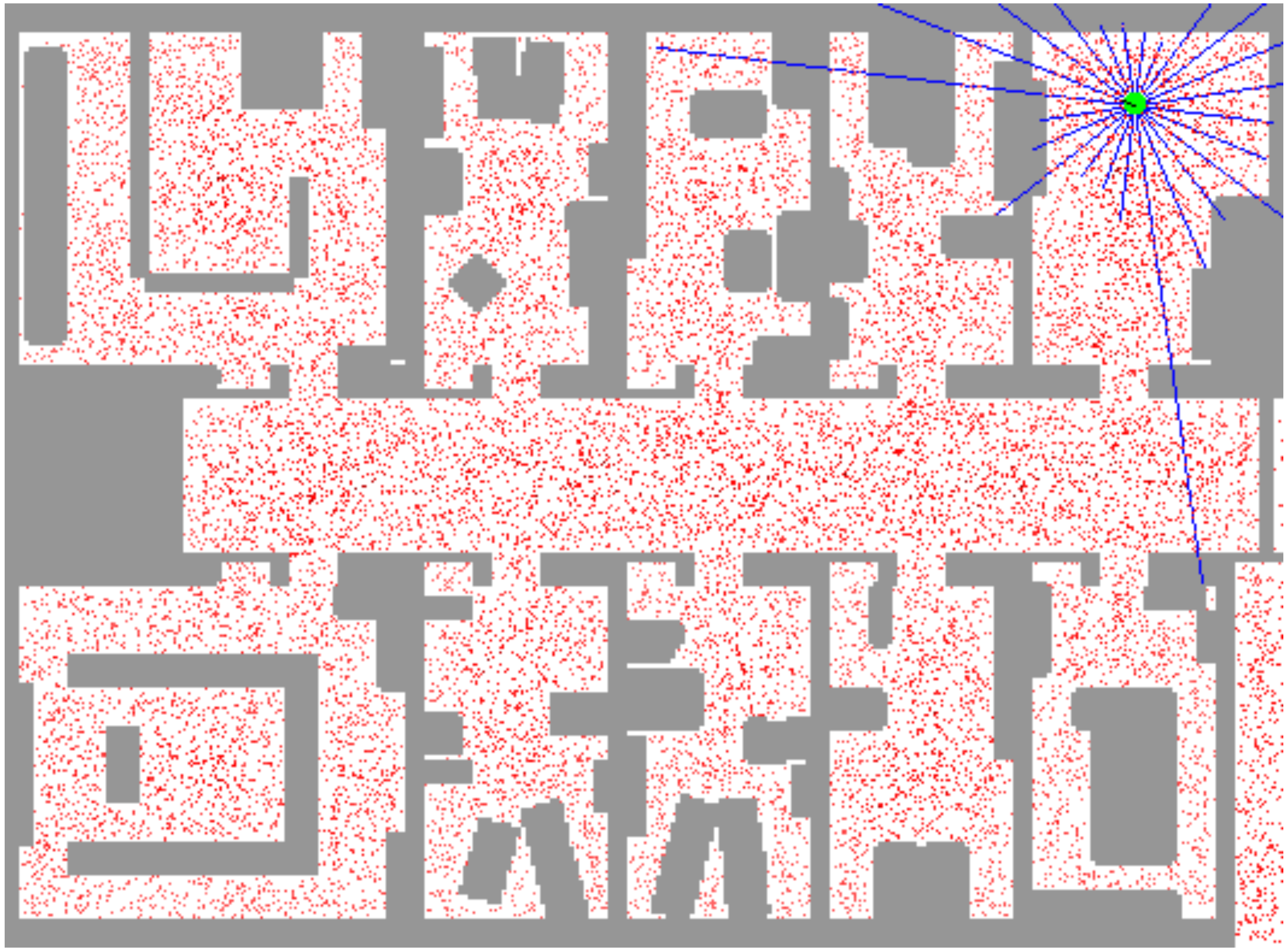
In robot localization:

- 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
- Particle filtering is a main technique





Particle Filter Localization (Laser)



[Dieter Fox, et al.]

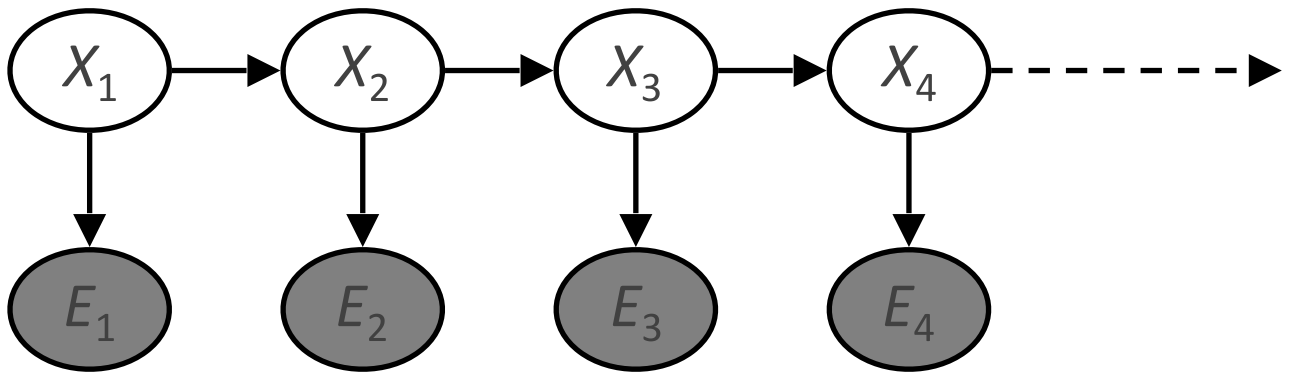
Most Likely Explanation

Inference tasks

- ***Filtering***: $P(X_t | e_{1:t})$
  - ***belief state***—input to the decision process of a rational agent
- ***Prediction***: $P(X_{t+k} | e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- ***Smoothing***: $P(X_k | e_{1:t})$ for $0 \le k < t$
  - better estimate of past states, essential for learning
- ***Most likely explanation***: $\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
  - speech recognition, decoding with a noisy channel

HMMs: MLE Queries

- HMMs defined by
  - States X
  - Observations E
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t \mid X_{t-1})$
  - Emissions: $P(E \mid X)$

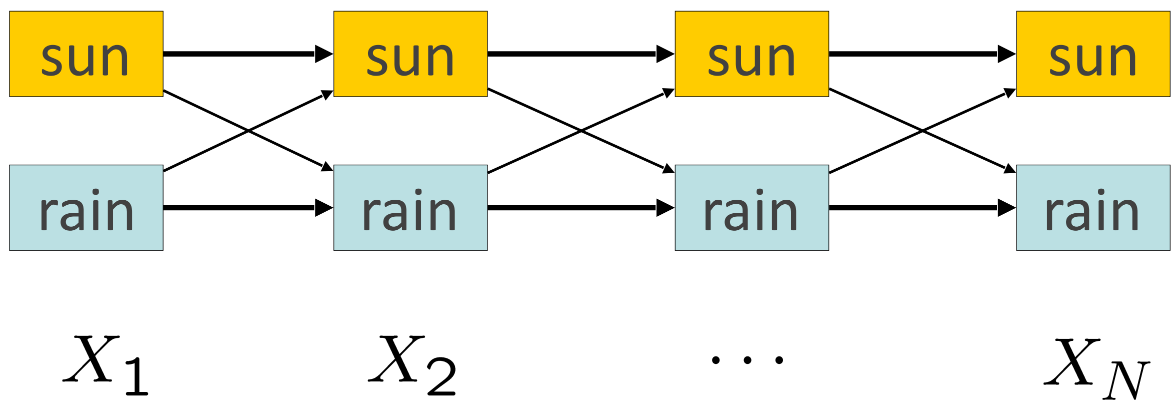


- New query: most likely explanation: $arg\,\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$

- New method: the Viterbi algorithm

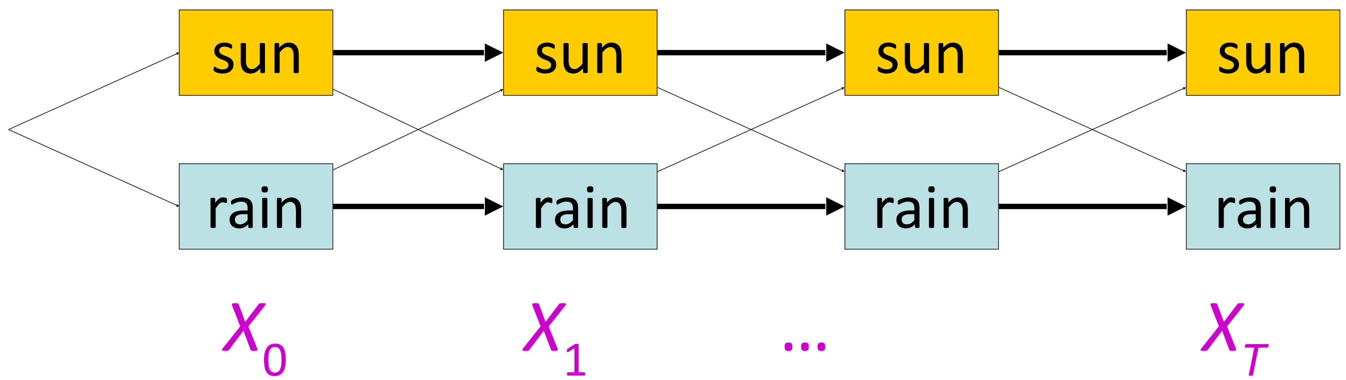
State Trellis

- 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
- Product of weights on a path = sequence's probability along with the evidence
- Forward algorithm computes sums of paths, Viterbi computes best paths

Forward / Viterbi algorithms



Forward Algorithm (sum)

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

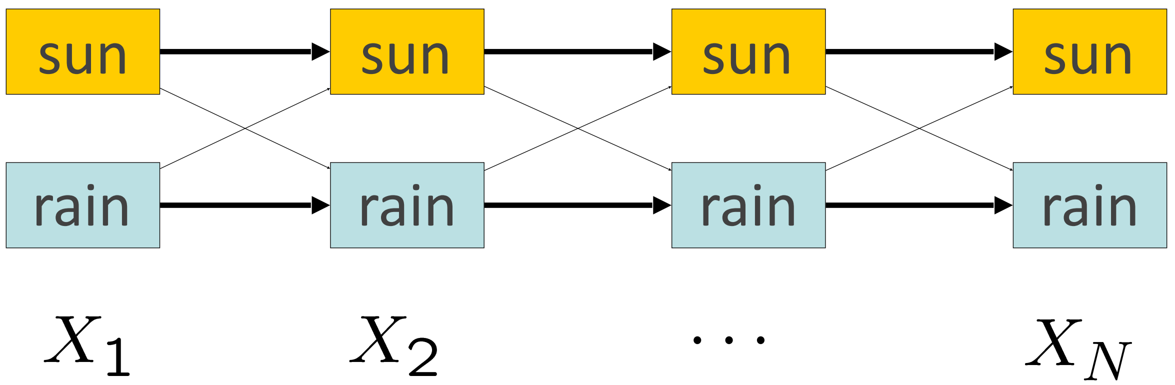
$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, e_{t+1})$$
$$= \alpha\, P(e_{t+1}|X_{t+1}) \textstyle\sum_{x_t} P(X_{t+1}\,|\, x_t)\, \mathbf{f}_{1:t}$$

Viterbi Algorithm (max)

For each state at time t, keep track of the ***maximum probability of any path*** to it

$$\mathbf{m}_{1:t+1} = \text{VITERBI}(\mathbf{m}_{1:t}, e_{t+1})$$
$$= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}\,|\, x_t)\, \mathbf{m}_{1:t}$$

Why is This True?



$$m_1[x_1] = P(e_1|x_1)P(x_1)$$

probability of best path $1:t$ that ends at x_t

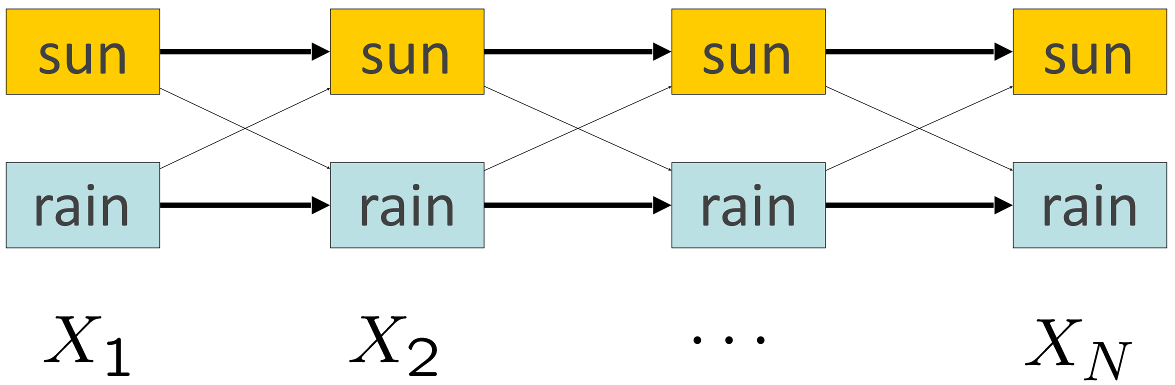
$$m_2[x_2] = \max\{P(e_2|x_2)P(x_2|x_1 = \text{rain})m_1[x_1 = \text{rain}], P(e_2|x_2)P(x_2|x_1 = \text{sun})m_1[x_1 = \text{sun}]\}$$

$$= \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

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

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

Now What?



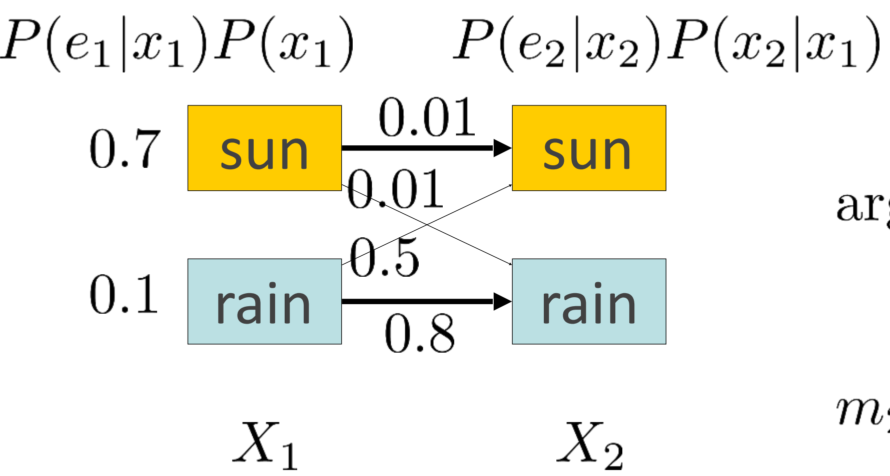
$m_N[x_N]$ probability of best path $1:N$ that ends at x_N

what is the last state on the most likely path?

$$\arg\max_{x_N} m_N[x_N]$$

what is the *second to last* state on the most likely path?

A Tricky Counter-Example



$$\arg\max_{x_1} m_1[x_1] = ?$$ sun!

$$m_2[x_2] = \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

$$m_2[x_2 = \text{sun}] = \max\{0.7 \times 0.01, 0.1 \times 0.5\} = 0.05$$

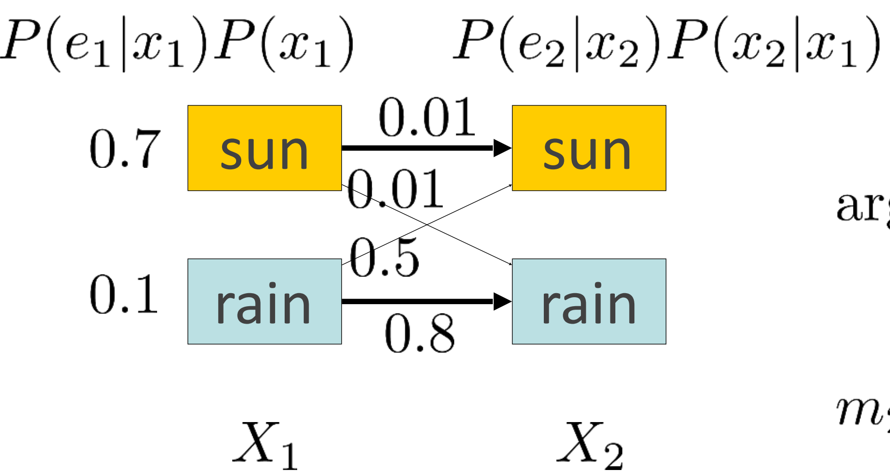
$$m_2[x_2 = \text{rain}] = \max\{0.7 \times 0.01, 0.1 \times 0.8\} = 0.08$$

$$\arg\max_{x_2} m_2[x_2] = \text{rain}$$

$$P(x_1 = \text{sun}, x_2 = \text{rain}, e_1, e_2) = 0.7 \times 0.01 = 0.007$$

$$\text{best path } 1:t \text{ that ends at } x_t \quad \neq \quad \text{best path } 1:N \text{ that goes through } x_t$$

What do We Do?



$\arg\max_{x_1} m_1[x_1] = ?$ sun!

this path starts at rain

$$m_2[x_2] = \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

$$m_2[x_2 = \text{sun}] = \max\{0.7 \times 0.01, 0.1 \times 0.5\} = 0.05$$

$$m_2[x_2 = \text{rain}] = \max\{0.7 \times 0.01, 0.1 \times 0.8\} = 0.08$$

$$\arg\max_{x_2} m_2[x_2] = \text{rain}$$

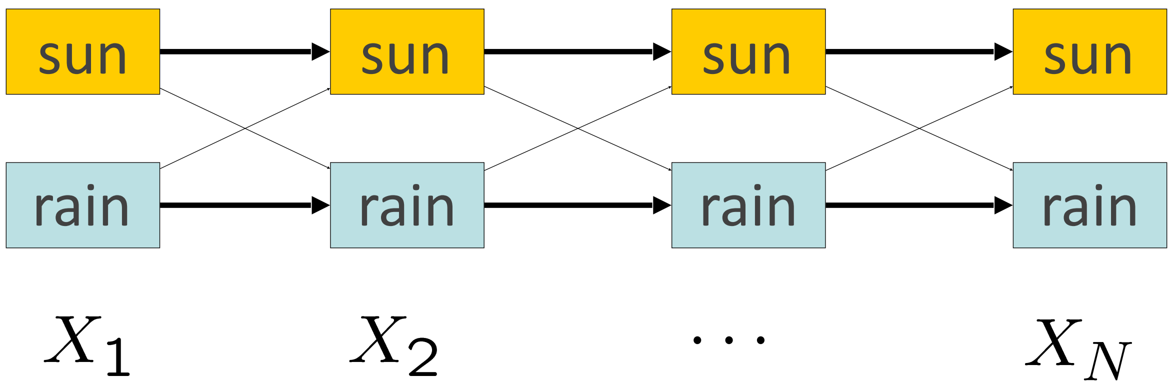
this path starts at rain

idea: what if we also save *where* the best path came from?

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

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

Follow the Breadcrumbs…



for $t = 1$ to N:

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

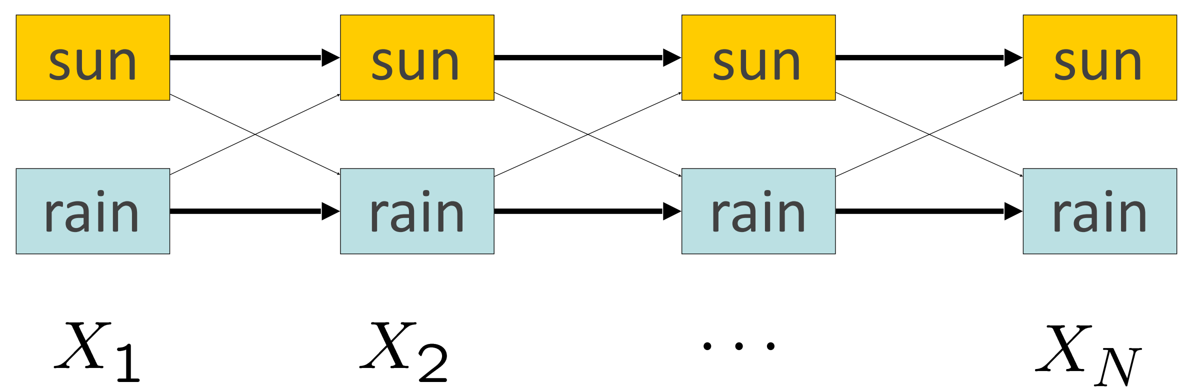
$$a_t[x_t] = \arg\max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

last state on most likely path: $x_N^\star = \arg\max_{x_N} m_N[x_N]$

second to last state on most likely path: $x_{N-1}^\star = a_N[x_N^\star]$

third to last state on most likely path: $x_{N-2}^\star = a_{N-1}[x_{N-1}^\star]$

Follow the Breadcrumbs…



for $t = 1$ to N:

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

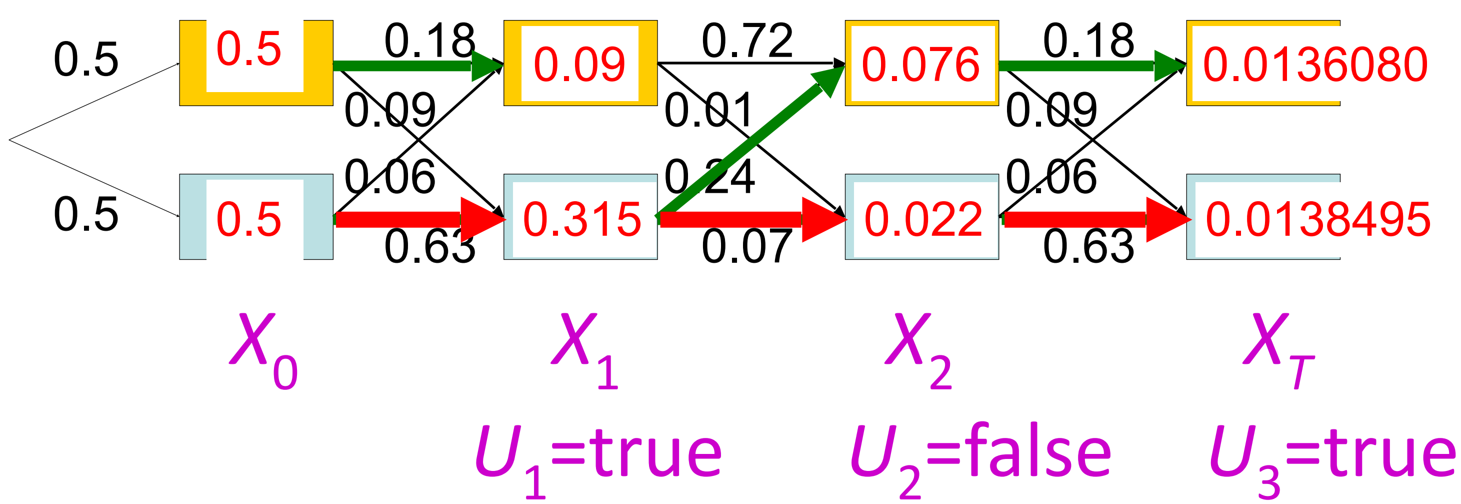
$$a_t[x_t] = \arg\max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

$x^\star_N = \arg\max_{x_N} m_N[x_N]$

for $t = N$ to 2:

$$x^\star_{t-1} = a_t[x^\star_t]$$

Viterbi Algorithm



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

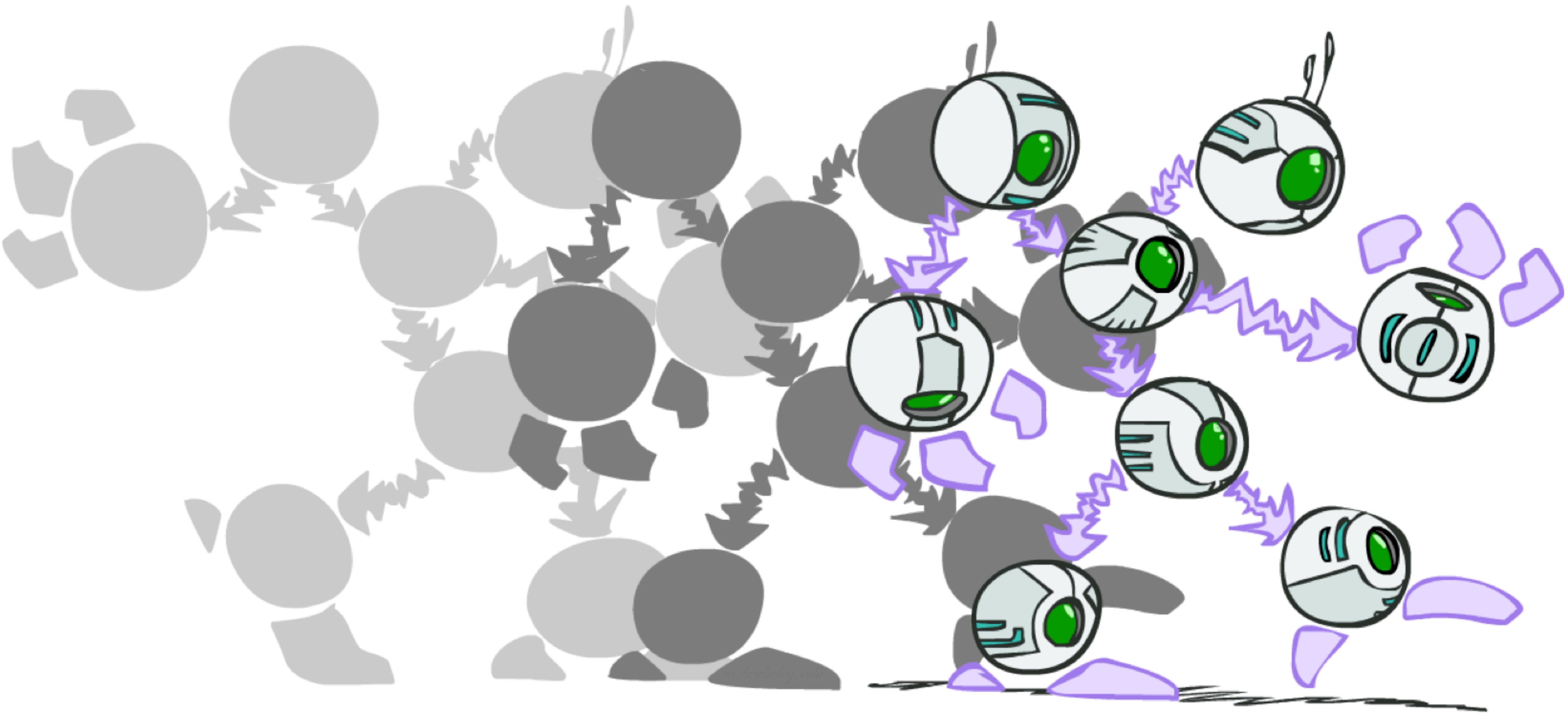
W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

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

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

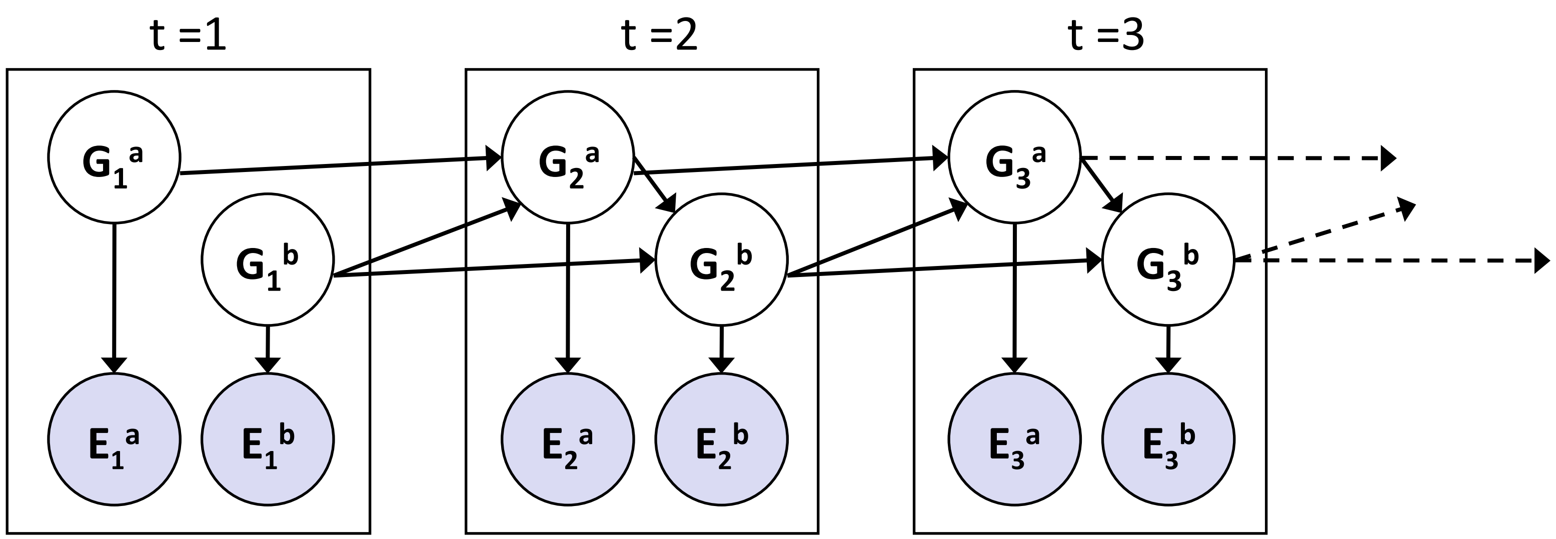
Number of paths?
$O(|X|^T)$

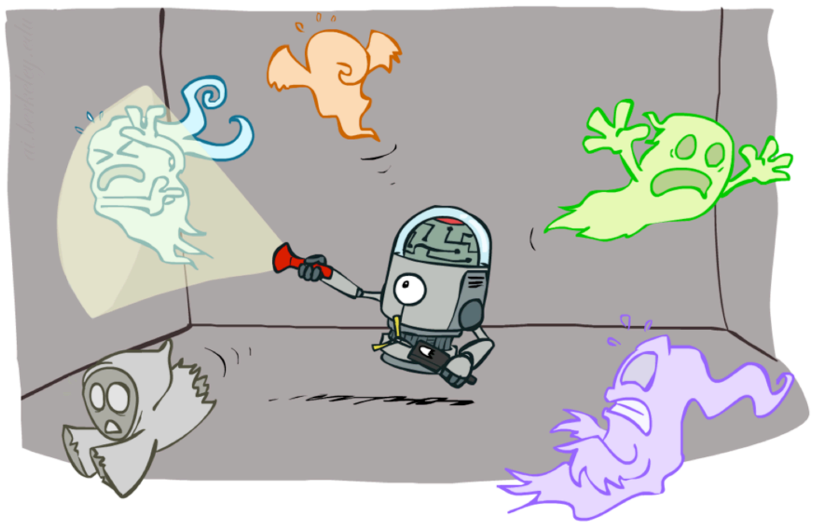
Dynamic Bayes Nets



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*

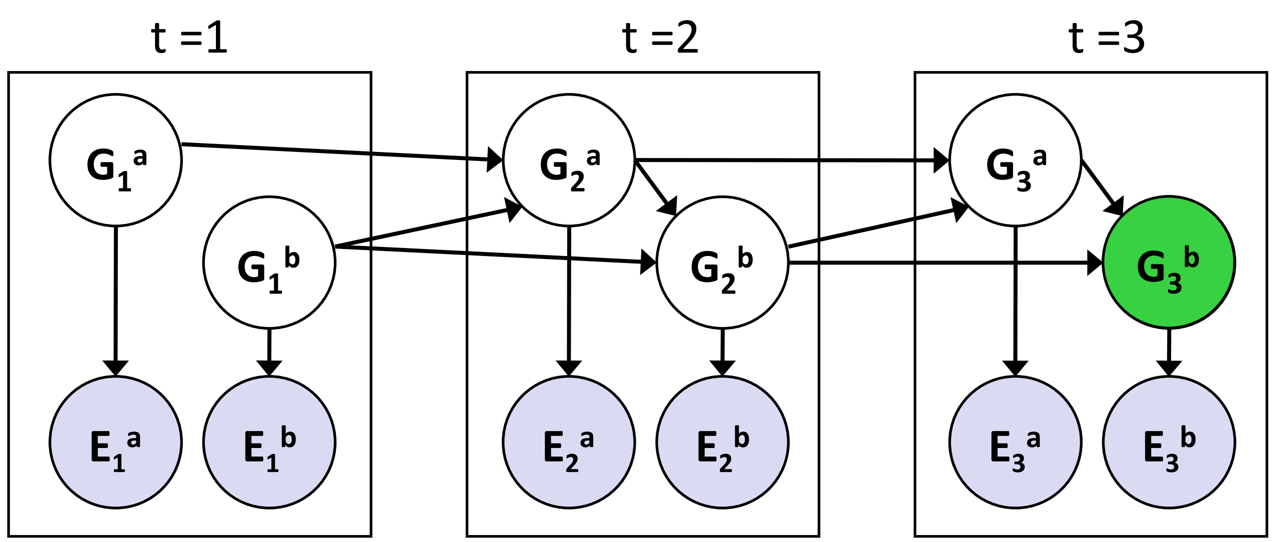




- Dynamic Bayes nets are a generalization of HMMs

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