

Probabilistic Reasoning over Time

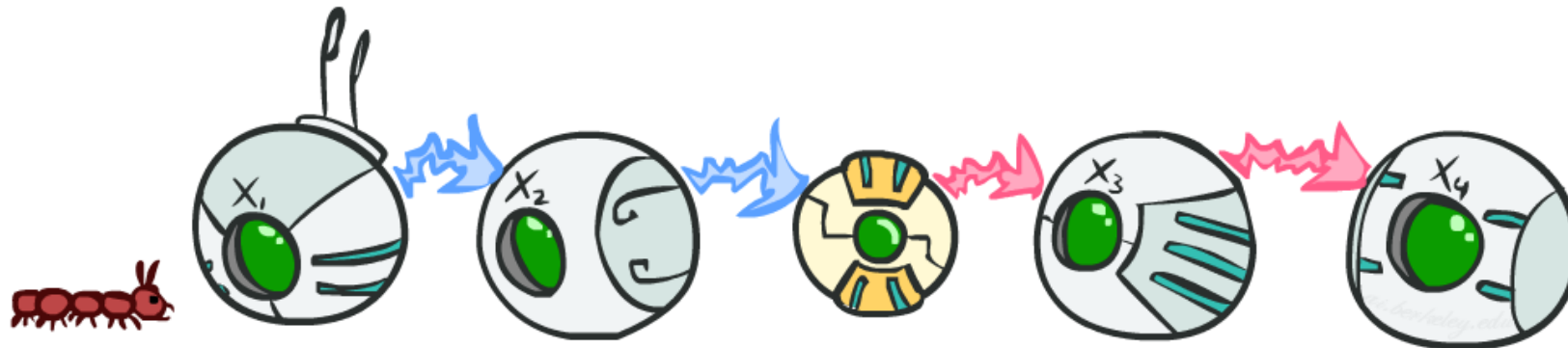


AIMA Chapter 15

Uncertainty and Time

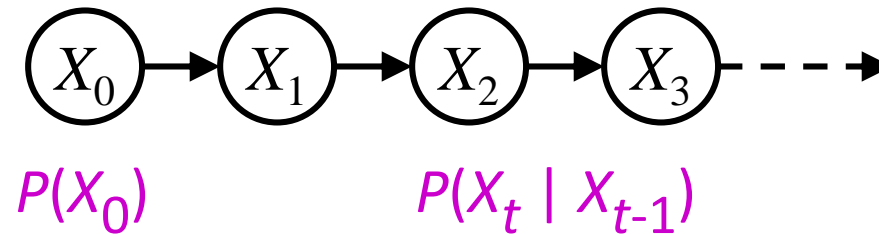
- Often, we want to reason about a *sequence* of observations
 - Speech recognition
 - Robot localization
 - Medical monitoring
 - User attention
- Need to introduce time into our models

Markov Models



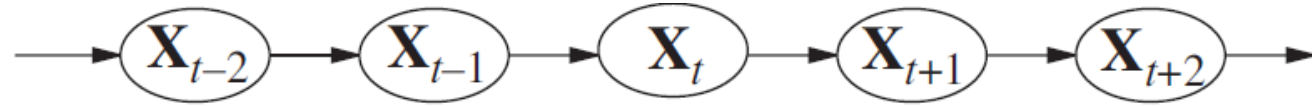
Markov Models (aka Markov chain/process)

- Value of X at a given time is called the **state** (usually discrete, finite)

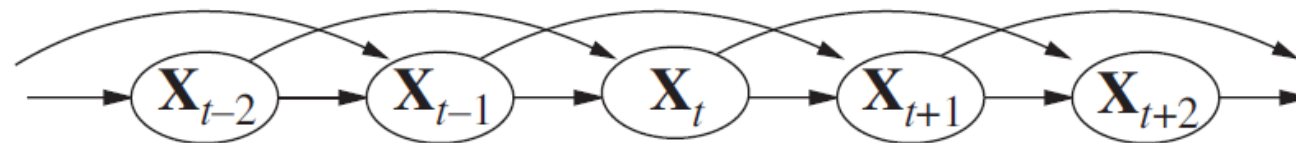


- The **transition model** $P(X_t | X_{t-1})$ specifies how the state evolves over time
- Stationarity** assumption: same transition probabilities at all time steps
- Joint distribution $P(X_0, \dots, X_T) = P(X_0) \prod_t P(X_t | X_{t-1})$

Markov Assumption: Conditional Independence



- **Markov** assumption: X_{t+1}, \dots are independent of X_0, \dots, X_{t-1} given X_t
 - Past and future independent given the present
 - Each time step only depends on the previous
- This is a **first-order** Markov model
- A k th-order model allows dependencies on k earlier steps



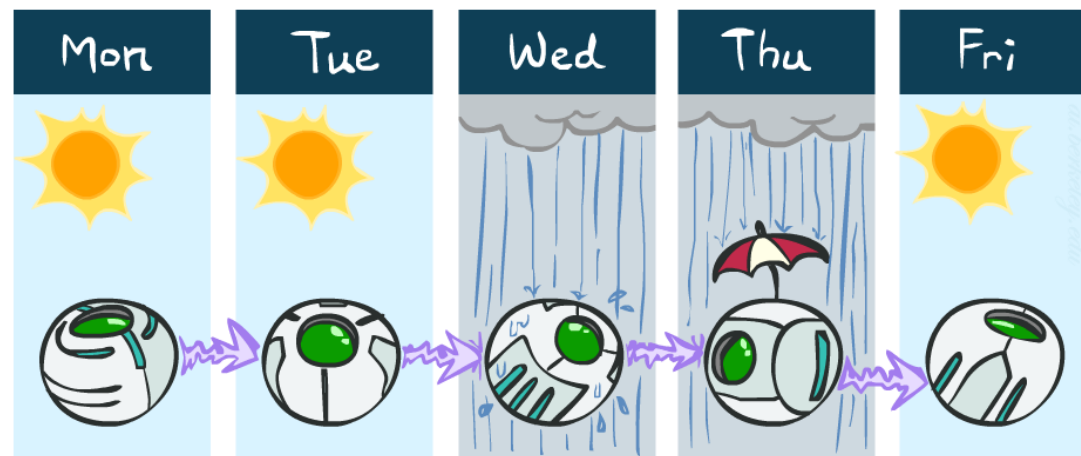
Example: Weather

- States {rain, sun}
- Initial distribution $P(X_0)$

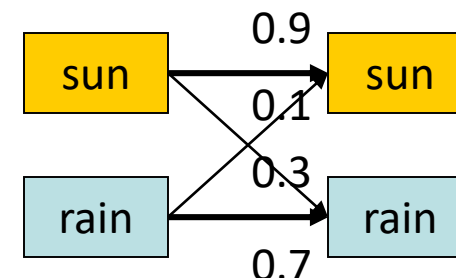
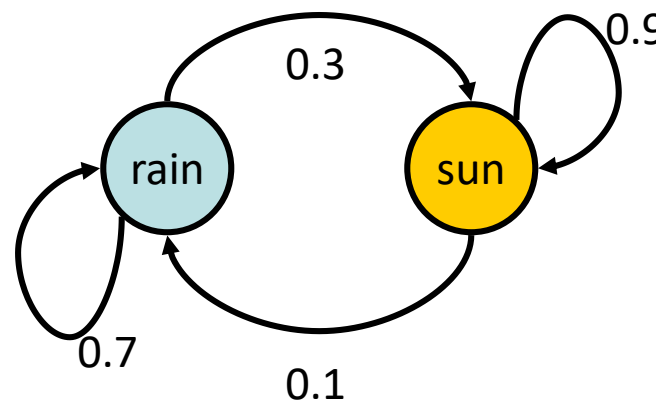
$P(X_0)$	
sun	rain
0.5	0.5

- Transition model $P(X_t | X_{t-1})$

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



Two new ways of representing the same CPT



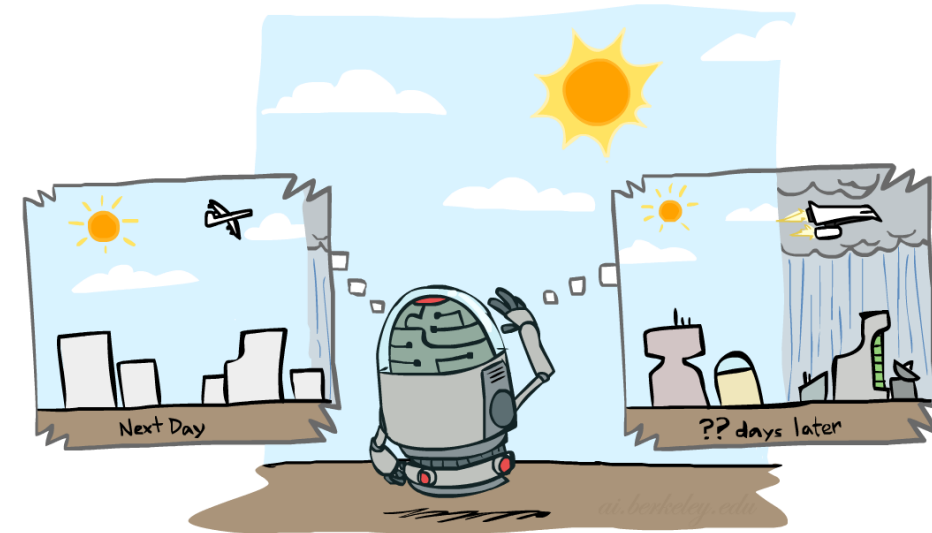
Weather prediction

- Time 0: $\langle 0.5, 0.5 \rangle$

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

- What is the weather like at time 1?

- $P(X_1) = \sum_{x_0} P(X_1, X_0 = x_0)$
- $= \sum_{x_0} P(X_0 = x_0) P(X_1 | X_0 = x_0)$
- $= 0.5 \langle 0.9, 0.1 \rangle + 0.5 \langle 0.3, 0.7 \rangle = \langle 0.6, 0.4 \rangle$



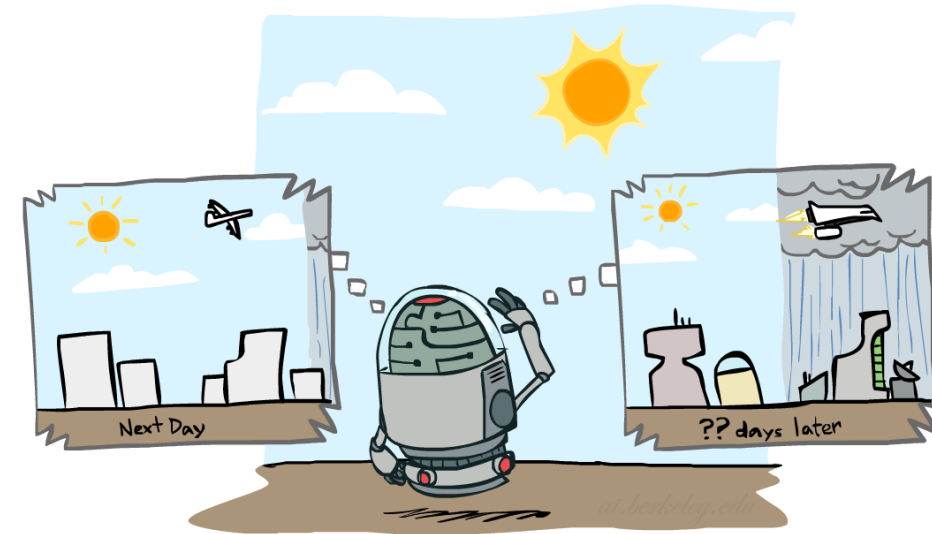
Weather prediction, contd.

- Time 1: $\langle 0.6, 0.4 \rangle$

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

- What is the weather like at time 2?

- $P(X_2) = \sum_{x_1} P(X_2, X_1 = x_1)$
- $= \sum_{x_1} P(X_1 = x_1) P(X_2 | X_1 = x_1)$
- $= 0.6 \langle 0.9, 0.1 \rangle + 0.4 \langle 0.3, 0.7 \rangle = \langle 0.66, 0.34 \rangle$



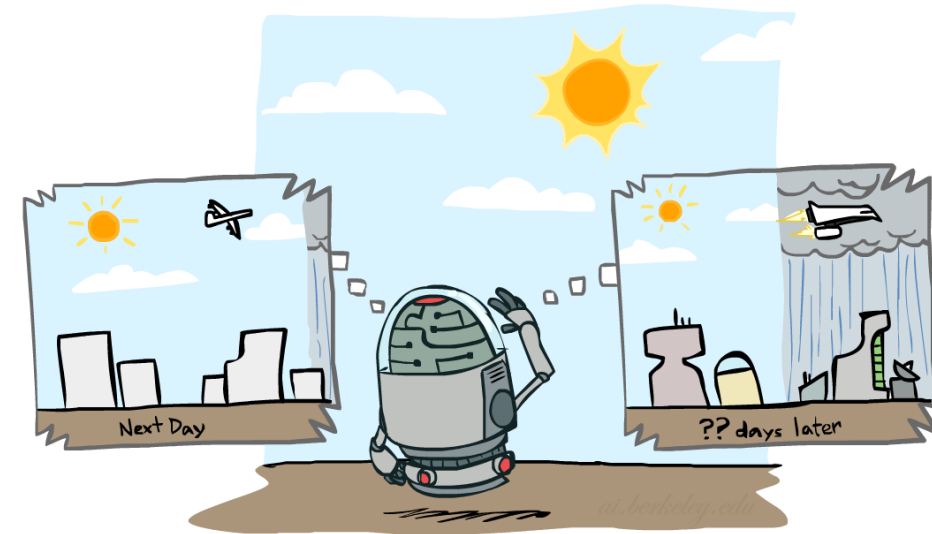
Weather prediction, contd.

- Time 2: $\langle 0.66, 0.34 \rangle$

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

- What is the weather like at time 3?

- $P(X_3) = \sum_{x_2} P(X_3, X_2 = x_2)$
- $= \sum_{x_2} P(X_2 = x_2) P(X_3 | X_2 = x_2)$
- $= 0.66 \langle 0.9, 0.1 \rangle + 0.34 \langle 0.3, 0.7 \rangle = \langle 0.696, 0.304 \rangle$



Forward algorithm (simple form)

- What is the state at time t (given an initial distribution $P(X_0)$)?

- $P(X_t) = \sum_{x_{t-1}} P(X_t, X_{t-1}=x_{t-1})$
- $= \sum_{x_{t-1}} P(X_{t-1}=x_{t-1}) P(X_t | X_{t-1}=x_{t-1})$

Probability from
previous iteration

Transition model

- Iterate this update starting at $t=0$

And the same thing in linear algebra

- What is the weather like at time 2?
 - $P(X_2) = 0.6\langle 0.9, 0.1 \rangle + 0.4\langle 0.3, 0.7 \rangle = \langle 0.66, 0.34 \rangle$

- In matrix-vector form:

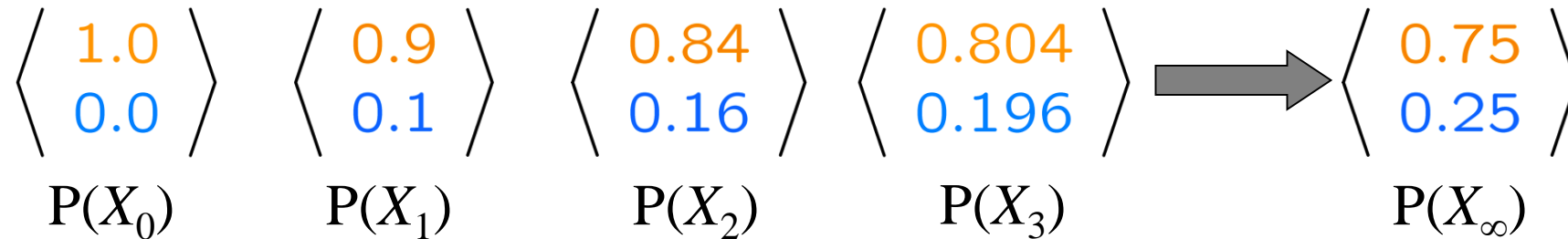
- $P(X_2) = \begin{pmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} = \begin{pmatrix} 0.66 \\ 0.34 \end{pmatrix}$

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

- I.e., multiply by T^T , transpose of transition matrix

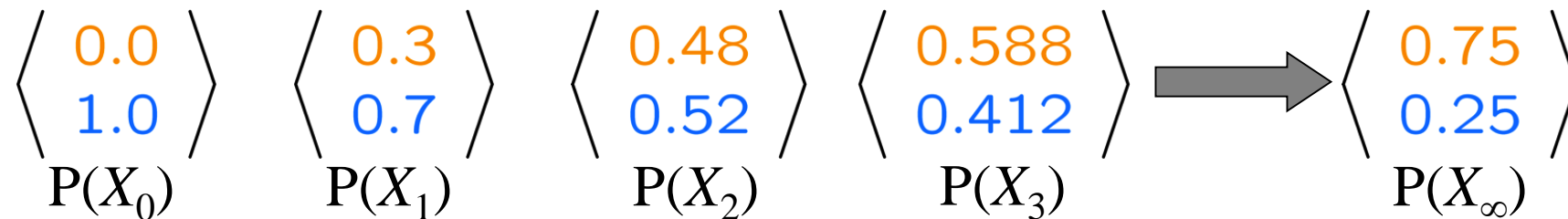
Example Run of Mini-Forward Algorithm

- From initial observation of sun



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

- From initial observation of rain



- From yet another initial distribution $P(X_0)$:



Stationary Distributions

- 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

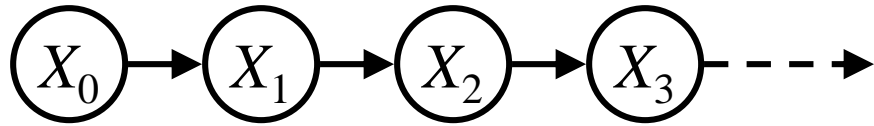
- Stationary distribution:

- 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 Distributions

- Computing the stationary distribution



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

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

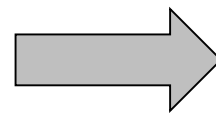
$$P_{\infty}(\text{sun}) = 0.9P_{\infty}(\text{sun}) + 0.3P_{\infty}(\text{rain})$$

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

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

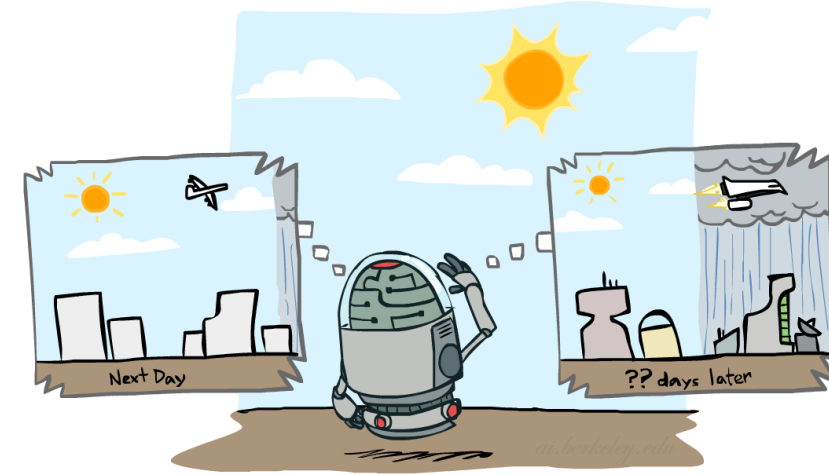
$$P_{\infty}(\text{rain}) = 1/3P_{\infty}(\text{sun})$$

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



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

$$P_{\infty}(\text{rain}) = 1/4$$

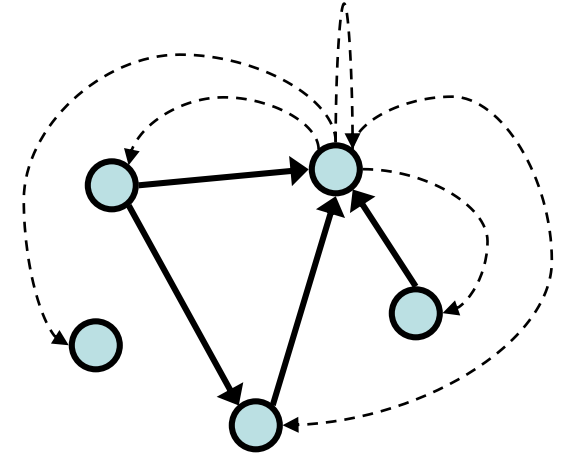


X_{t-1}	X_t	$P(X_t X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

Application of Stationary Distribution: Web Link Analysis

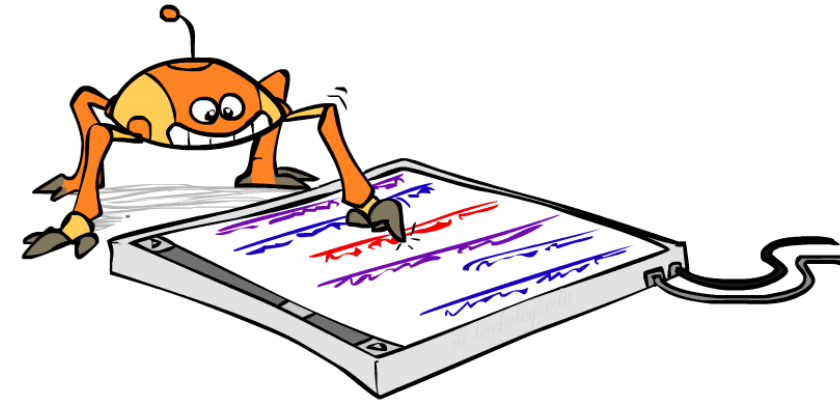
■ Web browsing

- Each web page is a state
- Initial distribution: uniform over pages
- Transitions:
 - With prob. c , uniform jump to a random page
 - With prob. $1-c$, follow a random outlink



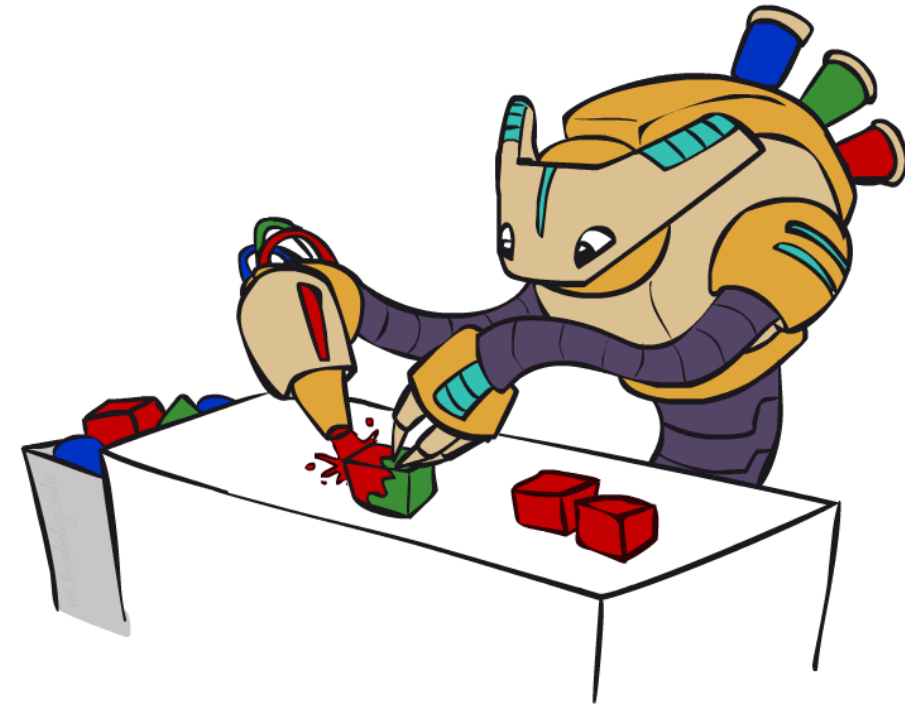
■ Stationary distribution: PageRank

- Will spend more time on highly reachable pages
- Google 1.0 returned the set of pages containing all your keywords in decreasing rank
- Now: use link analysis along with many other factors (rank actually getting less important)



Application of Stationary Distributions: Gibbs Sampling

- Each joint instantiation over all hidden and query variables is a state: $\{X_1, \dots, X_n\} = H \cup Q$
- Transitions:
 - Pick a variable and resample its value conditioned on its Markov blanket
- Stationary distribution:
 - Conditional distribution $P(X_1, X_2, \dots, X_n | e_1, \dots, e_m)$
 - When running Gibbs sampling long enough, we get a sample from the desired distribution

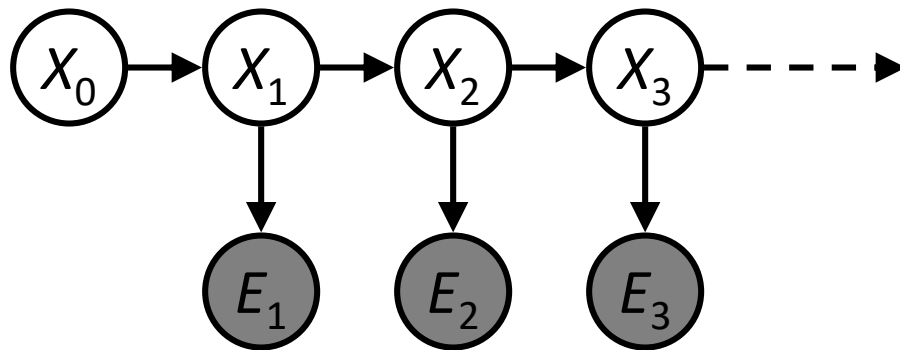


Hidden Markov Models



Hidden Markov Models

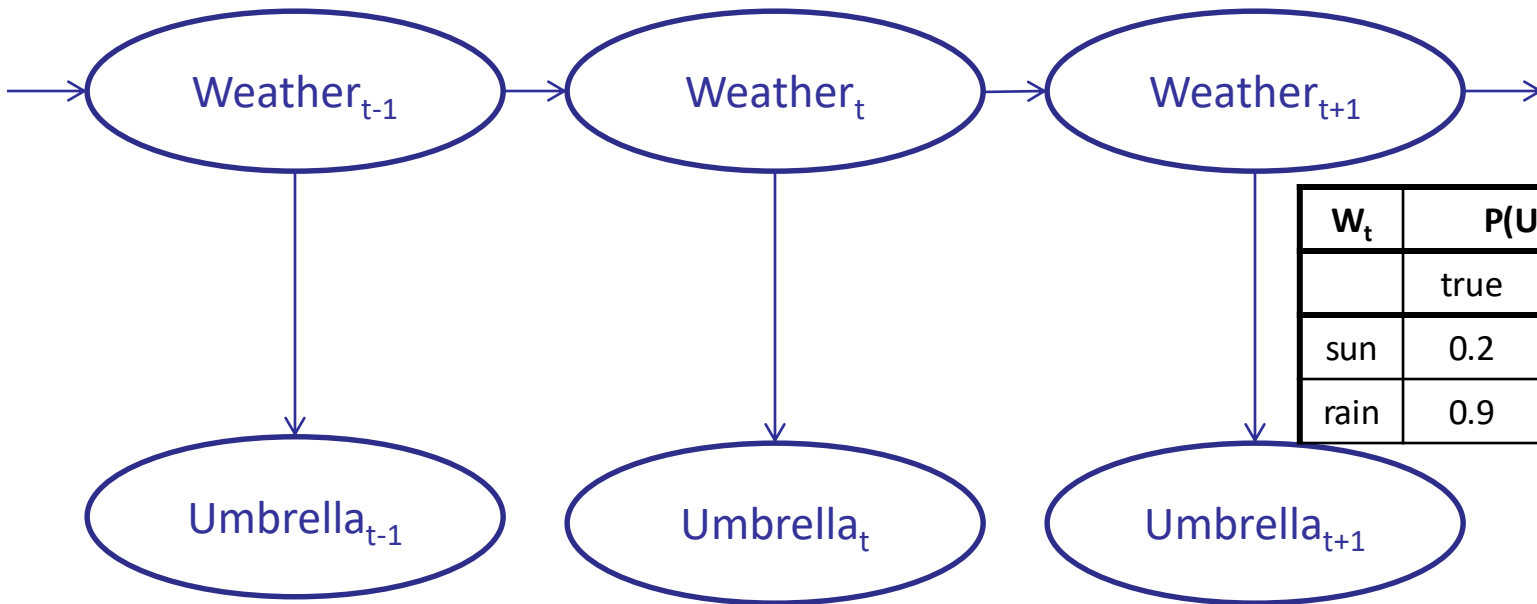
- Usually the true state is not observed directly
 - E.g., you stay indoor and cannot see the weather, but you can see if people come in with umbrella or not.
- Hidden Markov models (HMMs)
 - Underlying Markov chain over states X
 - You observe evidence E at each time step



Example: Weather HMM

- An HMM is defined by:
 - Initial distribution: $P(X_0)$
 - Transition model: $P(X_t | X_{t-1})$
 - Emission model: $P(E_t | X_t)$

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

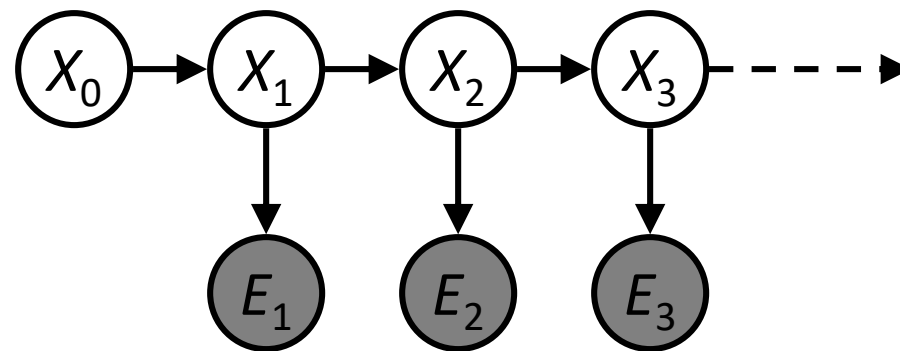


W_t	$P(U_t W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1



HMM as probability model

- Joint distribution for Markov model: $P(X_0, \dots, X_T) = P(X_0) \prod_{t=1:T} P(X_t | X_{t-1})$
- Joint distribution for hidden Markov model:
 $P(X_0, X_1, E_1, \dots, X_T, E_T) = P(X_0) \prod_{t=1:T} P(X_t | X_{t-1}) P(E_t | X_t)$
- Independence in HMM
 - Future states are independent of the past given the present
 - Current evidence is independent of everything else given the current state



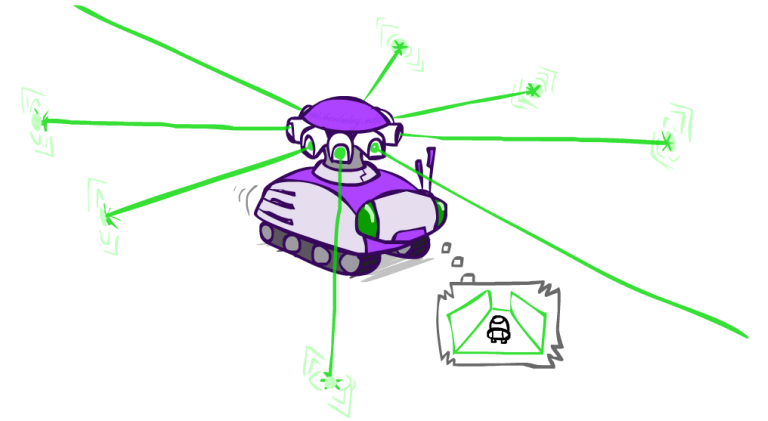
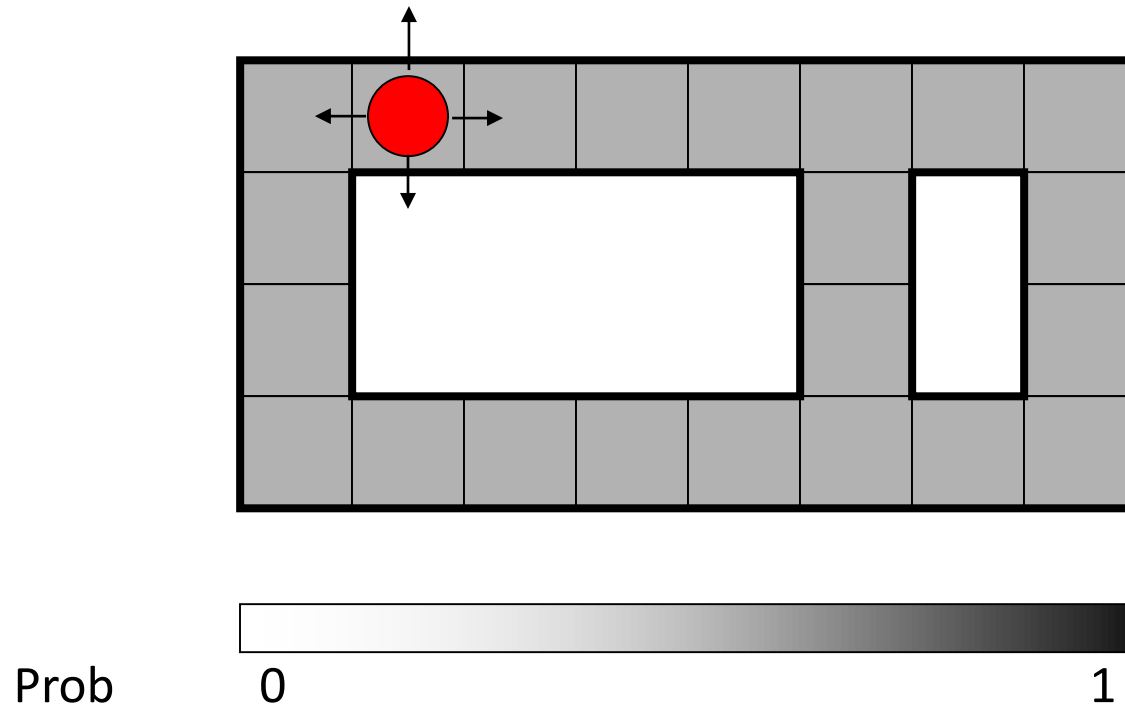
Real HMM Examples

- **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)
 - States are translation options
- **Robot tracking:**
 - Observations are range readings (continuous)
 - States are positions on a map (continuous)
- **Molecular biology:**
 - Observations are nucleotides ACGT
 - States are coding/non-coding/start/stop/splice-site etc.

Inference tasks

- Useful notation: $X_{a:b} = X_a, X_{a+1}, \dots, X_b$
- **Filtering**: $P(X_t | e_{1:t})$
 - **belief state** — posterior distribution over the most recent state given all evidence
- **Prediction**: $P(X_{t+k} | e_{1:t})$ for $k > 0$
 - posterior distribution over a future state given all evidence
- **Smoothing**: $P(X_k | e_{1:t})$ for $0 \leq k < t$
 - posterior distribution over a past state given all evidence
- **Most likely explanation**: $\arg \max_{x_{0:t}} P(x_{0:t} | e_{1:t})$
 - Ex: speech recognition, decoding with a noisy channel

Example: Robot Localization



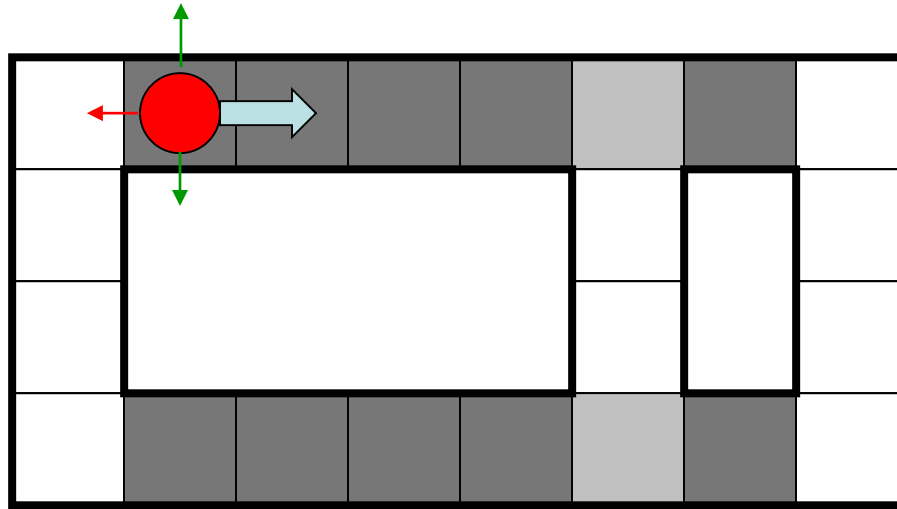
$t=1$

Hidden state: robot location

Sensor model: four bits for wall/no-wall in each direction, never more than 1 mistake

Transition model: action may fail with small prob.

Example: Robot Localization



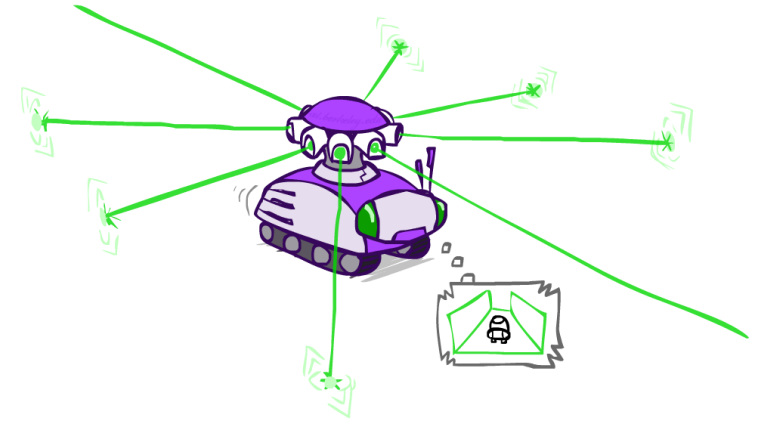
Prob

0

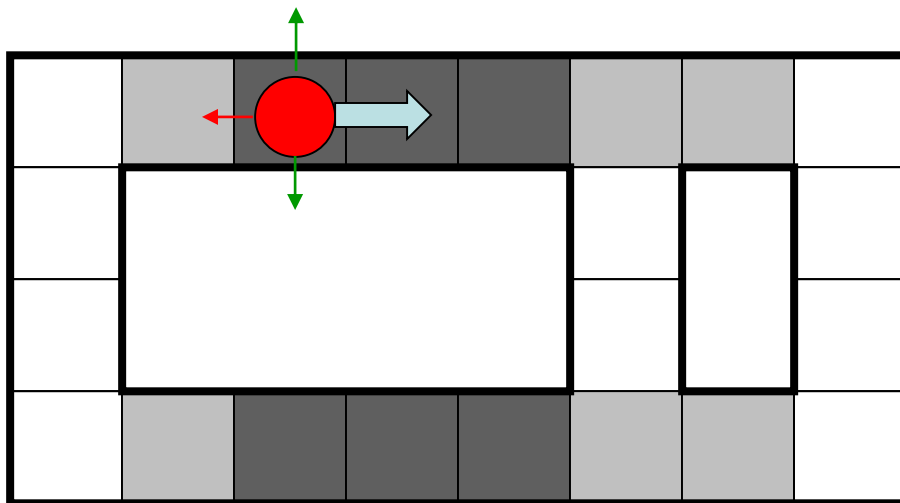
1

t=1

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



Example: Robot Localization



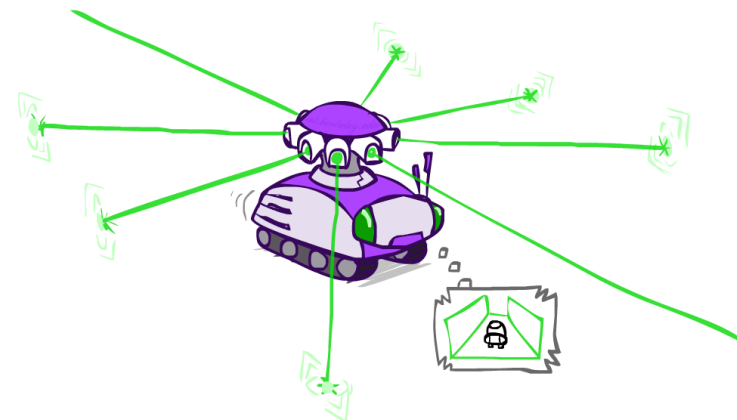
Prob



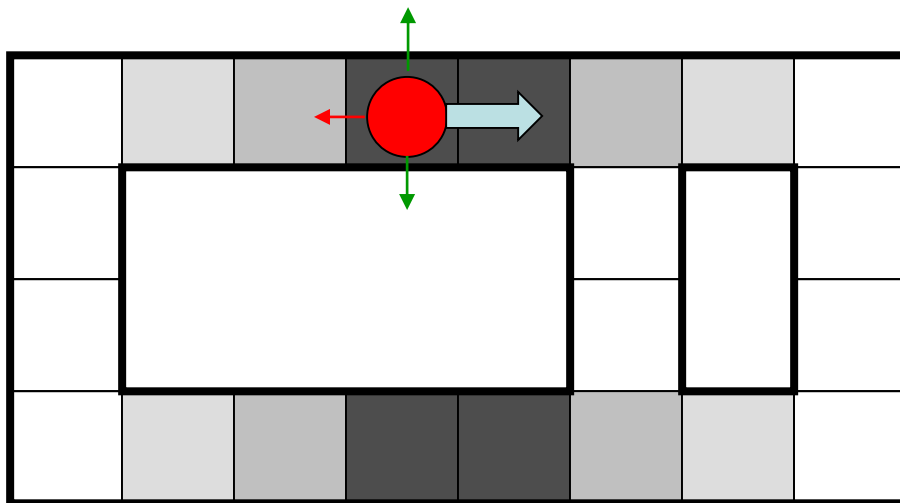
0

1

t=2



Example: Robot Localization



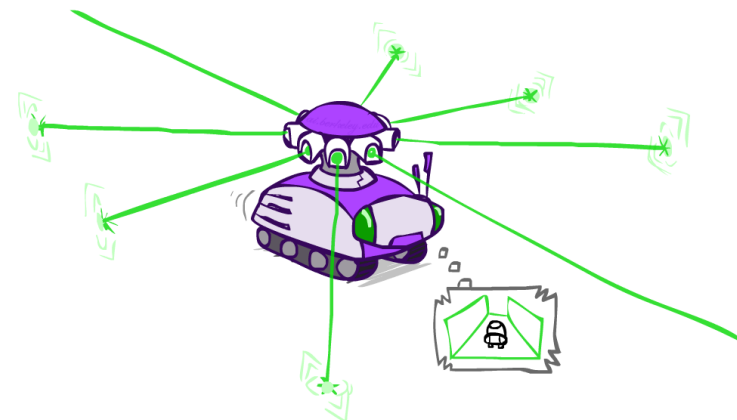
Prob



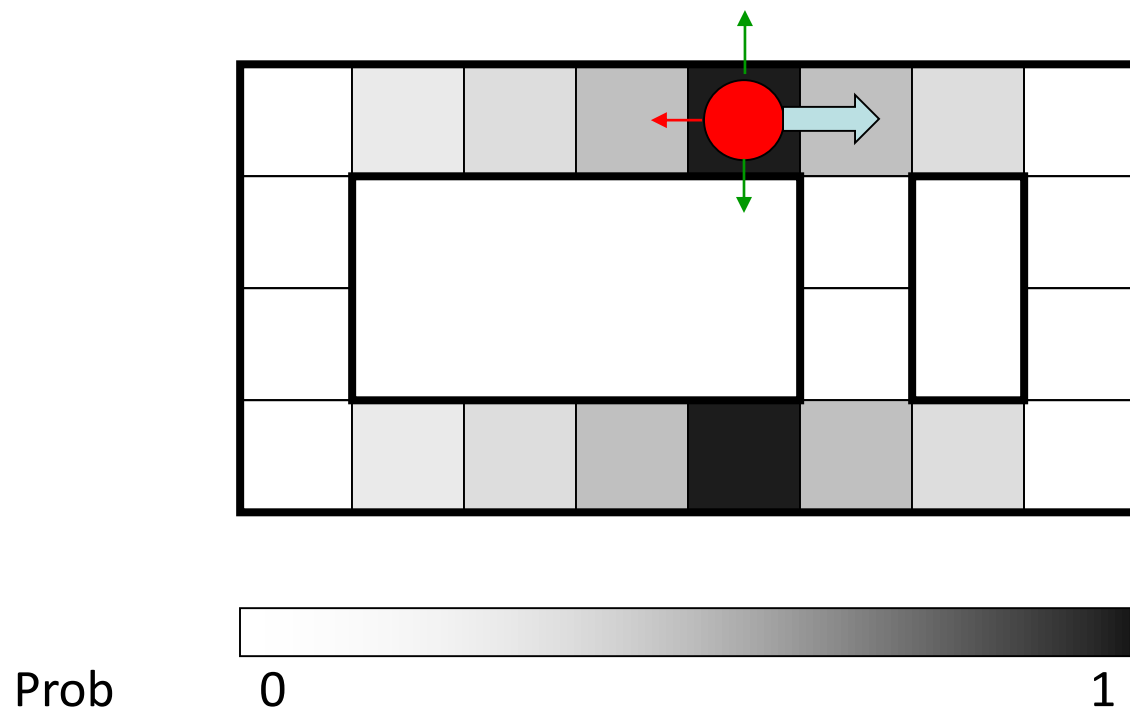
0

1

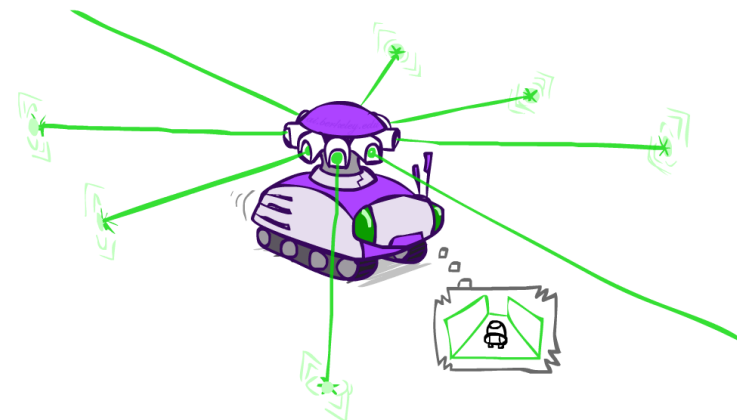
$t=3$



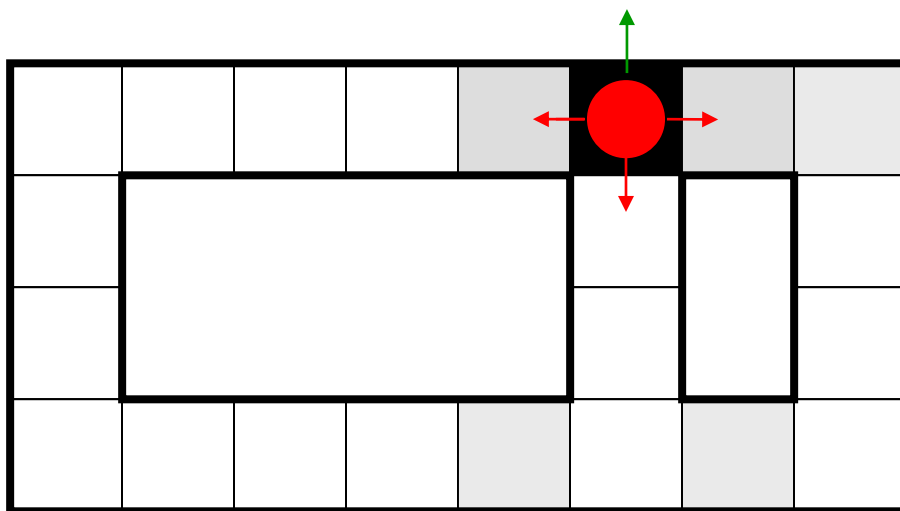
Example: Robot Localization



$t=4$



Example: Robot Localization



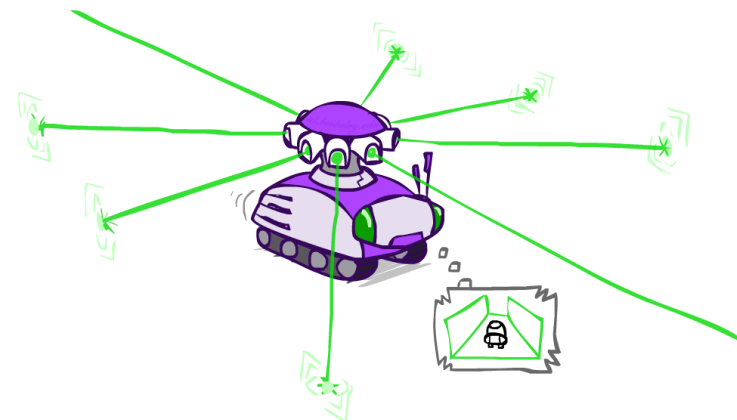
Prob



0

1

$t=5$



Filtering algorithm

- Filtering: infer current state given all evidence
- Aim: a **recursive filtering** algorithm of the form

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

Apply Bayes' rule

- $P(X_{t+1} | e_{1:t+1}) = P(\underline{X_{t+1}} | e_{1:t}, \underline{e_{t+1}})$
- $= \alpha P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t})$

$$\alpha = 1 / P(e_{t+1} | e_{1:t})$$

Filtering algorithm

- Filtering: infer current state given all evidence
- Aim: a **recursive filtering** algorithm of the form
 - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$

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

Apply conditional independence

Normalize

Update

Predict

Filtering algorithm

- Filtering: infer current state given all evidence
- Aim: a **recursive filtering** algorithm of the form

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

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

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

- $= \alpha P(e_{t+1} | X_{t+1}) \underbrace{P(X_{t+1} | e_{1:t})}_{\text{Condition on } X_t}$

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

Filtering algorithm

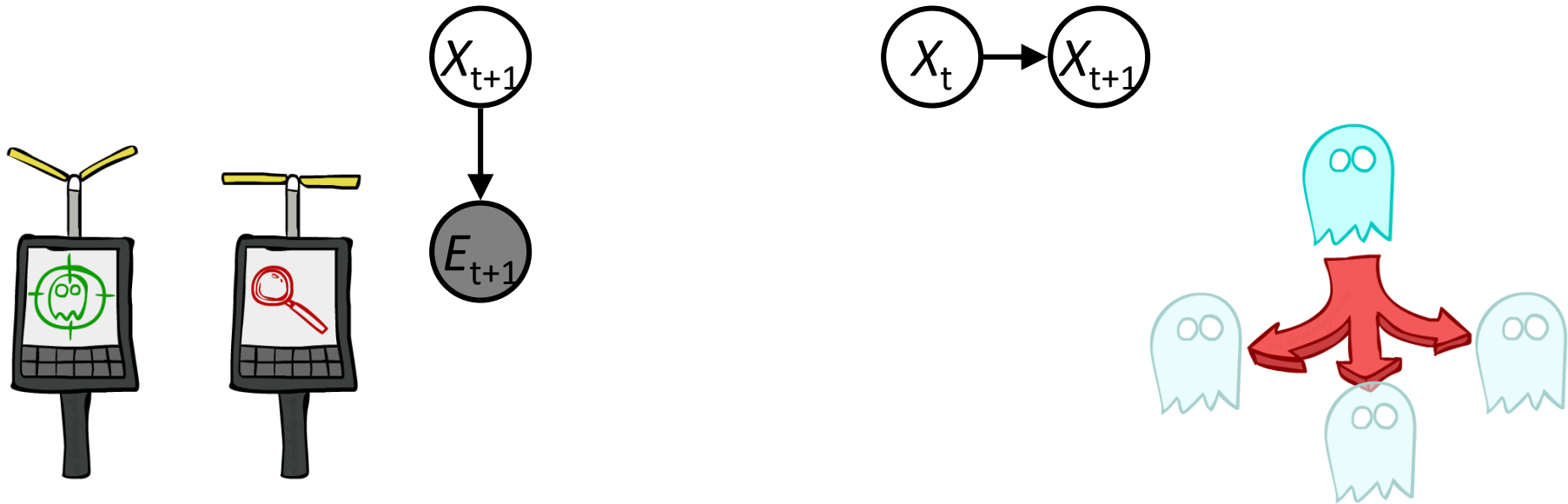
- Filtering: infer current state given all evidence
- Aim: a **recursive filtering** algorithm of the form
 - $P(X_{t+1} | e_{1:t+1}) = g(e_{t+1}, P(X_t | e_{1:t}))$

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

Apply conditional independence

Filtering algorithm

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



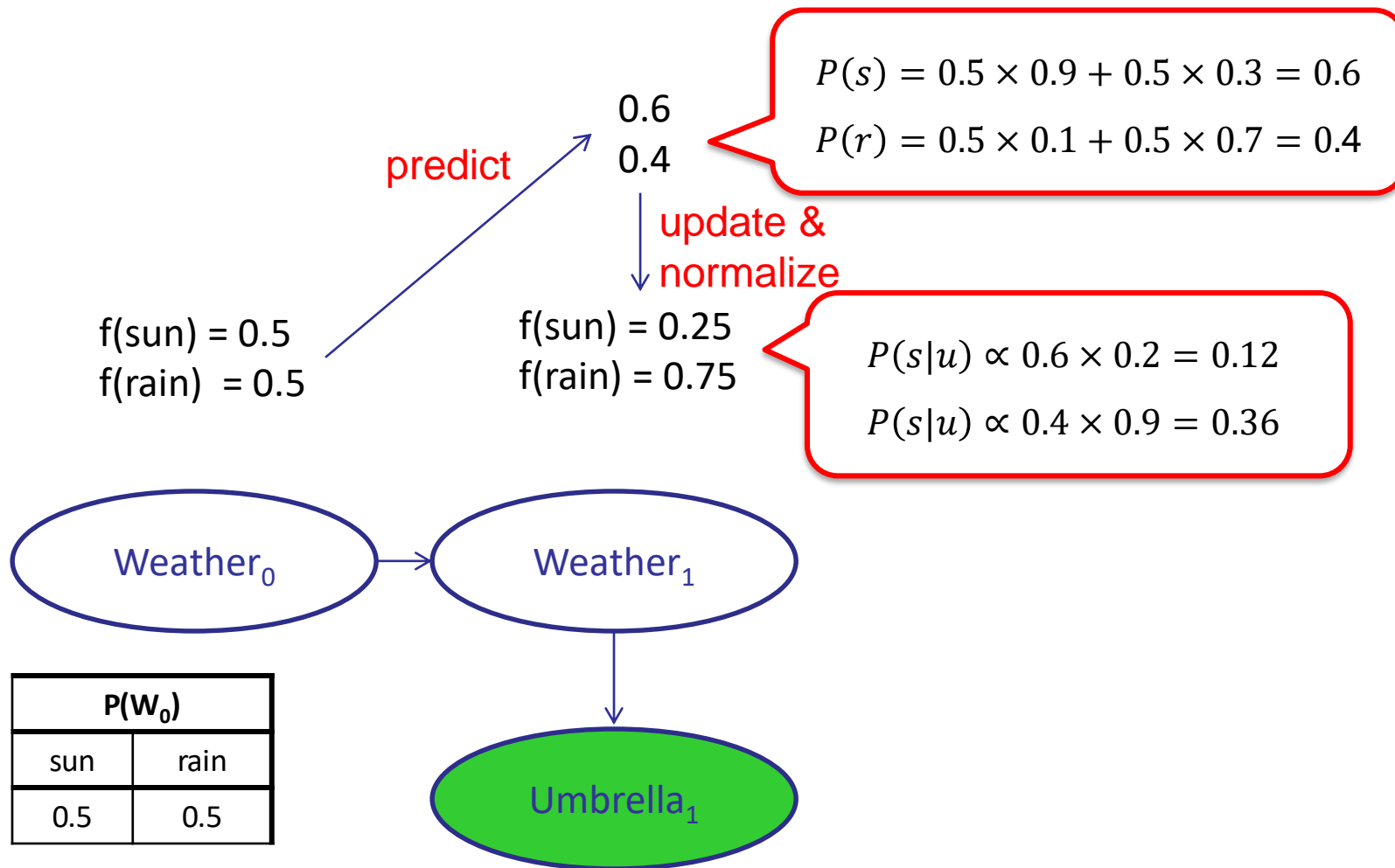
Filtering algorithm

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



- $f_{1:t+1} = \text{FORWARD}(f_{1:t}, e_{t+1})$
- We start with $f_{1:0} = P(X_0)$ and then iterate
- Cost per time step: $O(|X|^2)$ where $|X|$ is the number of states

Example: Weather HMM



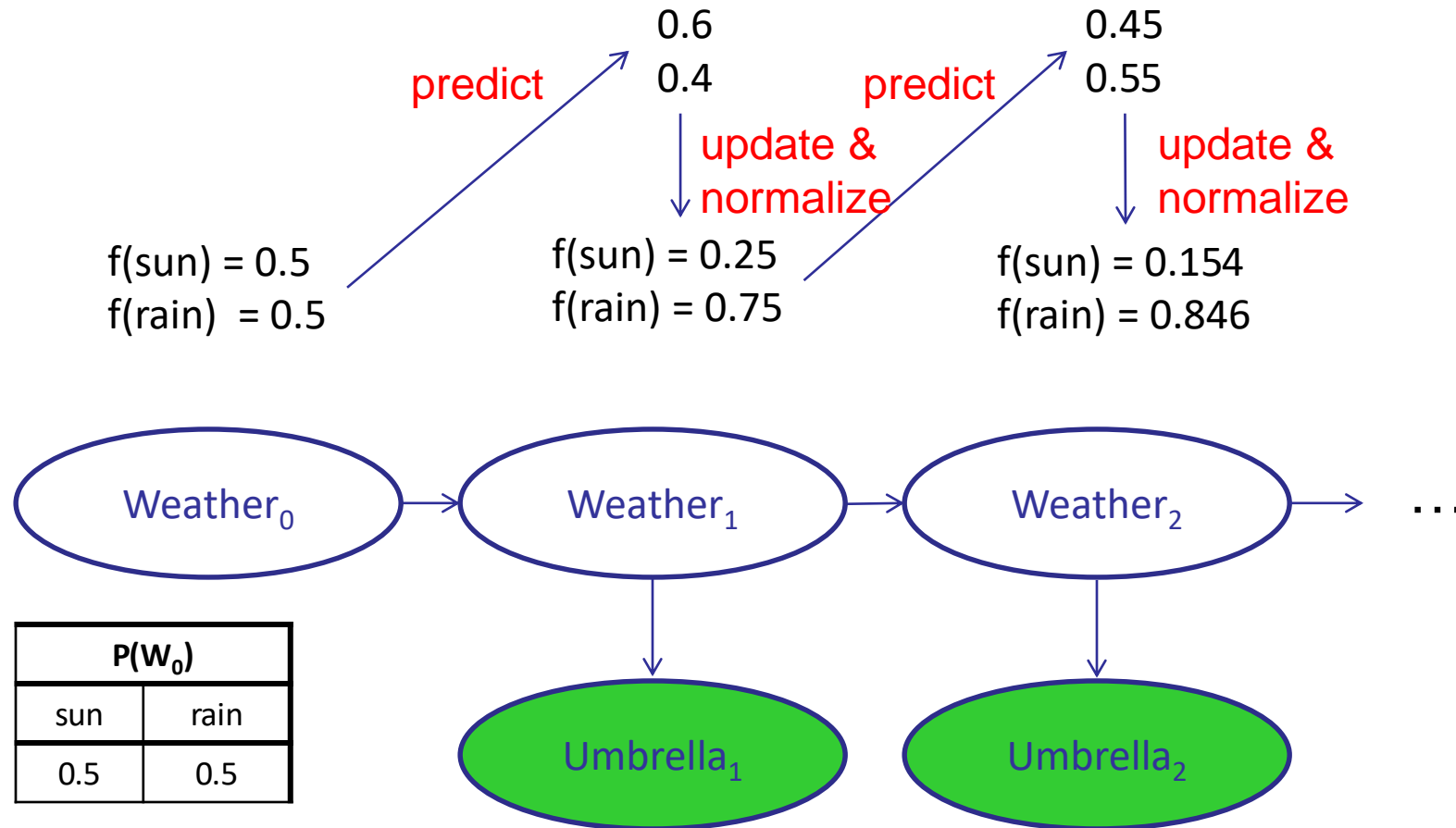
$$P(X_{t+1} | e_{1:t+1}) = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t)$$



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

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

Example: Weather HMM



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

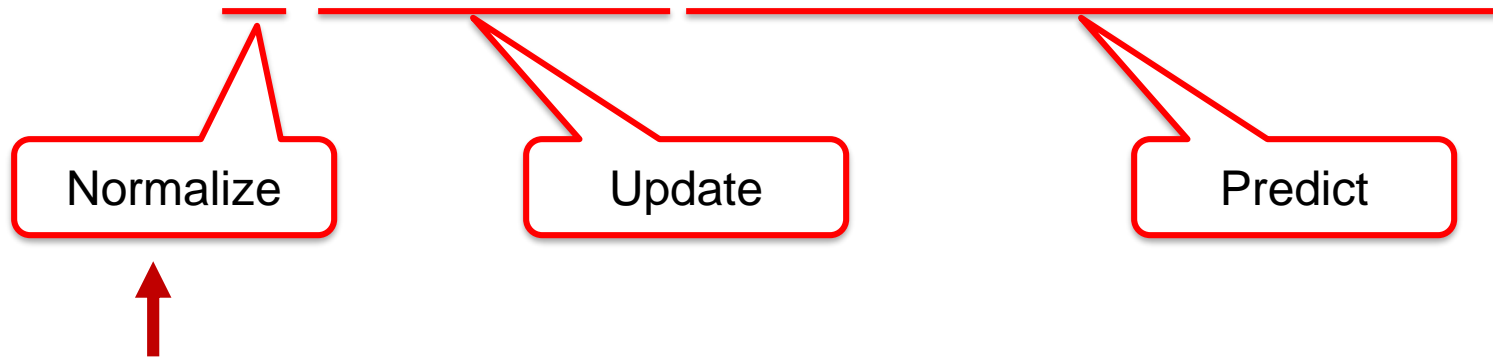


W_{t-1}	$P(W_t W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

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

Filtering algorithm

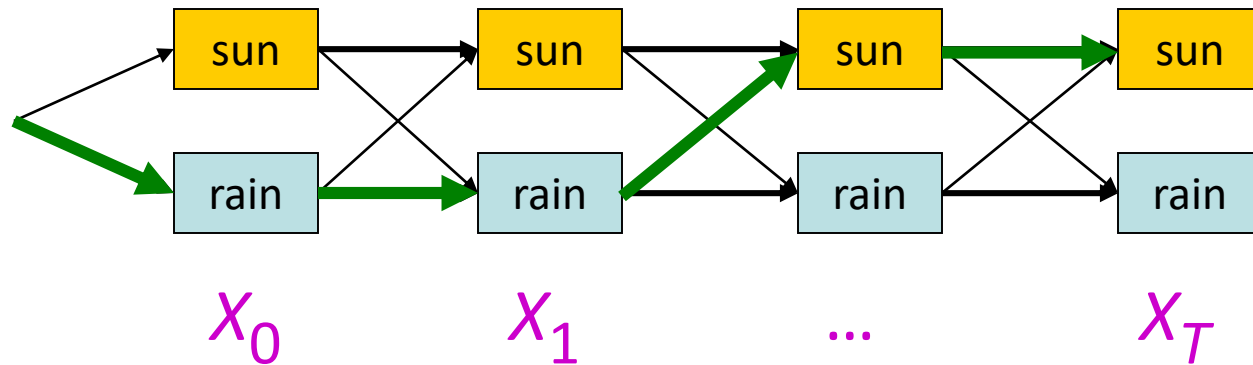
- $P(X_{t+1} | e_{1:t+1}) = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t)$



α is a constant. So if we only want to compute $P(x_t | e_{1:t})$, then we can skip normalization when computing $P(x_1 | e_1)$, $P(x_2 | e_{1:2})$, ..., $P(x_{t-1} | e_{1:t-1})$

Another view of the algorithm

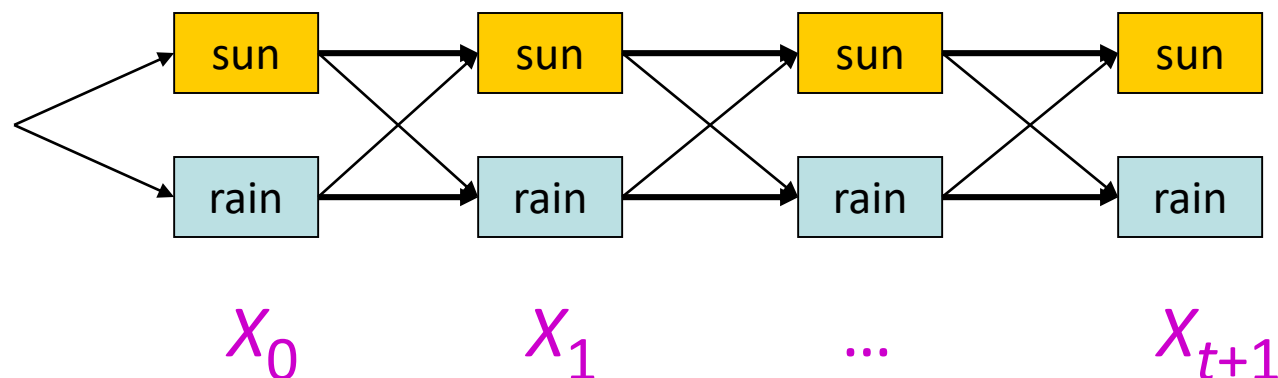
- **State trellis**: graph of states and transitions over time



- 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 $P(x_0)$)
- Each path is a sequence of states
- The **product** of weights on a path is proportional to that state sequence's probability

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

Another view of the algorithm



- Forward algorithm computes sum over all possible paths

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

- It uses dynamic programming to sum over all paths
 - For each state at time t, keep track of the total probability of all paths to it

$$\begin{aligned} f_{1:t+1} &= \text{FORWARD}(f_{1:t}, e_{t+1}) \\ &= \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) f_{1:t}[x_t] \end{aligned}$$

Most Likely Explanation

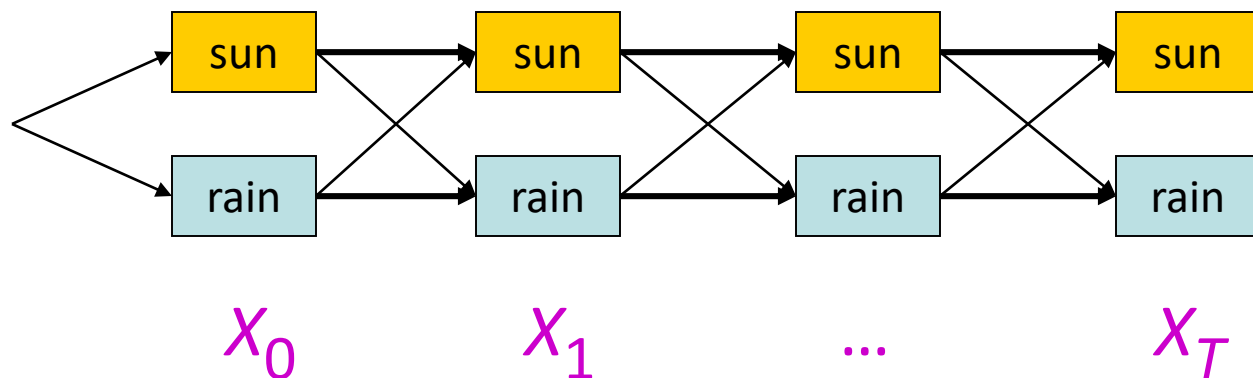


Inference tasks

- **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 $0 \leq k < t$
 - better estimate of past states, essential for learning
- **Most likely explanation**: $\arg \max_{x_{0:t}} P(x_{0:t} | e_{1:t})$
 - speech recognition, decoding with a noisy channel

Most likely explanation = most probable path

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



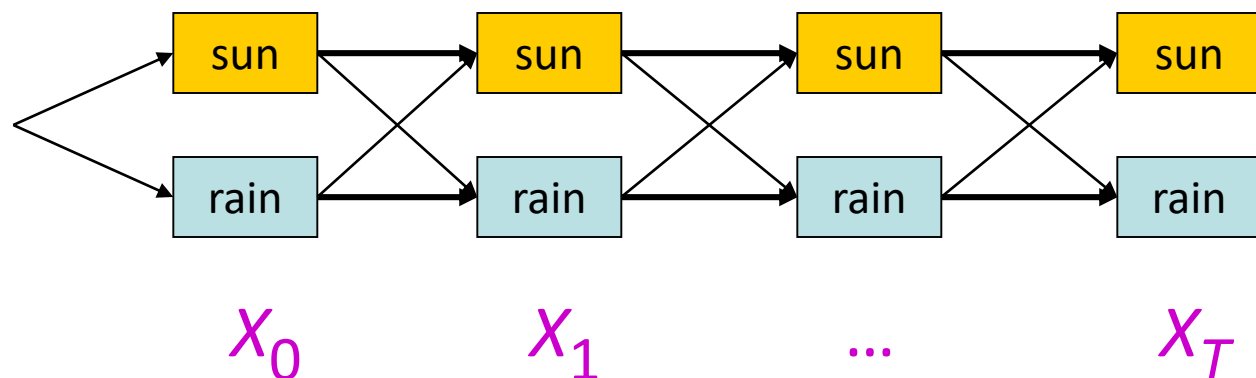
- The **product** of weights on a path is proportional to that state sequence's probability

$$P(x_0) \prod_t P(x_t | x_{t-1}) P(e_t | x_t) = P(x_{0:t}, e_{1:t}) \propto P(x_{0:t} | e_{1:t})$$

- **Viterbi algorithm** computes best paths

$$\arg \max_{x_{0:t}} P(x_{0:t} | e_{1:t})$$

Forward / Viterbi algorithms



Viterbi Algorithm (max)

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

$$\begin{aligned} m_{1:t+1} &= \text{VITERBI}(m_{1:t}, e_{t+1}) \\ &= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}|x_t) m_{1:t}[x_t] \end{aligned}$$

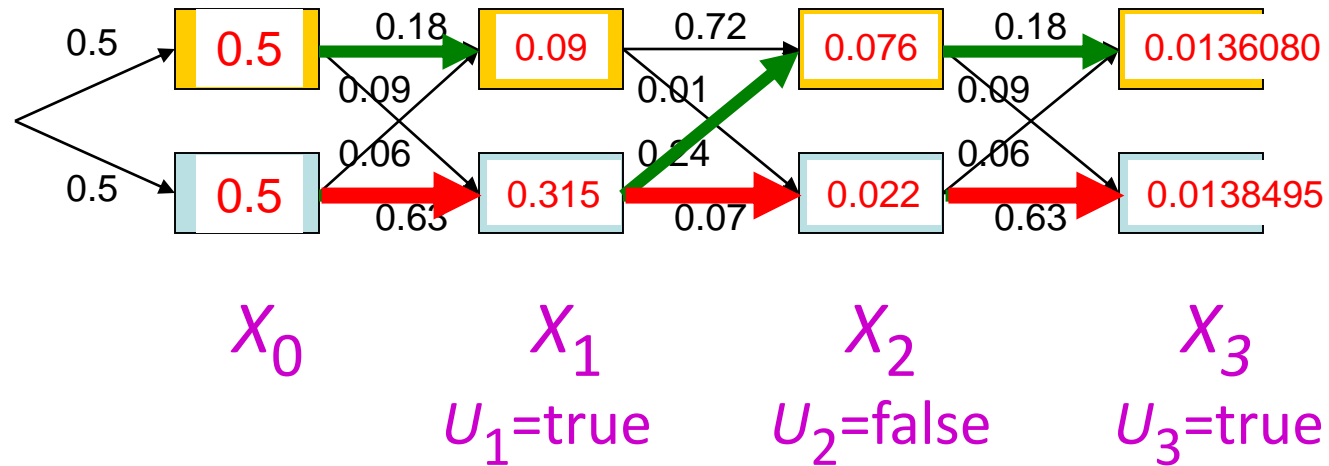
Forward Algorithm (sum)

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

$$\begin{aligned} f_{1:t+1} &= \text{FORWARD}(f_{1:t}, e_{t+1}) \\ &= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) f_{1:t}[x_t] \end{aligned}$$

Viterbi algorithm contd.

P(W ₀)	
sun	rain
0.5	0.5



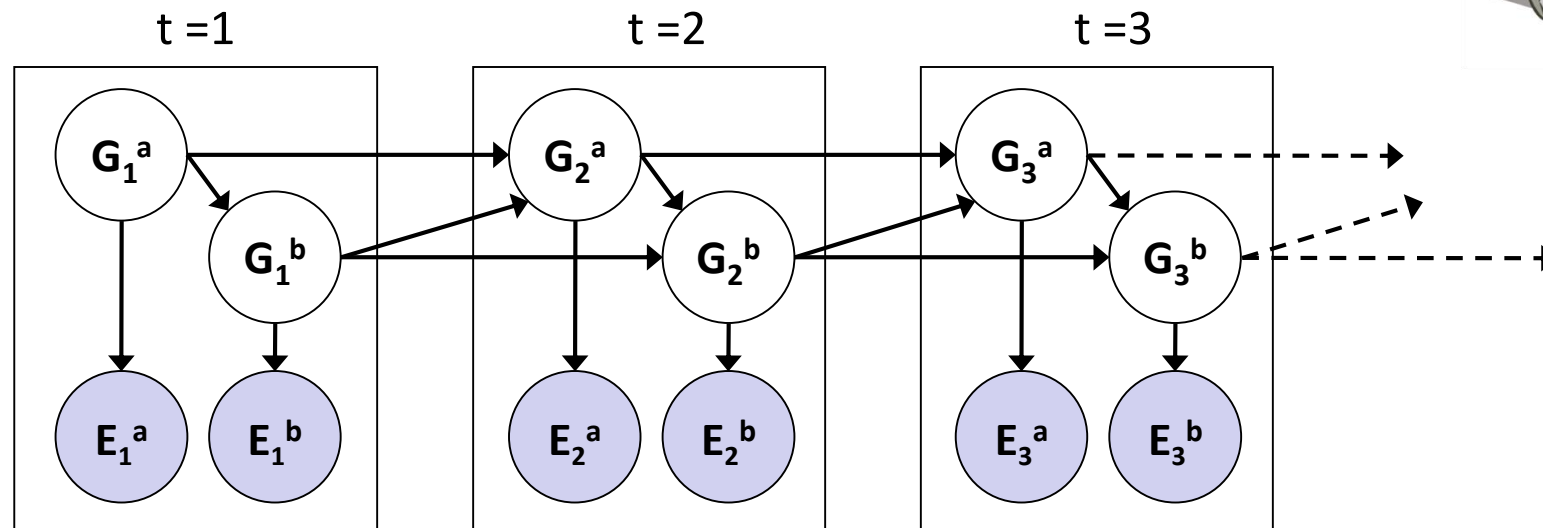
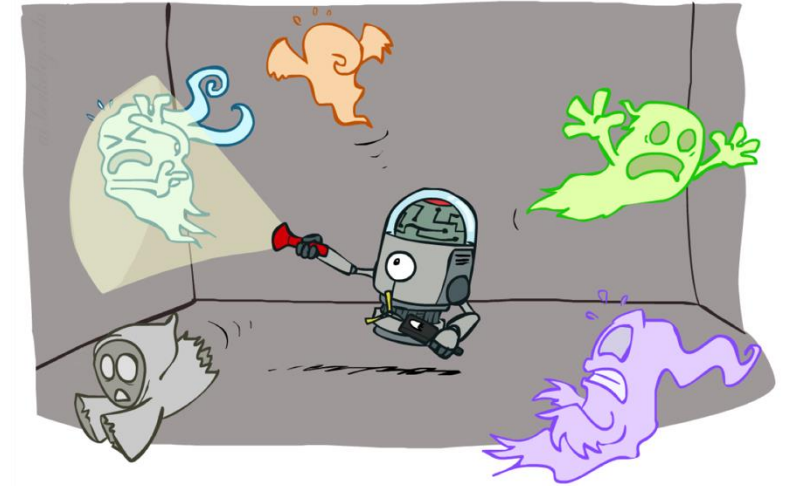
W _{t-1}	P(W _t W _{t-1})	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

W _t	P(U _t W _t)	
	true	false
sun	0.2	0.8
rain	0.9	0.1

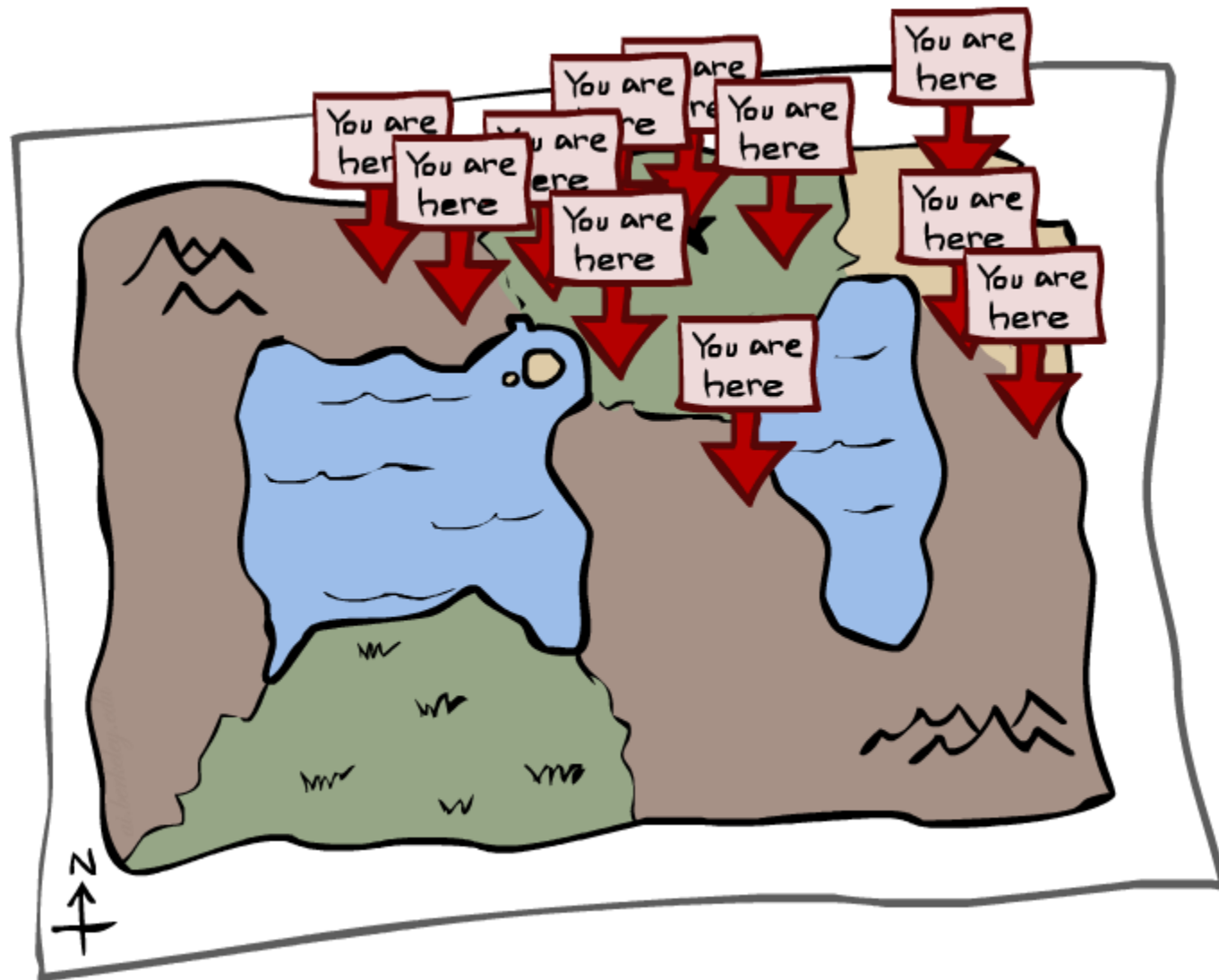
- $m_{1:t+1} = P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}|x_t) m_{1:t}[x_t]$
- Time complexity: $O(|X|^2 T)$
- Space complexity: $O(|X| T)$

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 $t-1$

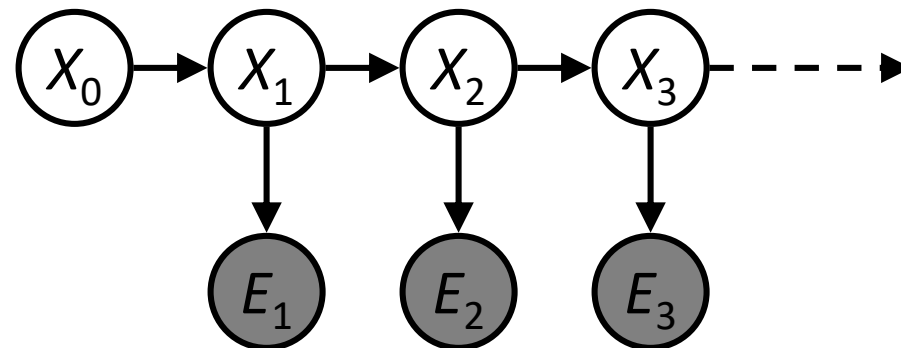


Particle Filtering



Large state space

- When $|X|$ is huge (e.g., position in a building), exact inference becomes infeasible
- Can we use approximate inference, e.g., likelihood weighting?
 - Evidences are “downstream”
 - By ignoring the evidence: with more states sampled over time, the weight drops quickly (going into low-probability region)
 - Hence: too few “reasonable” samples



Particle Filtering

- Represent belief state at each step by a set of samples
 - Samples are called *particles*
- Our representation of $P(X)$ is now a list of N particles (samples)
 - $P(x)$ approximated by number of particles with value x
 - So, many x may have $P(x) = 0$
 - Generally, $N \ll |X|$
 - More particles, more accuracy; but a large N would defeat the point.

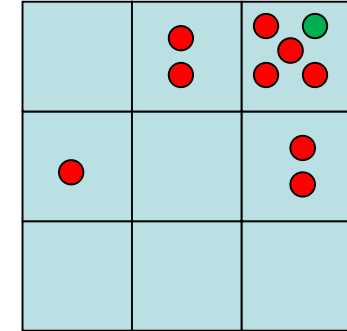
0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5



	●	
		● ●
	● ●	● ● ● ●

Representation: Particles

- At first, all particles have a weight of 1



Particles:

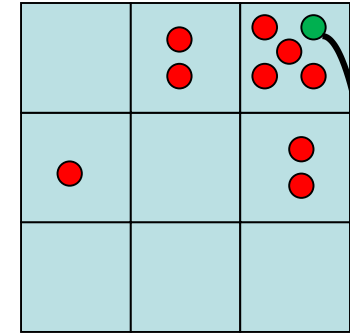
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Propagate forward

- Each particle is moved by sampling its next position from the transition model:
 - $x_{t+1} \sim P(X_{t+1} | x_t)$
- This captures the passage of time
 - If enough samples, close to exact probabilities (consistent)

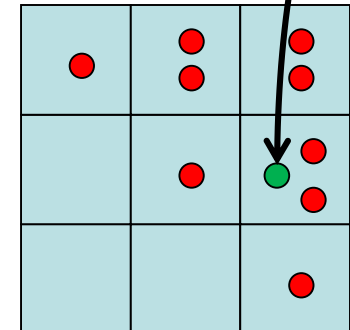
Particles:

(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)

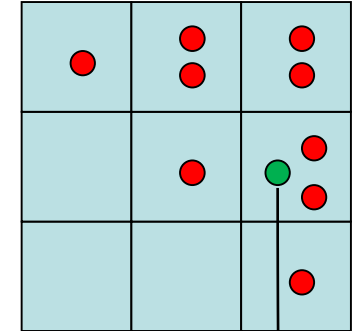


Particle Filtering: Observe

- Similar to likelihood weighting, weight samples based on the evidence
 - $W = P(e_t | x_t)$
 - Particles that fit the evidence better get higher weights, others get lower weights
- What happens if we repeat the Propagate-Observe procedure over time?
 - It is exactly likelihood weighting (if we multiply the weights)
 - Weights drop quickly...

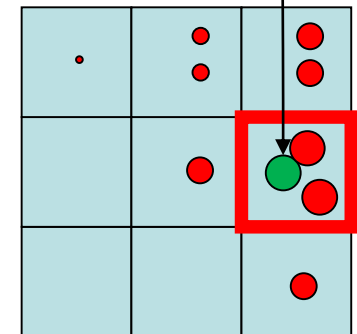
Particles:

(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

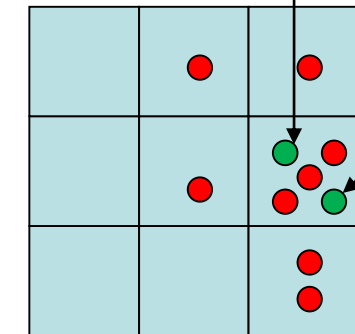
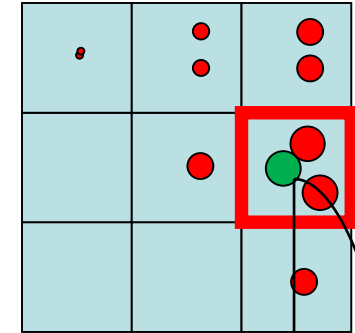
- Rather than tracking weighted samples, we *resample*
 - Generate N new samples from our weighted sample distribution
 - Each new sample is selected from the current population of samples; the probability is proportional to its weight.
 - The new samples have weight of 1
- Now the update is complete 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) $w=.4$

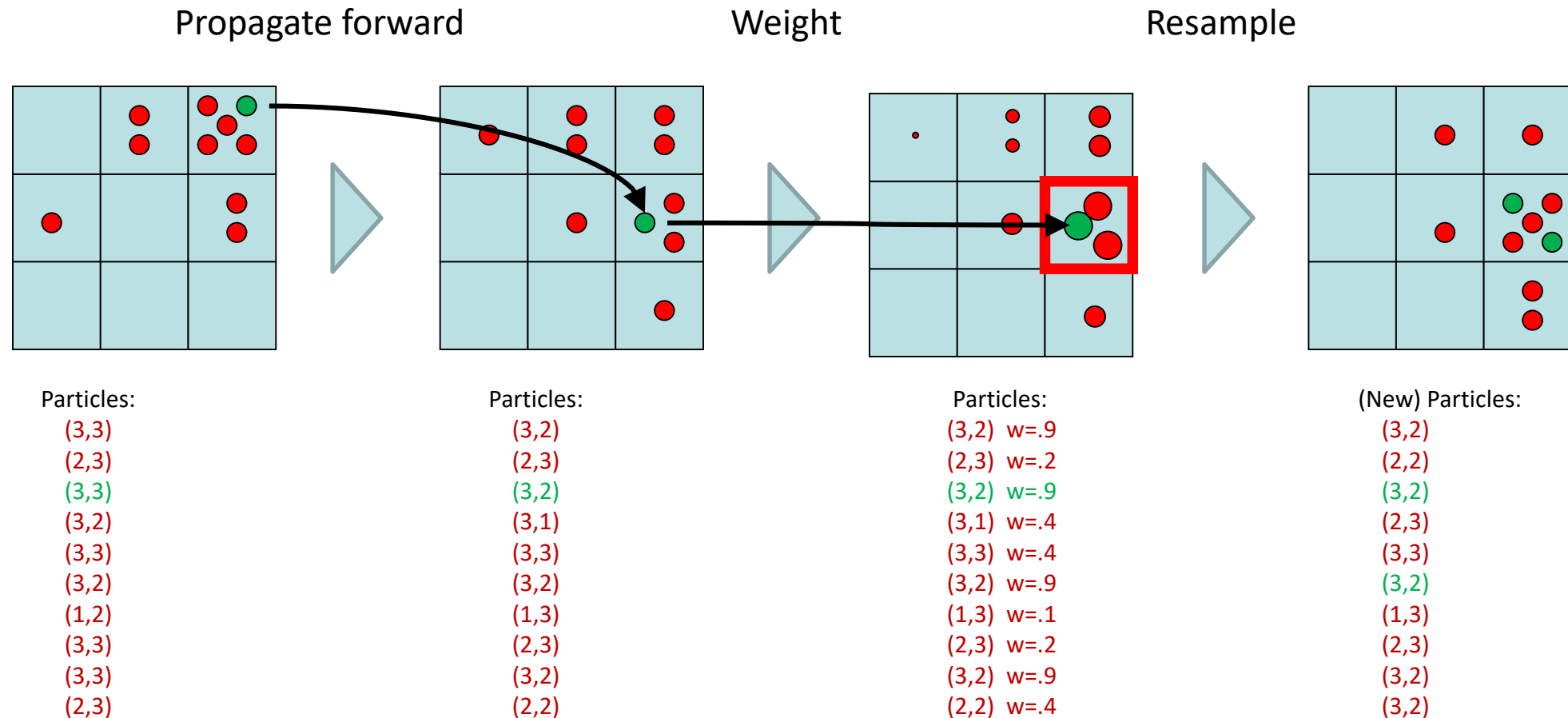
(New) Particles:

(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



Summary: Particle Filtering

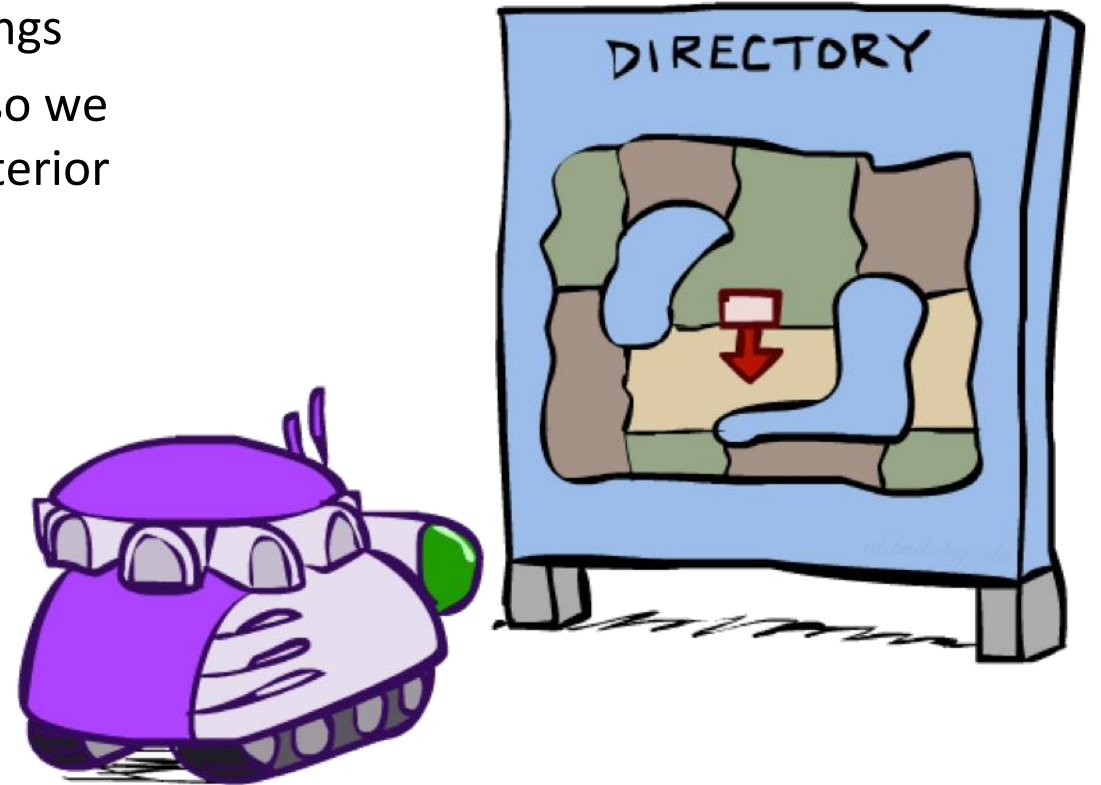
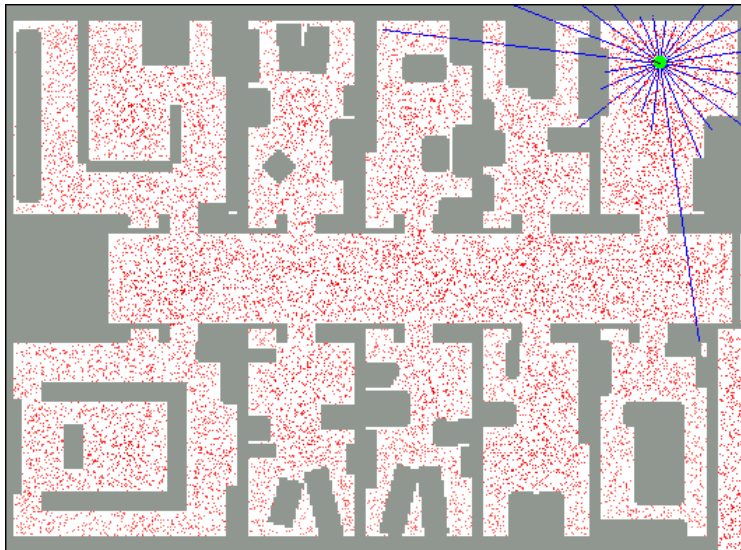
- Particles: track samples of states rather than an explicit distribution



Consistency: see proof in AIMA Ch. 15

Robot Localization

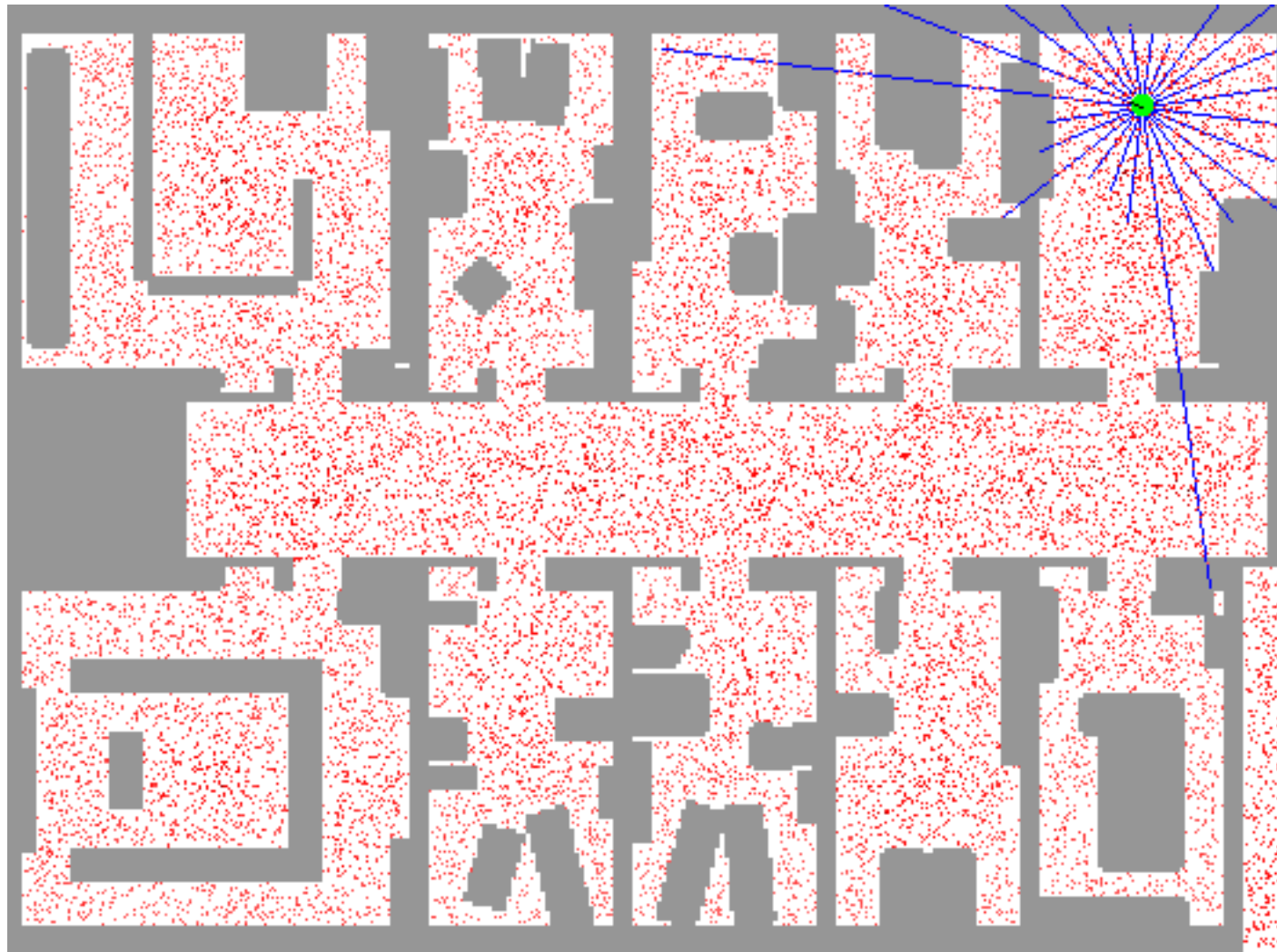
- In robot localization:
 - We know the map, but not the robot's position
 - Observations may be vectors of range finder readings
 - State space and readings are typically continuous so we cannot usually represent or compute an exact posterior
 - Particle filtering is a main technique



Particle Filter Localization (Sonar)

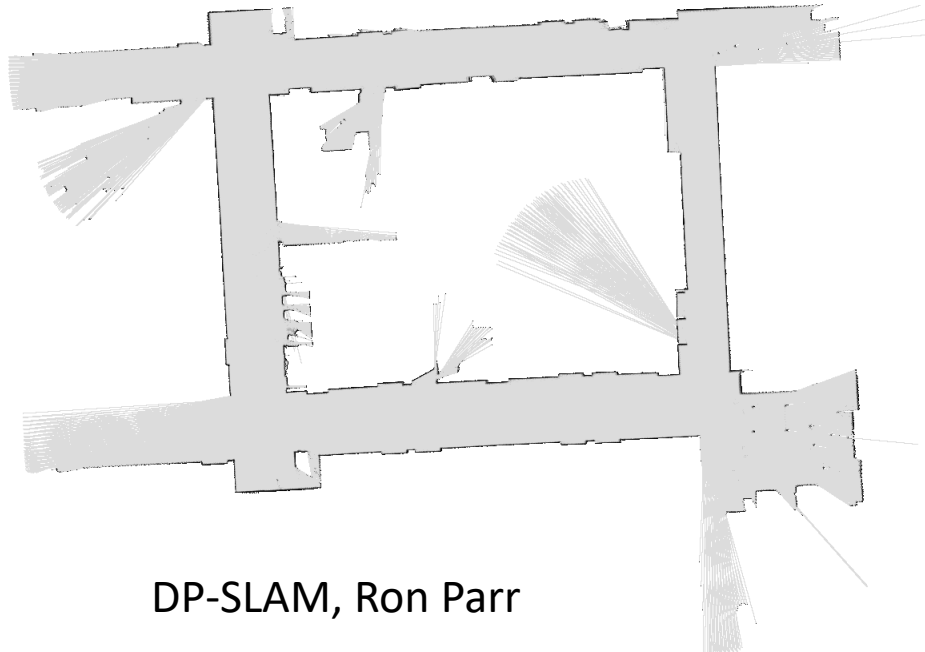


Particle Filter Localization (Laser)

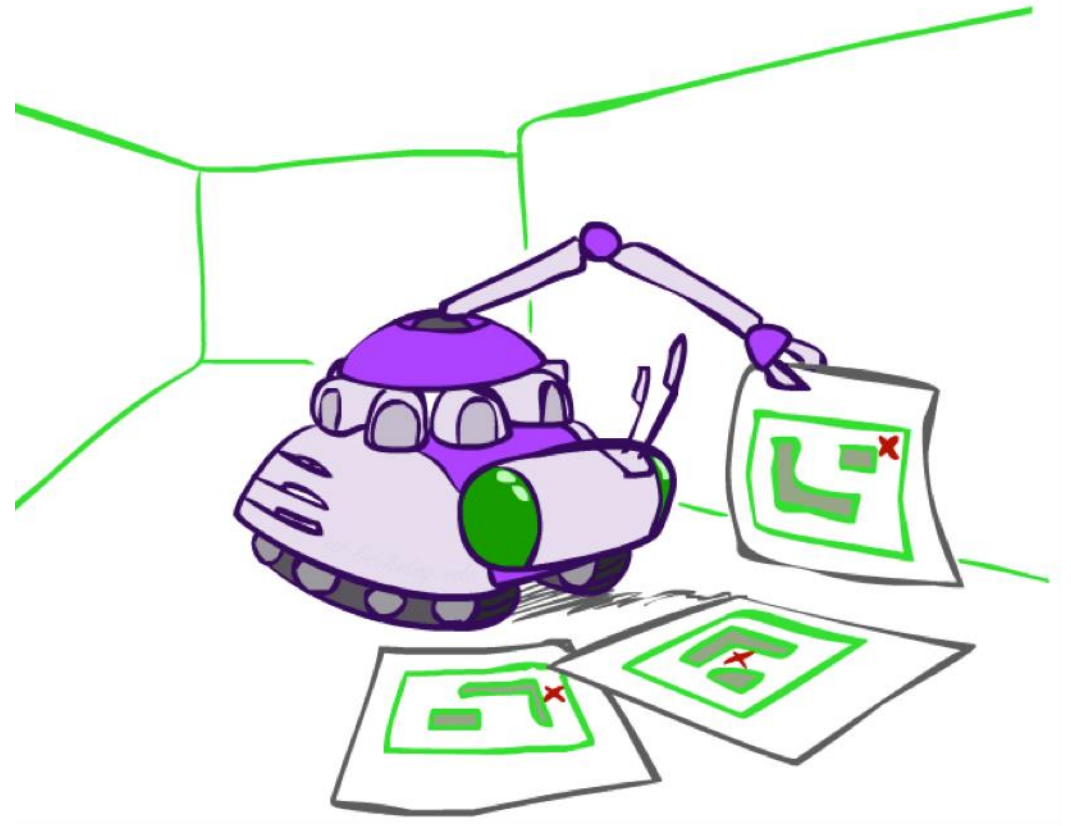


Robot Mapping

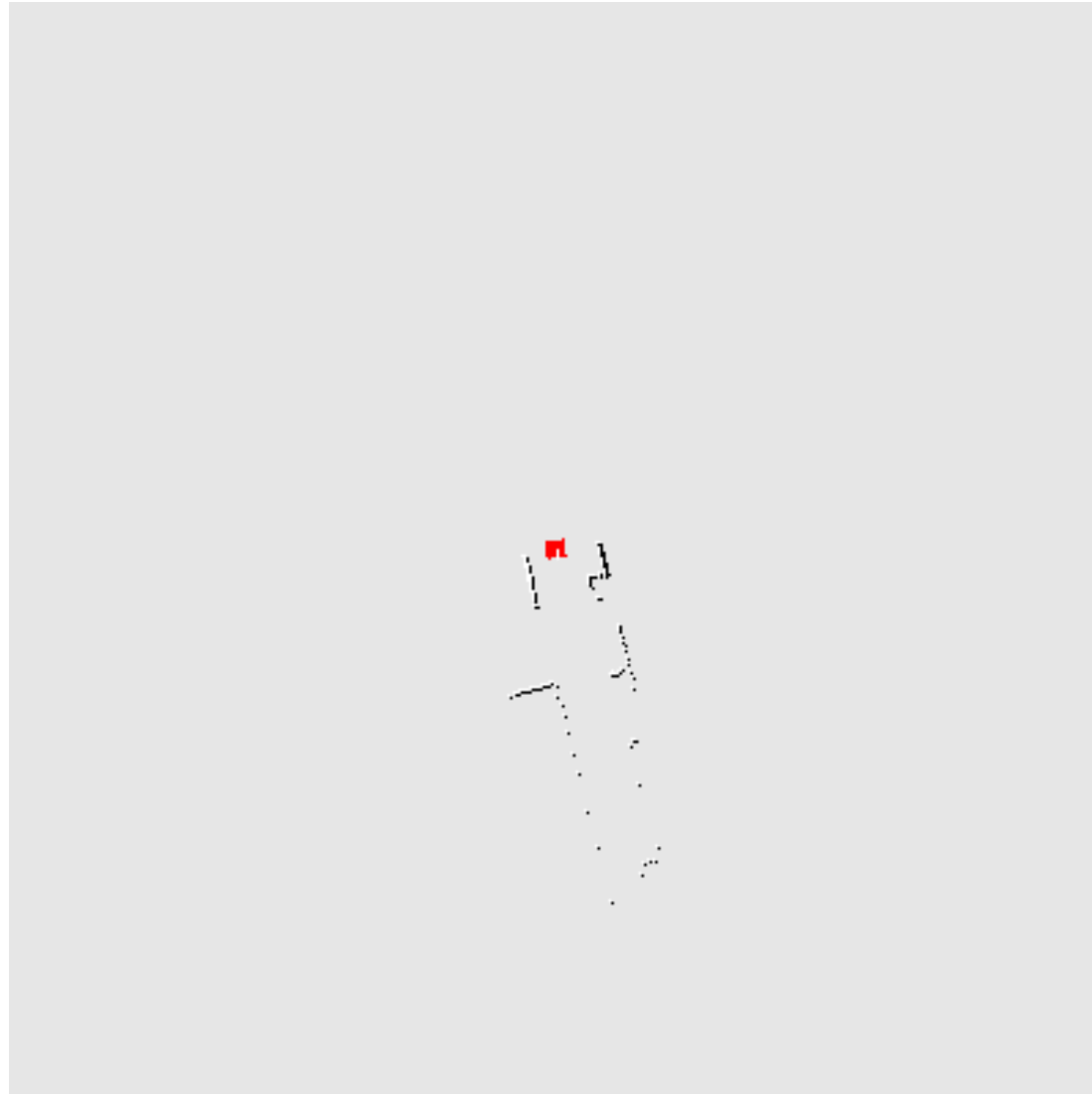
- SLAM: Simultaneous Localization And Mapping
 - We do not know the map or our location
 - State consists of position AND map!
 - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods



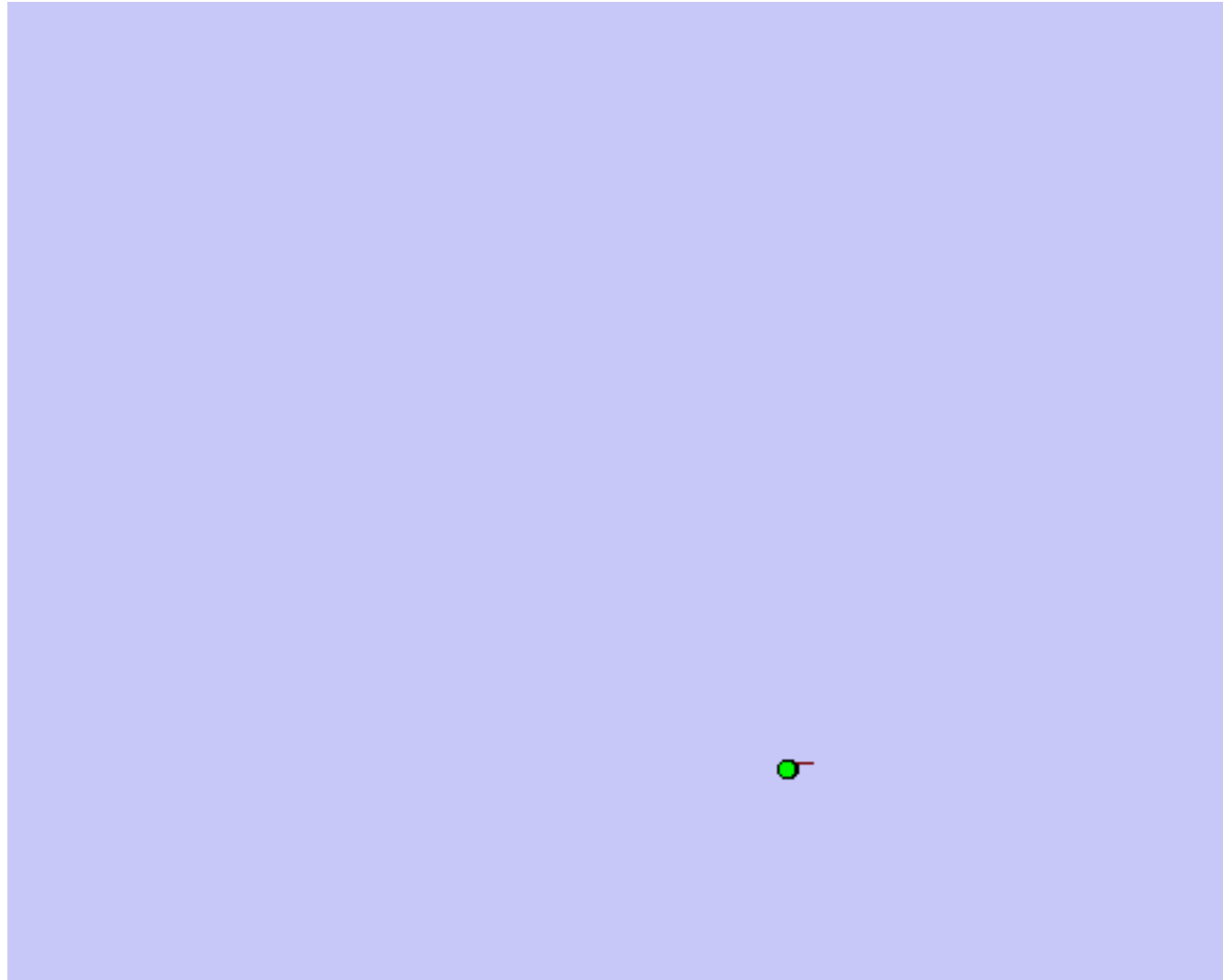
DP-SLAM, Ron Parr



Particle Filter SLAM – Video 1



Particle Filter SLAM – Video 2



Summary

- Probabilistic temporal models
 - Markov model
 - Hidden Markov model
 - Filtering: forward algorithm
 - MLE: Viterbi algorithm
 - Dynamic Bayesian network
 - Approximate inference by particle filtering

