Markov Models

Uncertainty and Time

- Often, we want to reason about a ***sequence*** of observations where the state of the underlying system is ***changing***
  - Speech recognition
  - Robot localization
  - User attention
  - Medical monitoring
  - Global climate

- Need to introduce time into our models

Markov Models (aka Markov chain/process)

$X \to Y \to Z$

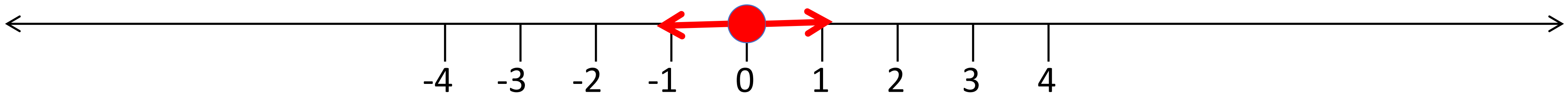
$P(X_t = x \mid X_{t-1} = y) = f(x, y)$

$$X_0 \to X_1 \to X_2 \to X_3 \dashrightarrow$$

$P(X_0)$ $\qquad$ $P(X_t \mid X_{t-1})$

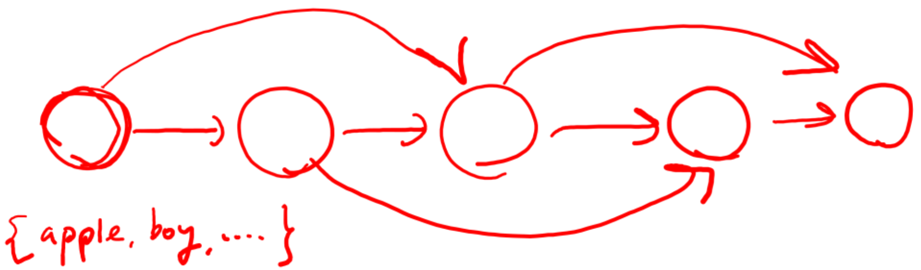
- Value of X at a given time is called the **state**
- The **transition model** $P(X_t \mid X_{t-1})$ specifies how the state evolves over time
- **Stationarity** assumption: transition probabilities are the same at all times
- **Markov** assumption: “future is independent of the past given the present”
  - X_{t+1} is independent of $X_0, \ldots, X_{t-1}$ given X_t

Example: Random walk in one dimension



- State: location on the unbounded integer line
- Initial probability: starts at 0
- Transition model: $P(X_t = k | X_{t-1} = k \pm 1) = 0.5$
- Applications: particle motion in crystals, stock prices, etc.

Example: n-gram models



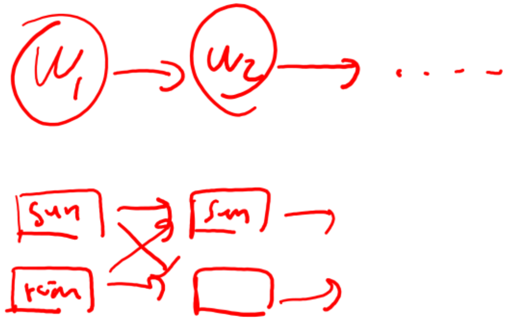
- State: word at position t in text (can also build letter n-grams)
- Transition model (probabilities come from empirical frequencies):
  - Unigram (zero-order): $P(Word_t = i)$
    - "logical are as are confusion a may right tries agent goal the was . . ."
  - Bigram (first-order): $P(Word_t = i \mid Word_{t-1} = j)$
    - "systems are very similar computational approach would be represented . . ."
  - Trigram (second-order): $P(Word_t = i \mid Word_{t-1} = j, Word_{t-2} = k)$
    - "planning and scheduling are integrated the success of naive bayes model is . . ."
- Applications: text classification, spam detection, author identification, language classification, speech recognition

Example: Web browsing

- State: URL visited at step t
- Transition model:
  - With probability p, choose an outgoing link at random
  - With probability $(1-p)$, choose an arbitrary new page
- Question: What is the ***stationary distribution*** over pages?
  - I.e., if the process runs forever, what fraction of time does it spend in any given page?
- Application: Google page rank

Example: Weather

Bayes net



- States {rain, sun}

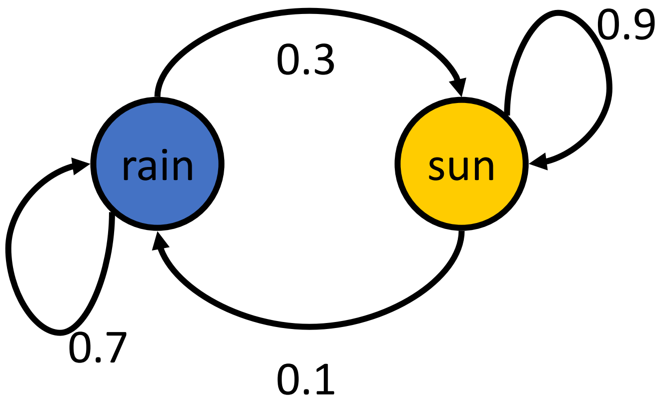
- Initial distribution $P(X_0)$

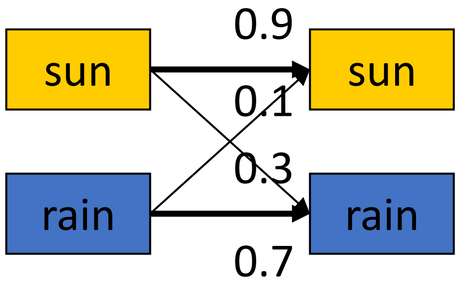
$P(X_0)$	
sun	rain
0.5	0.5

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

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

Two ways to represent Markov chains





Weather prediction

$P(X_{t-1})$ join $P(X_{t-1}, X_t) \xrightarrow{\text{margin}} P(X_t)$

- Time 0: <0.5,0.5>

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

- What is the weather like at time 1?

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

Weather prediction, contd.

- Time 1: <0.6,0.4>

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

- What is the weather like at time 2?

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

Weather prediction, contd.

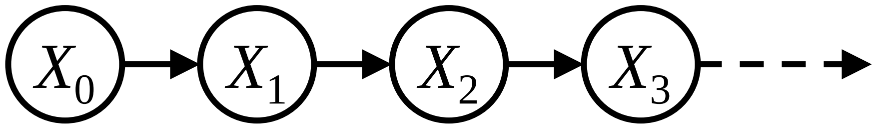
- Time 2: <0.66,0.34>

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

- What is the weather like at time 3?

$$P(X_3) = \sum_{x_2} P(X_3, X_2 = x_2)$$
$$= \sum_{x_2} P(X_2 = x_2)\, P(X_3 \mid X_2 = x_2)$$
$$= 0.66<0.9,0.1> + 0.34<0.3,0.7> = <0.696,0.304>$$

Forward algorithm (simple form)



$P(X_0)$ $\quad$ $P(X_t \mid X_{t-1})$

What is the state at time t?

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

Forward algorithm in Matrices

- What is the weather like at time 2?
  - $P(X_2) = 0.6<0.9,0.1> + 0.4<0.3,0.7> = <0.66,0.34>$
- In matrix-vector form:
  - $P(X_2) = \begin{pmatrix} 0.9 & 0.3 \\ 0.1 & 0.7 \end{pmatrix} \underbrace{\begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix}}_{P(X_1)} = \begin{pmatrix} 0.66 \\ 0.34 \end{pmatrix}$

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

Stationary Distributions

- The limiting distribution is called the ***stationary distribution*** P_{∞} of the chain
- It satisfies $P_{\infty} = P_{\infty+1} = T^{T} P_{\infty}$

  Stationary distribution is <0.75,0.25> ***regardless of starting distribution***

sum

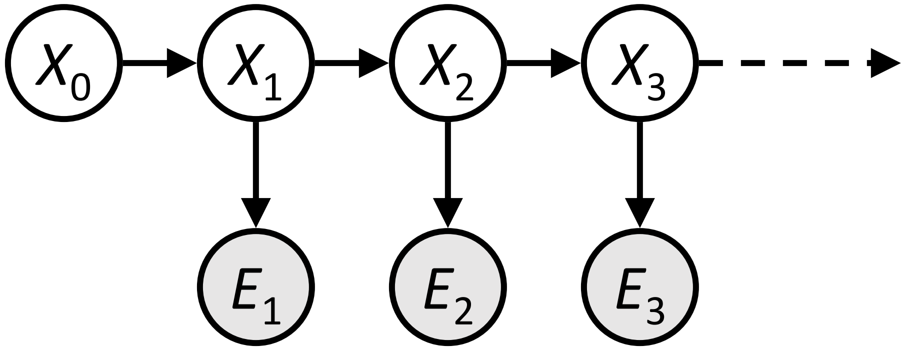
$$\begin{pmatrix}0.9 & 0.3\\ 0.1 & 0.7\end{pmatrix}\begin{pmatrix}p\\ 1-p\end{pmatrix} = \begin{pmatrix}p\\ 1-p\end{pmatrix}$$

Hidden Markov Models

Hidden Markov Models

- Usually the true state is not observed directly

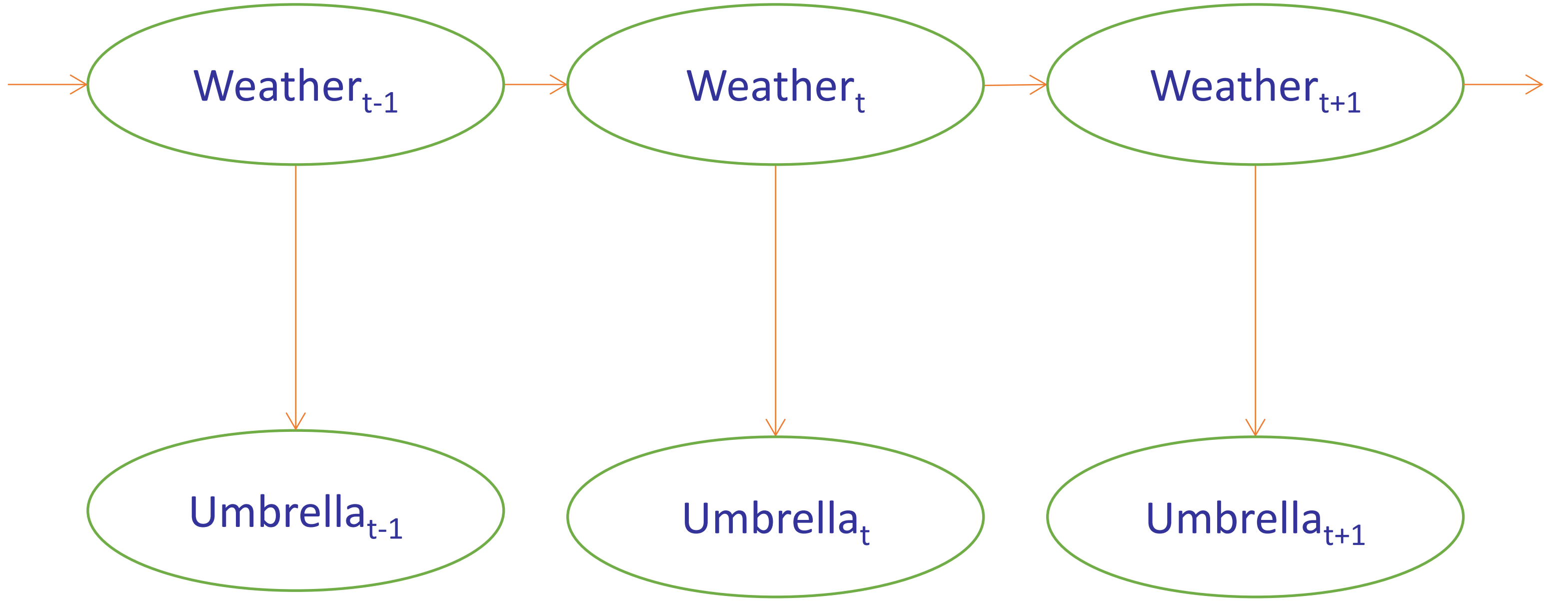
- Hidden Markov models (HMMs)
  - Underlying Markov chain over states X
  - You observe evidence E at each time step
  - X_t is a single discrete variable; E_t may be continuous and may consist of several variables



Example: Weather HMM

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

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

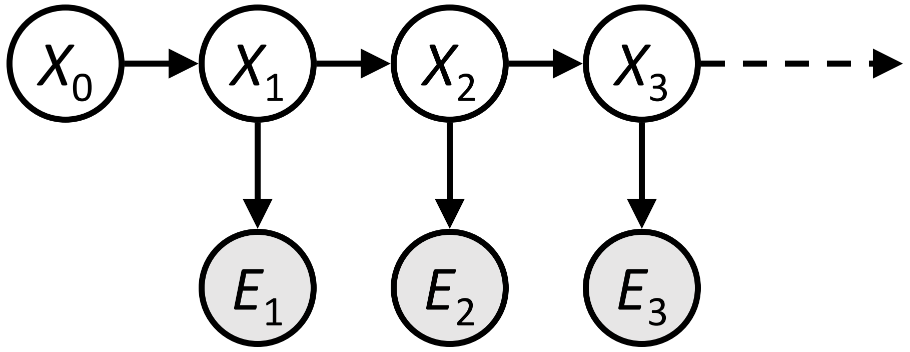


W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

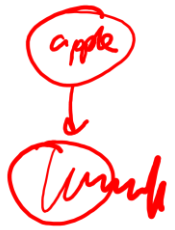
HMM as probability model

- Joint distribution for Markov model: $P(X_0,\ldots, X_T) = P(X_0) \prod_{t=1:T} P(X_t \mid X_{t-1})$
- Joint distribution for hidden Markov model:

  $P(X_0,E_0,X_1,E_1,\ldots, X_T,E_T) = P(X_0) \prod_{t=1:T} P(X_t \mid X_{t-1})\, P(E_t \mid X_t)$
- Future states are independent of the past given the present
- Current evidence is independent of everything else given the current state
- Are evidence variables independent of each other?



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

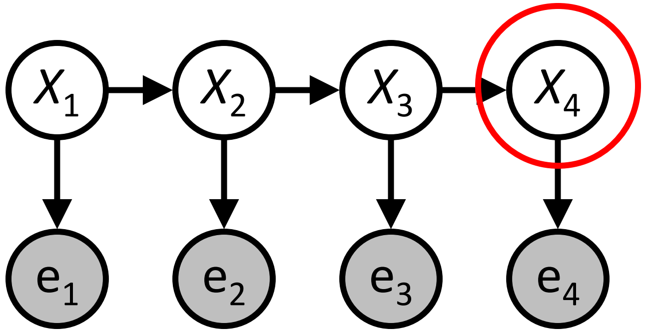
Useful notation:

$X_{a:b} = X_a, X_{a+1}, \ldots, X_b$

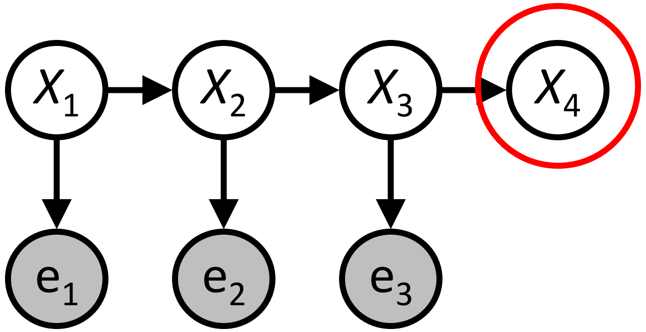
- **Filtering**: $P(X_t | e_{1:t})$
  - **belief state**—input to the decision process of a rational agent
- **Prediction**: $P(X_{t+k} | e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- **Smoothing**: $P(X_k | e_{1:t})$ for $0 \leq k < t$
  - better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

Inference tasks

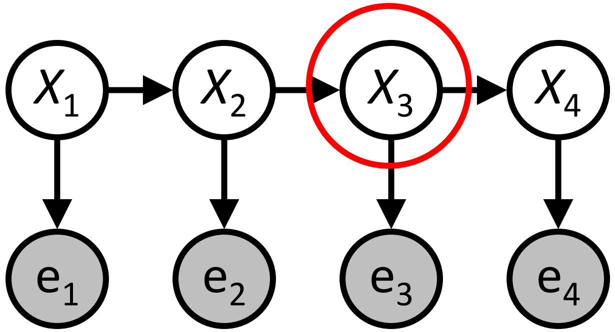
Filtering: $P(X_t|e_{1:t})$



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

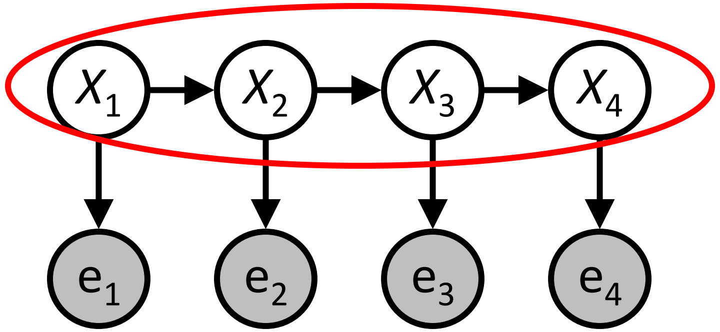


Smoothing: $P(X_k|e_{1:t})$, $k<t$



argmax

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



Filtering / Monitoring

- Filtering, or monitoring, or state estimation, is the task of maintaining the distribution $P(X_t|e_{1:t})$ over time

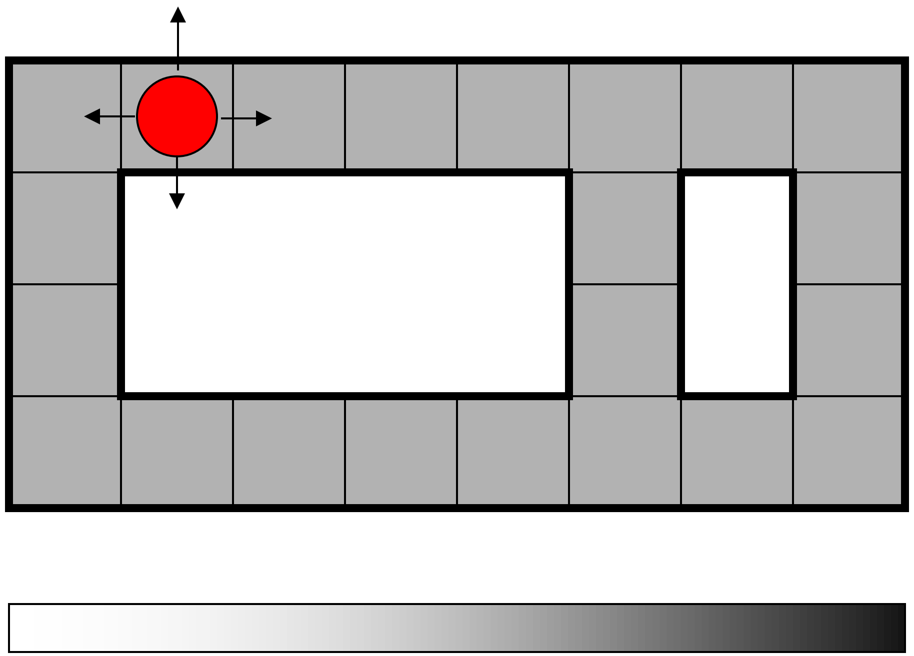
(transition) (emission)

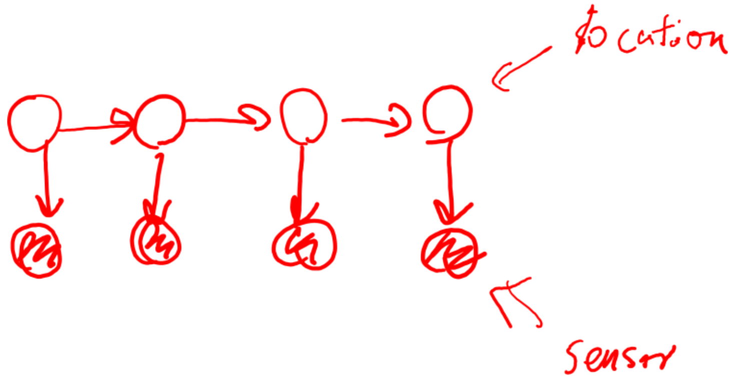
$P(X_t|X_{t-1})$ $P(e_t|X_t)$

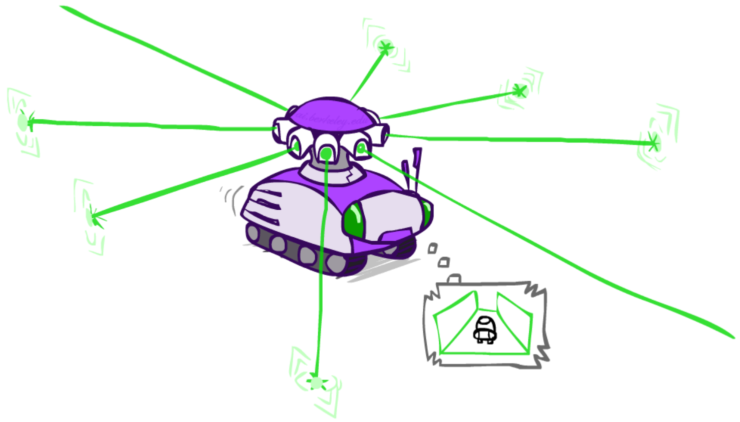
- The Kalman filter (continuous variables, ~~linear~~ $\boxed{\text{dynamics,}}$ Gaussian noise) was invented in 1960 and used for trajectory estimation in the Apollo program.

Example: Robot Localization

Example from Michael Pfeiffer







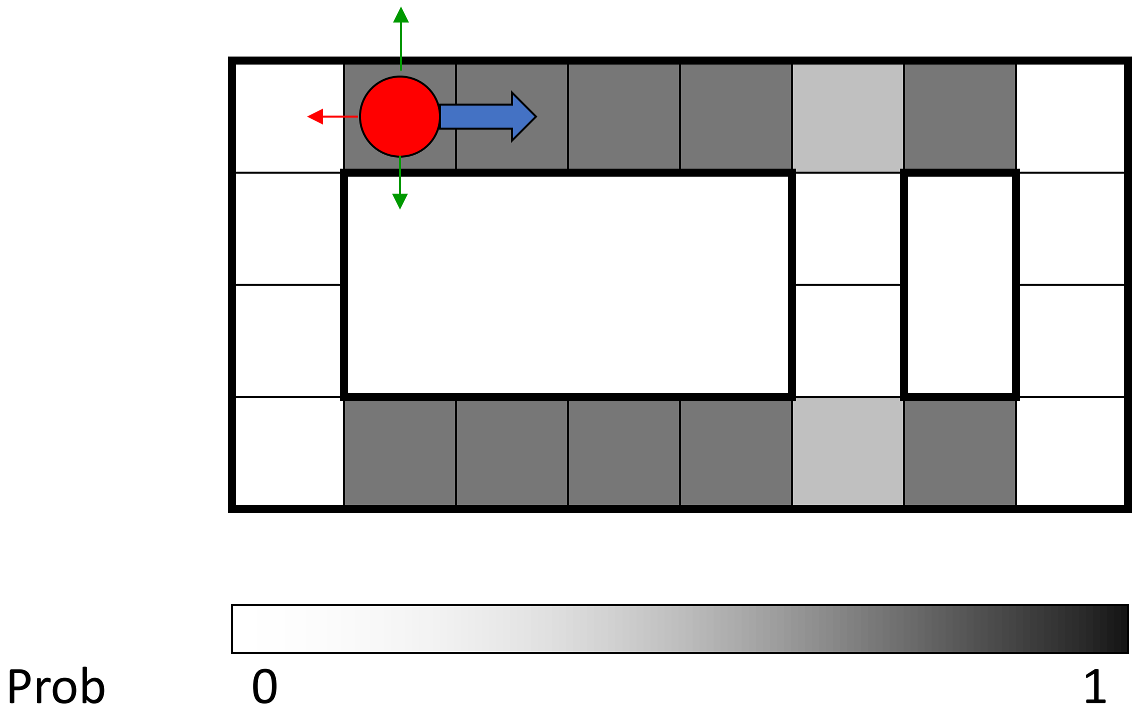
Prob 0 1

t=0

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

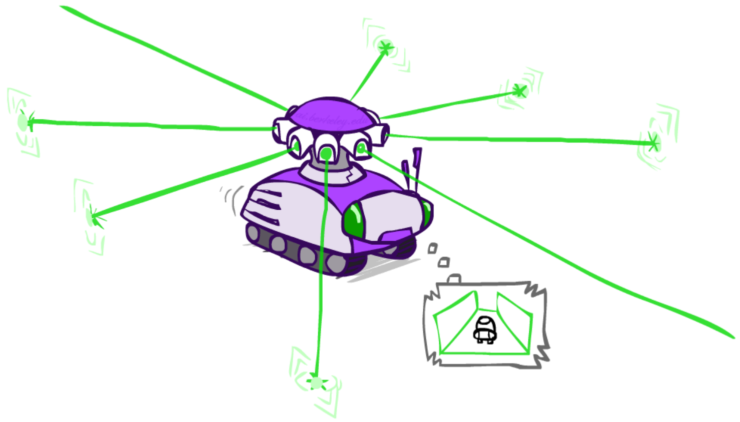
Transition model: action may fail with small prob.

Example: Robot Localization

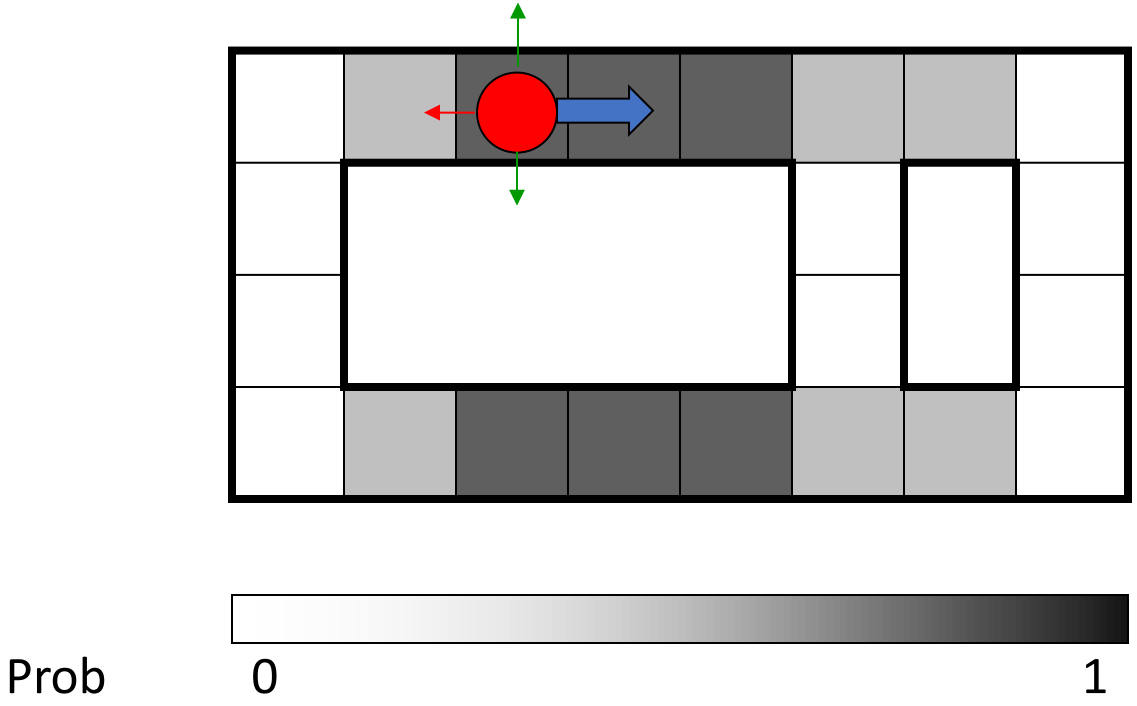


t=1

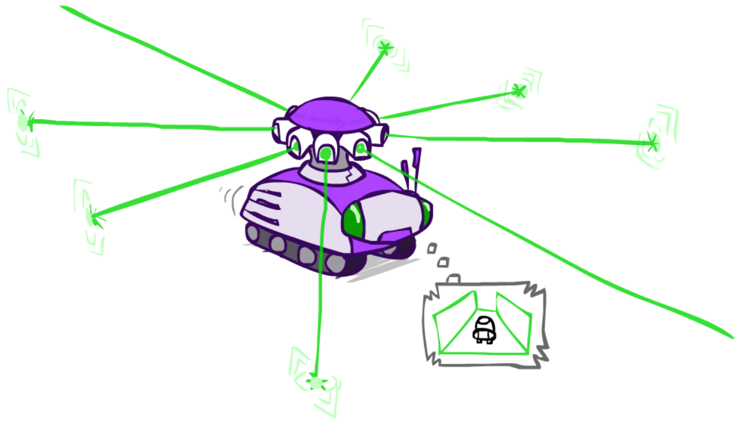
Lighter grey: was ***possible*** to get the reading, but ***less likely*** (required 1 mistake)



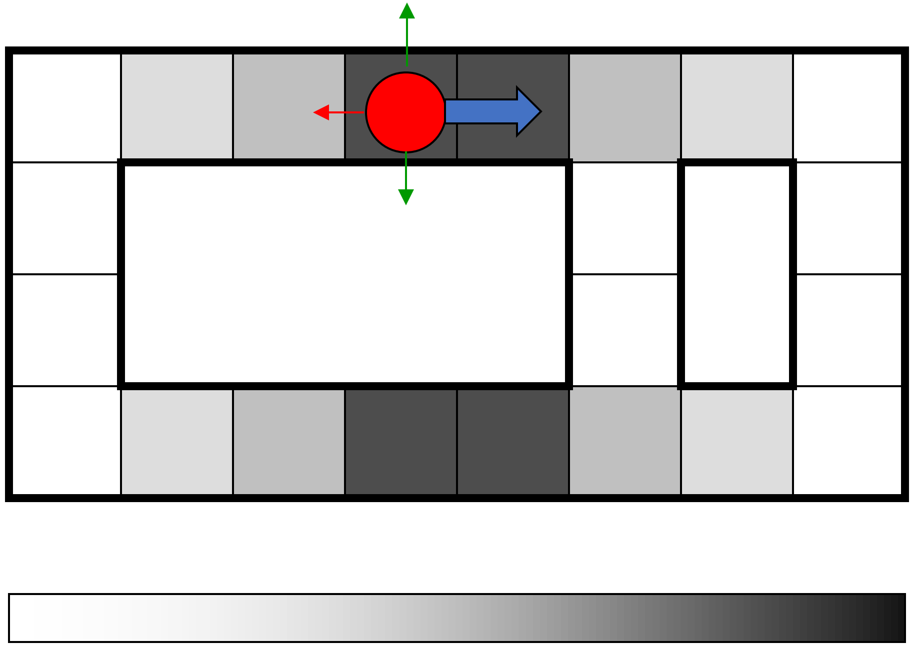
Example: Robot Localization

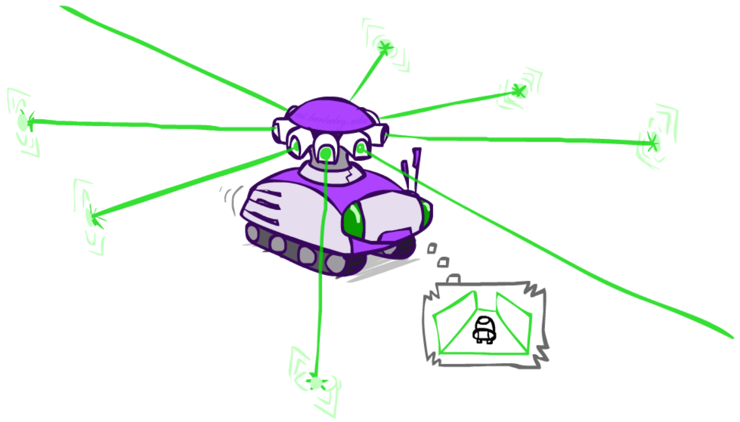


t=2



Example: Robot Localization

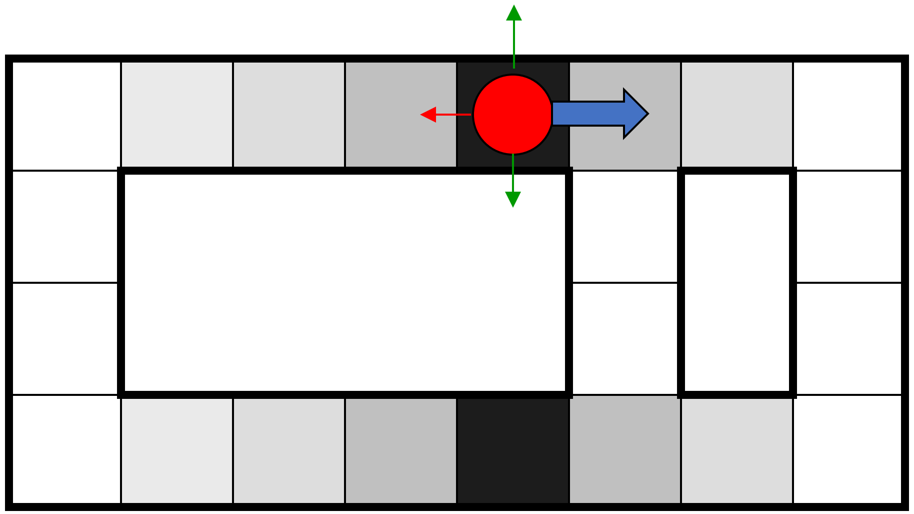


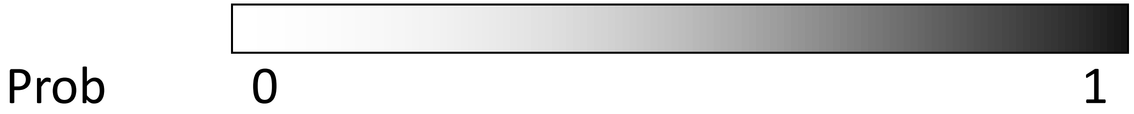


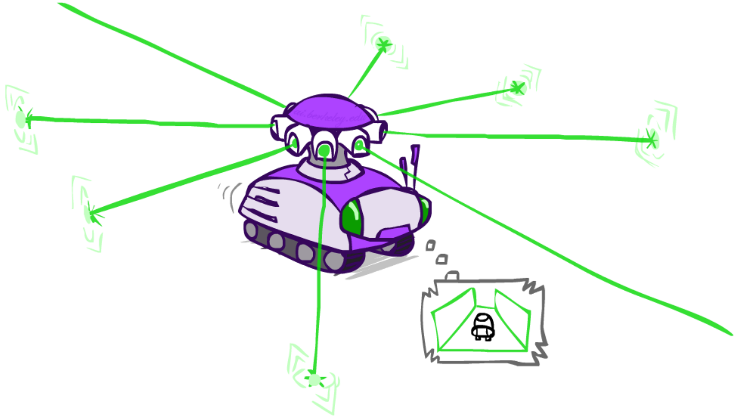
Prob 0 1

t=3

Example: Robot Localization

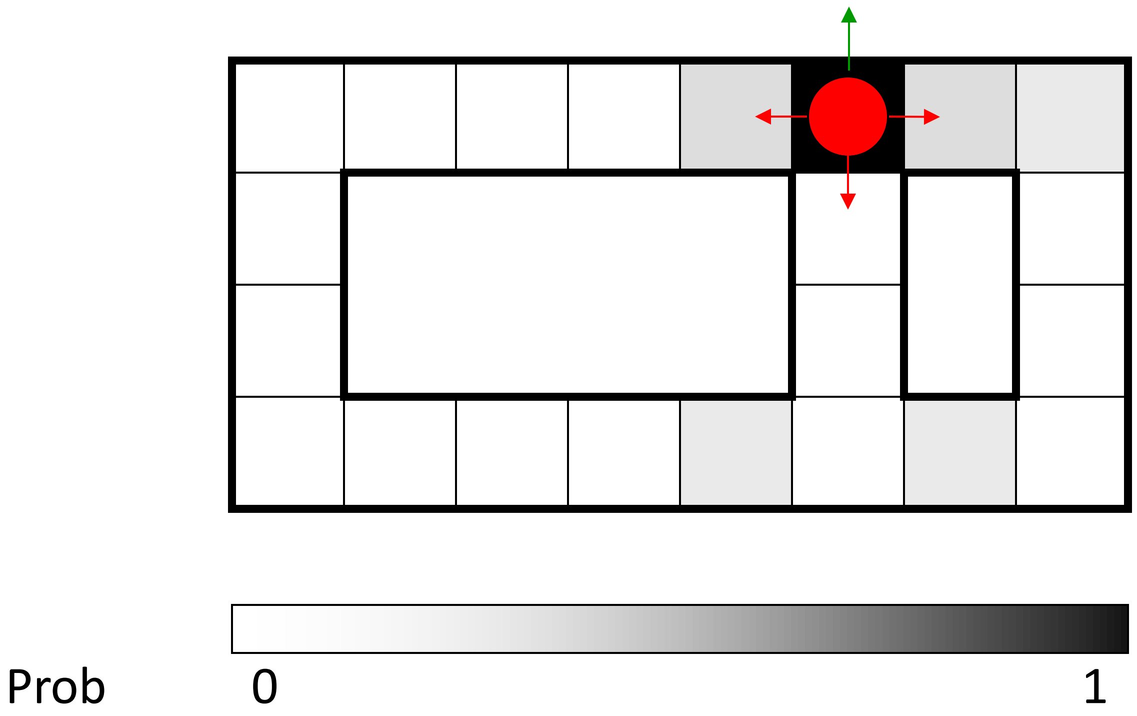




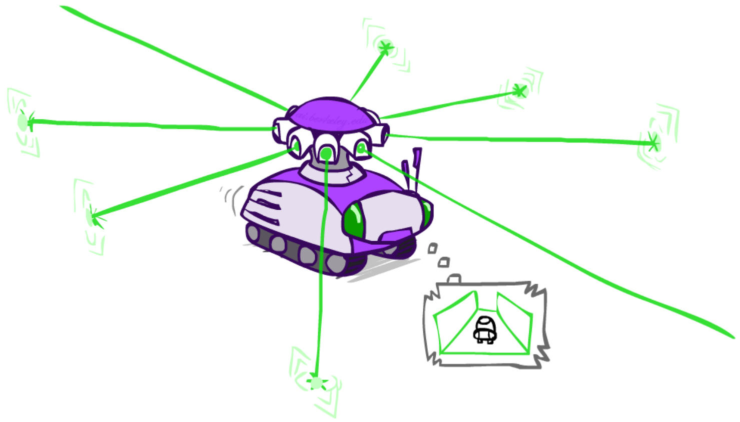


t=4

Example: Robot Localization



t=5

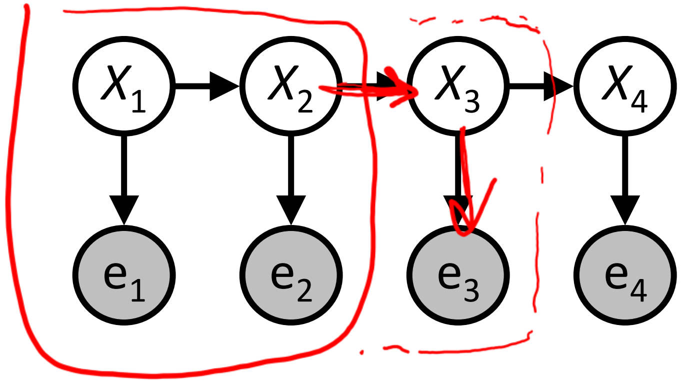


Exact Inference in HMM

Filtering

$P(X_1)$ $\quad P(X_t \mid X_{t-1})$

$P(E_t \mid X_t)$



$$\boxed{P(X_t \mid e_{1:t}) = ?}$$

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

Base case: $P(X_1 \mid e_1) \propto P(X_1, e_1) = P(X_1)P(e_1|X_1)$

Passage of time:

Suppose we have $P(X_t \mid e_{1:t})$.

How to calculate $P(X_{t+1} \mid e_{1:t+1})$?

$|X|$ $\quad |E|$

$$P(X_t \mid e_{1:t}) \xrightarrow{\text{Joining } P(X_{t+1} \mid X_t)} P(X_{t+1}, X_t \mid e_{1:t}) \xrightarrow{\text{Joining } P(e_{t+1} \mid X_{t+1})} P(X_{t+1}, e_{t+1}, X_t \mid e_{1:t}) \xrightarrow{\text{Marginalize out } X_t} P(X_{t+1}, e_{t+1} \mid e_{1:t}) \xrightarrow{\text{Normalize}} P(X_{t+1} \mid e_{1:t+1})$$

$\varnothing$

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

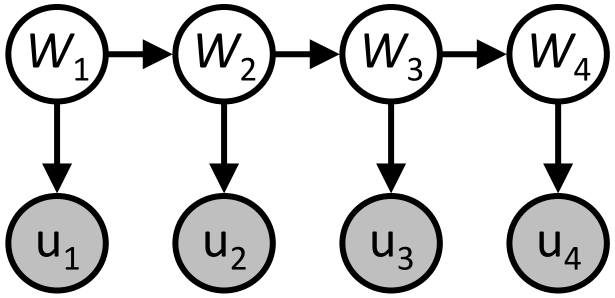
Time complexity?

$O(|X| \cdot |X|\, t)$

Exercise

$$P(W_1 \mid U_1 = T) \approx$$

W_1	$P(W_1 \mid U_1=T)$
s	2/11
r	9/11



$$P(W_2 \mid U_{1:2} = (T, F)\,) = ?$$

$$P(W_1 \mid U_1 = T) \propto P(W_1, U_1 = T) = P(W_1)\, P(U_1 = T \mid W_1)$$

$$= \begin{cases} W_1 = \text{sun}: & 0.5 \times 0.2 \\ W_1 = \text{rain}: & 0.5 \times 0.9 \end{cases}$$

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

W_t	$P(U_t \mid W_t)$	
	T	F
sun	0.2	0.8
rain	0.9	0.1

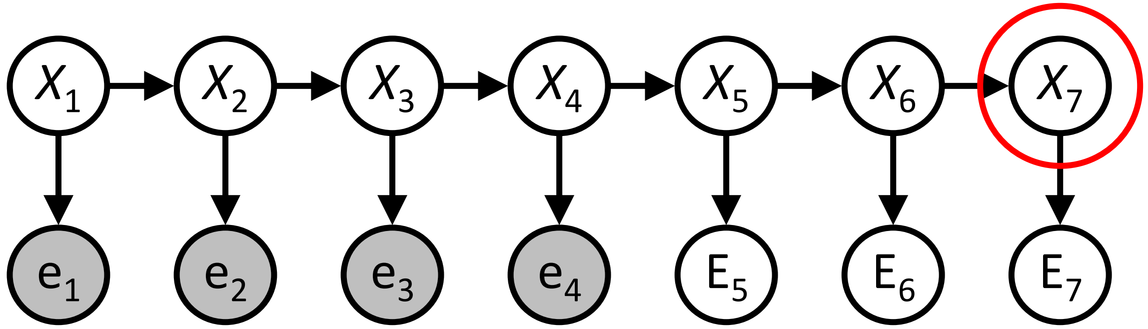
W_1	$P(W_1)$
s	0.5
r	0.5

$$P(W_2 \mid U_1 = T, U_2 = F) \propto \sum_{w_1} P(w_1 \mid U_1 = T)\, P(W_2 \mid w_1)\, P(U_2 = F \mid W_2)$$

$$= P(W_1 = s \mid U_1 = T)\, P(W_2 \mid W_1 = s)\, P(U_2 = F \mid W_2)$$

$$+ P(W_1 = r \mid U_1 = T)\, P(W_2 \mid W_1 = r)\, P(U_2 = F \mid W_2)$$

Prediction



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

We already have $P(X_t \mid e_{1:t})$ by filtering

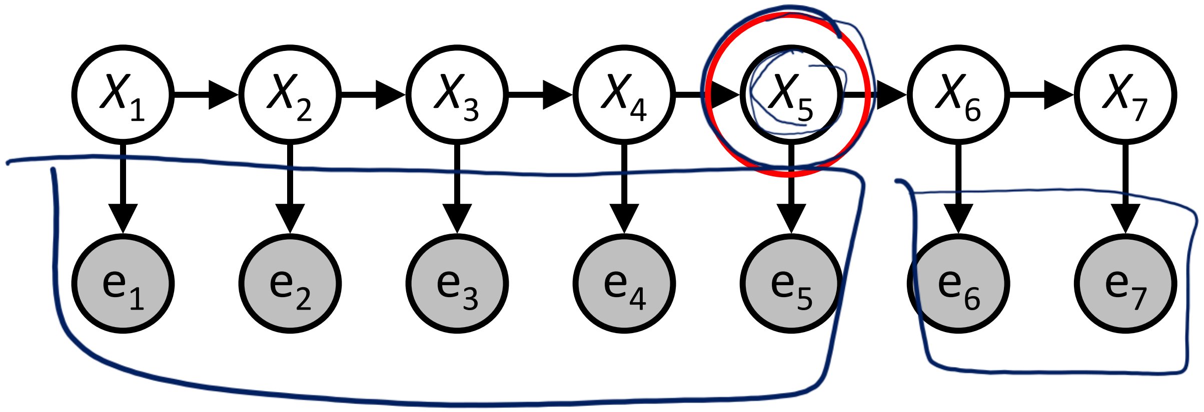
$$P(X_{t+1} \mid e_{1:t}) = \sum_{x_t} P(x_t \mid e_{1:t}) P(X_{t+1} \mid x_t)$$

$$P(X_{t+2} \mid e_{1:t}) = \sum_{x_{t+1}} P(x_{t+1} \mid e_{1:t}) P(X_{t+2} \mid x_{t+1})$$

$$\vdots$$

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

Smoothing



$P(X_k \mid e_{1:t}) = ?$ for some $k < t$

Here we introduce an approach slightly different from variable elimination.

$\times P(e_{k+1:t} \mid X_k, \cancel{e_{1:k}})$

$$P(X_k \mid e_{1:t}) \propto P(X_k, e_{k+1:t} \mid e_{1:k}) = P(X_k \mid e_{1:k})\, P(e_{k+1:t} \mid X_k)$$

Forward algorithm (filtering) Backward algorithm

Just with one forward pass and one backward pass, we can calculate $P(X_k \mid e_{1:t})$ **for all k.**

$$\boxed{P(e_{k+1:t} \mid X_k)} \qquad k < t$$

Base Case : $(k = t-1)$

$$P(e_t \mid X_{t-1}) = \sum_{x_t} \overbrace{P(X_t \mid X_{t-1})\, P(e_t \mid X_t)}^{P(X_t, e_t \mid X_{t-1})}$$

Backward Pass: Given $P(e_{k+2:t} \mid X_{k+1})$

$$P(e_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(x_{k+1}, e_{k+1:t} \mid X_k)$$

$$= \sum_{X_{k+1}} P(X_{k+1} \mid X_k)\, \underline{P(e_{k+1:t} \mid X_{k+1}, \cancel{x_k})}$$

$$= \sum_{X_{k+1}} \underline{P(X_{k+1} \mid X_k)}\, \underline{P(e_{k+1} \mid X_{k+1})}\, \boxed{P(e_{k+2:t} \mid X_{k+1})}$$

$$e_{k+1} \perp\!\!\!\perp e_{k+2:t} \mid X_{k+1}$$

Most-Likely Sequence

$$\boxed{\operatorname*{argmax}_{X_{1:t}} P(X_{1:t} \mid e_{1:t}) = ?}$$

Find the sequence that maximizes the probability (e.g., speech recognition, sequence decoding)

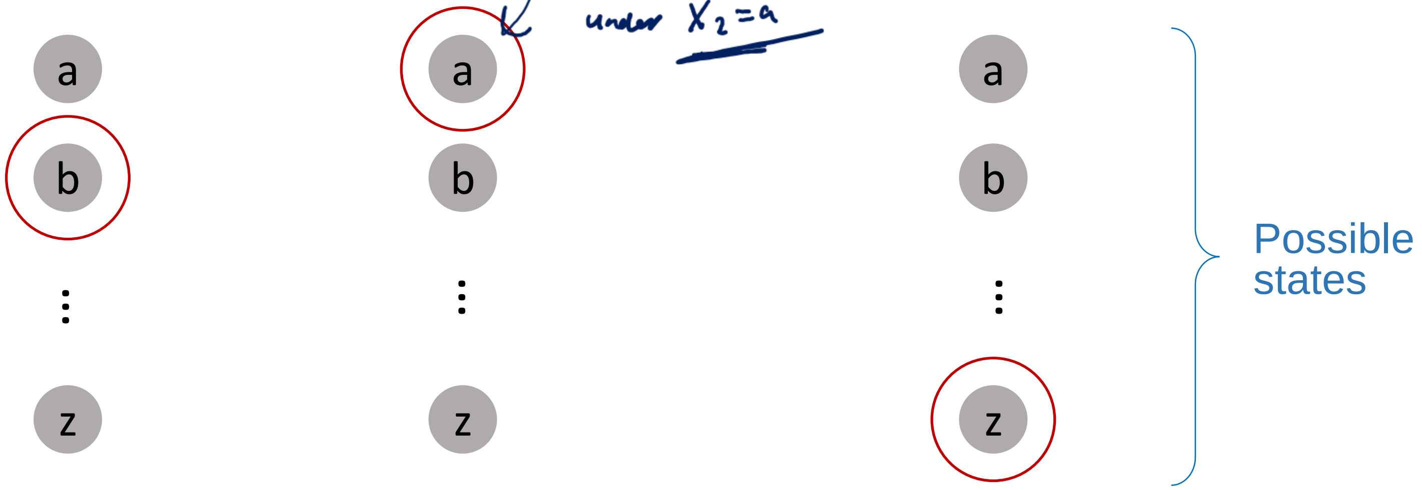
$$P(X_{1:t} \mid e_{1:t}) \propto P(X_{1:t}, e_{1:t}) = \underbrace{P(X_1)P(e_1|X_1)}_{\text{Time 1}}\ \underbrace{P(X_2|X_1)P(e_2|X_2)}_{\text{Time 2}} \cdots \underbrace{P(X_t|X_{t-1})P(e_t|X_t)}_{\text{Time t}}$$

Find a sequence, e.g. $X_1 = b,\ X_2 = a,\ \ldots,\ X_t = z$ that maximize $P(X_{1:t}, e_{1:t})$

Most-Likely Sequence through Dynamic Programming

$$P(X_{1:t} \mid e_{1:t}) \propto P(X_{1:t}, e_{1:t}) = \underbrace{P(X_1)P(e_1|X_1)}_{\text{Time 1}}\ \underbrace{P(X_2|X_1)P(e_2|X_2)}_{\text{Time 2}} \cdots \underbrace{P(X_t|X_{t-1})P(e_t|X_t)}_{\text{Time t}}$$

maximum value under $X_2 = a$



Viterbi Algorithm

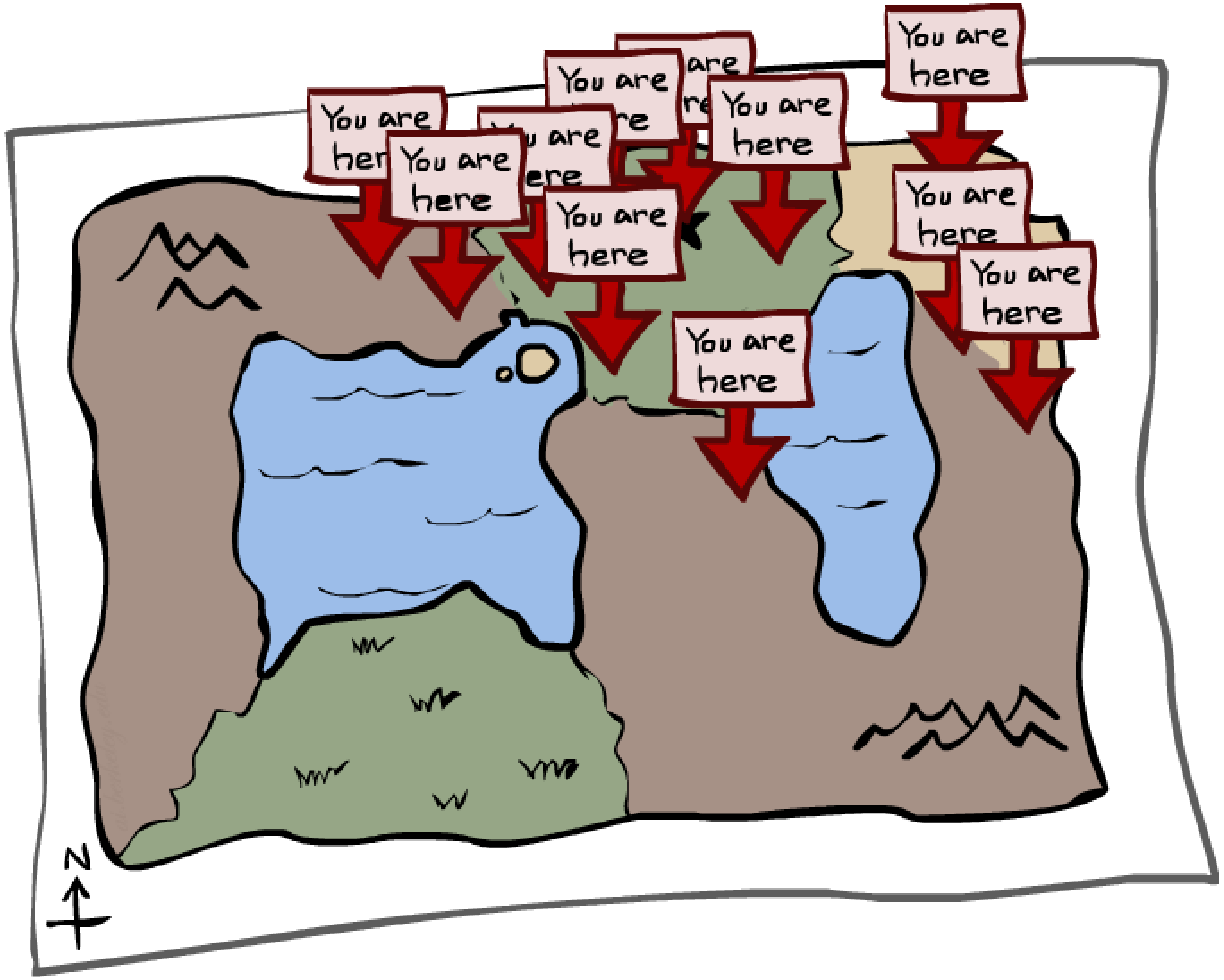
For each state s, let $\text{Prob}[1][s] = P(X_1 = s)\ P(e_1|X_1 = s)$

For $k = 2, \ldots, t$:

For each states s, let $\text{Prob}[k][s] = \max_{s'} \text{Prob}[k-1][s'] \times P(X_k = s \mid X_{k-1} = s') \times P(e_k|X_k = s)$

Approximate Inference in HMM

Particle Filtering



Particle Filtering

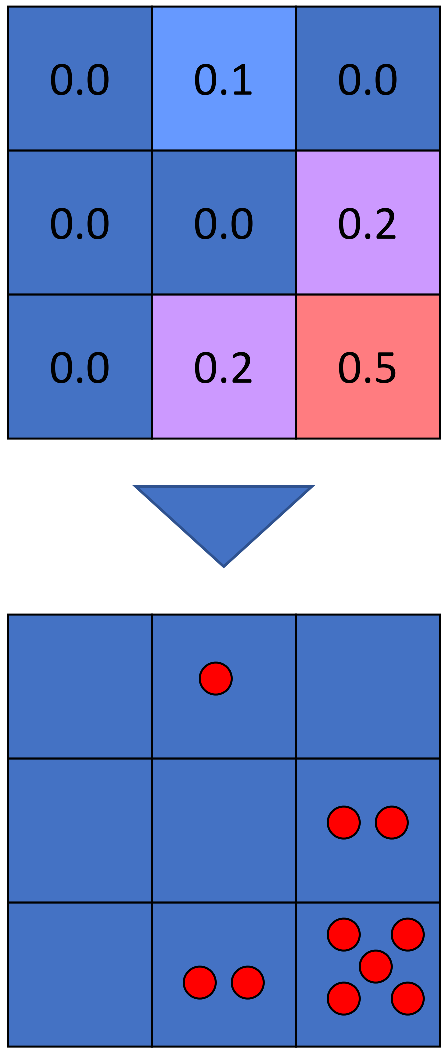
- Filtering: approximate solution

- Sometimes |X| is too big to use exact inference
  - |X| may be too big to even store P(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

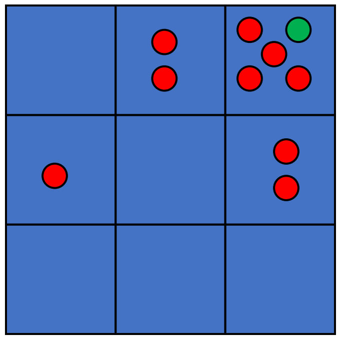


Representation: Particles

- 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

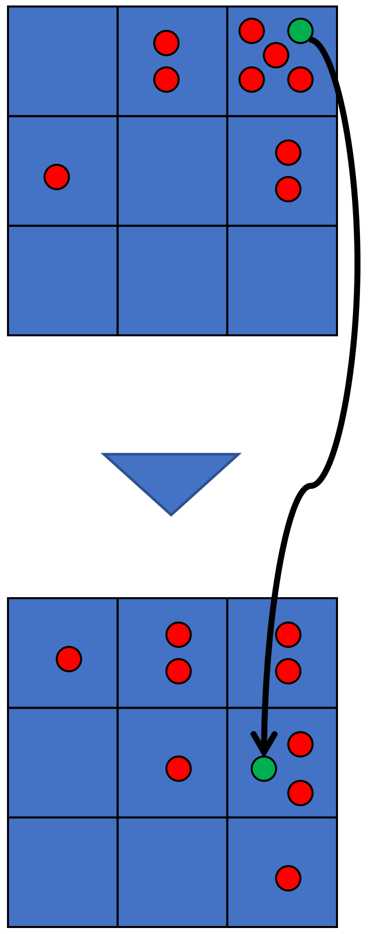
$$x' = \text{sample}(P(X'|x))$$

  - This is like prior sampling – samples' frequencies reflect the transition probabilities

- 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)



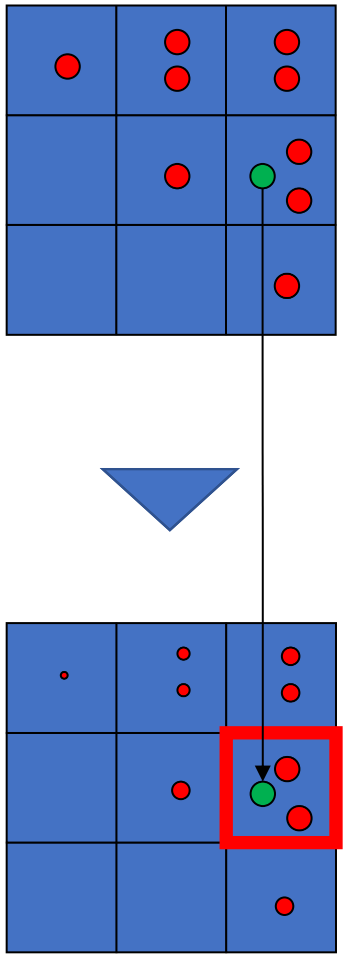
Particle Filtering: Observe

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

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

- As before, the probabilities don’t sum to one, since all have been downweighted

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

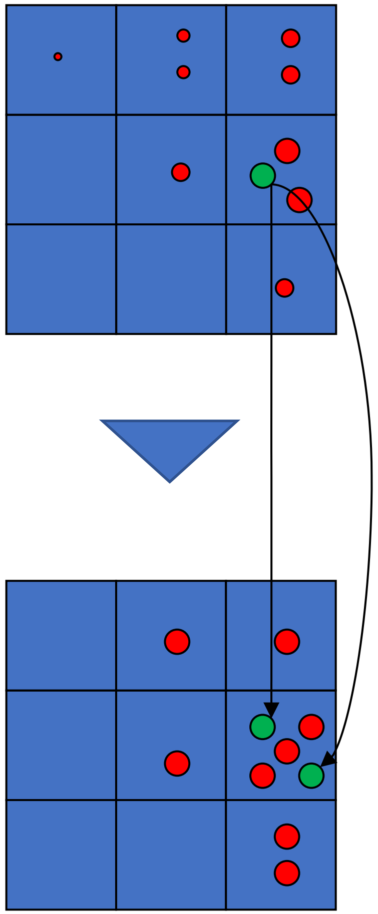


Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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 similar 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

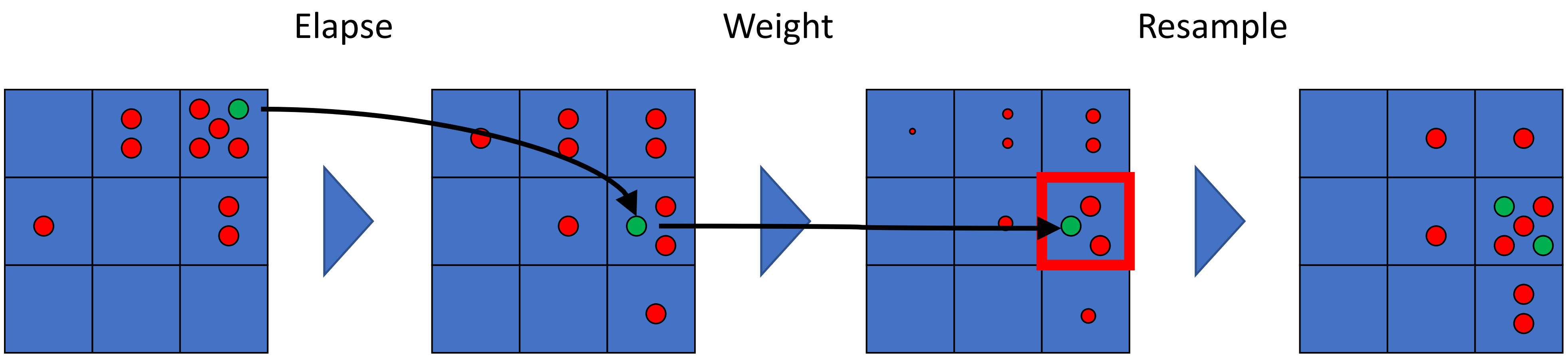


(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)

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

Recap: Particle Filtering

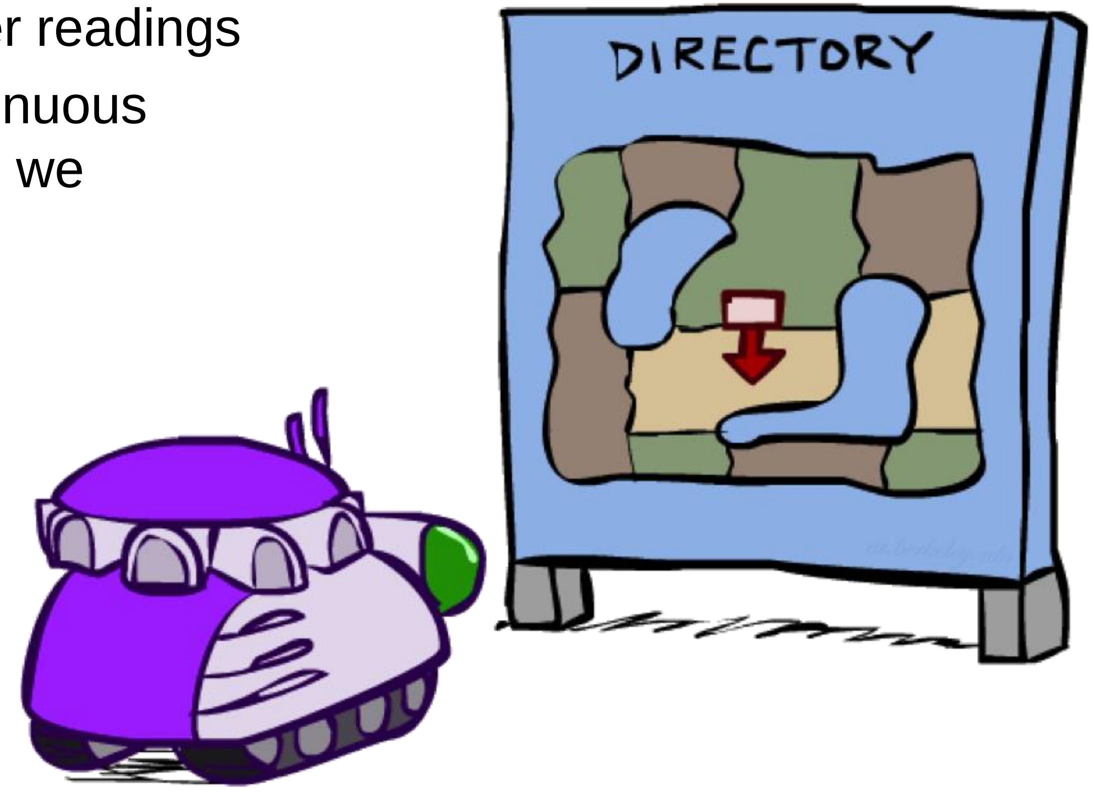
- Particles: track samples of states rather than an explicit distribution



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)

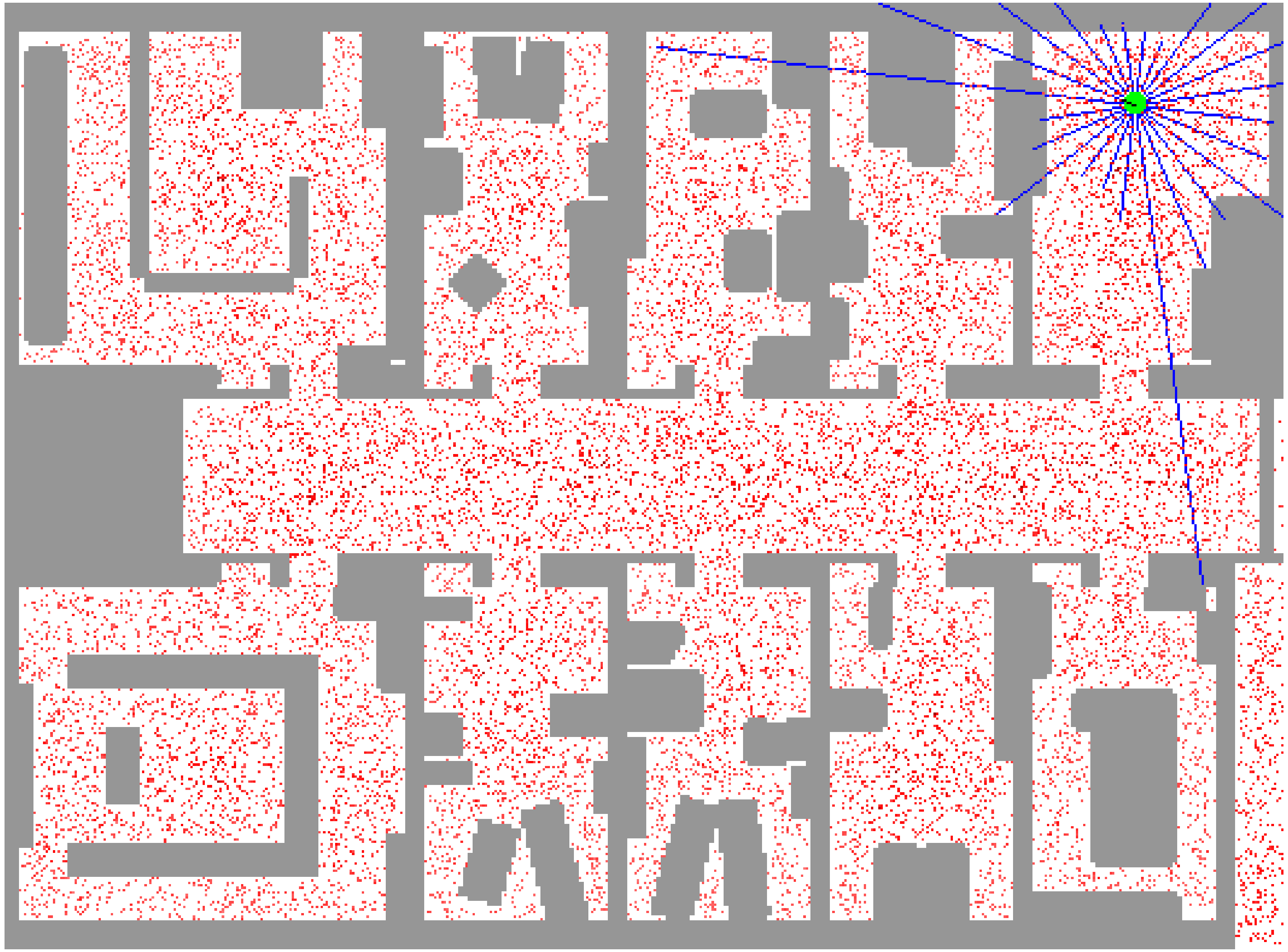
Robot Localization

- 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



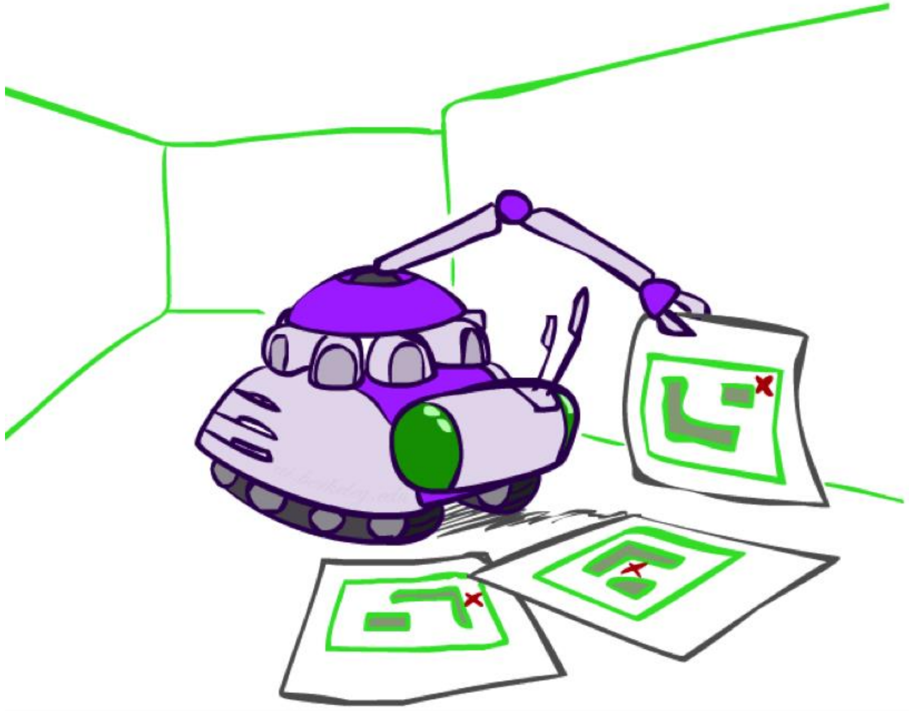
Particle Filter Localization (Sonar)

Particle Filter Localization (Laser)

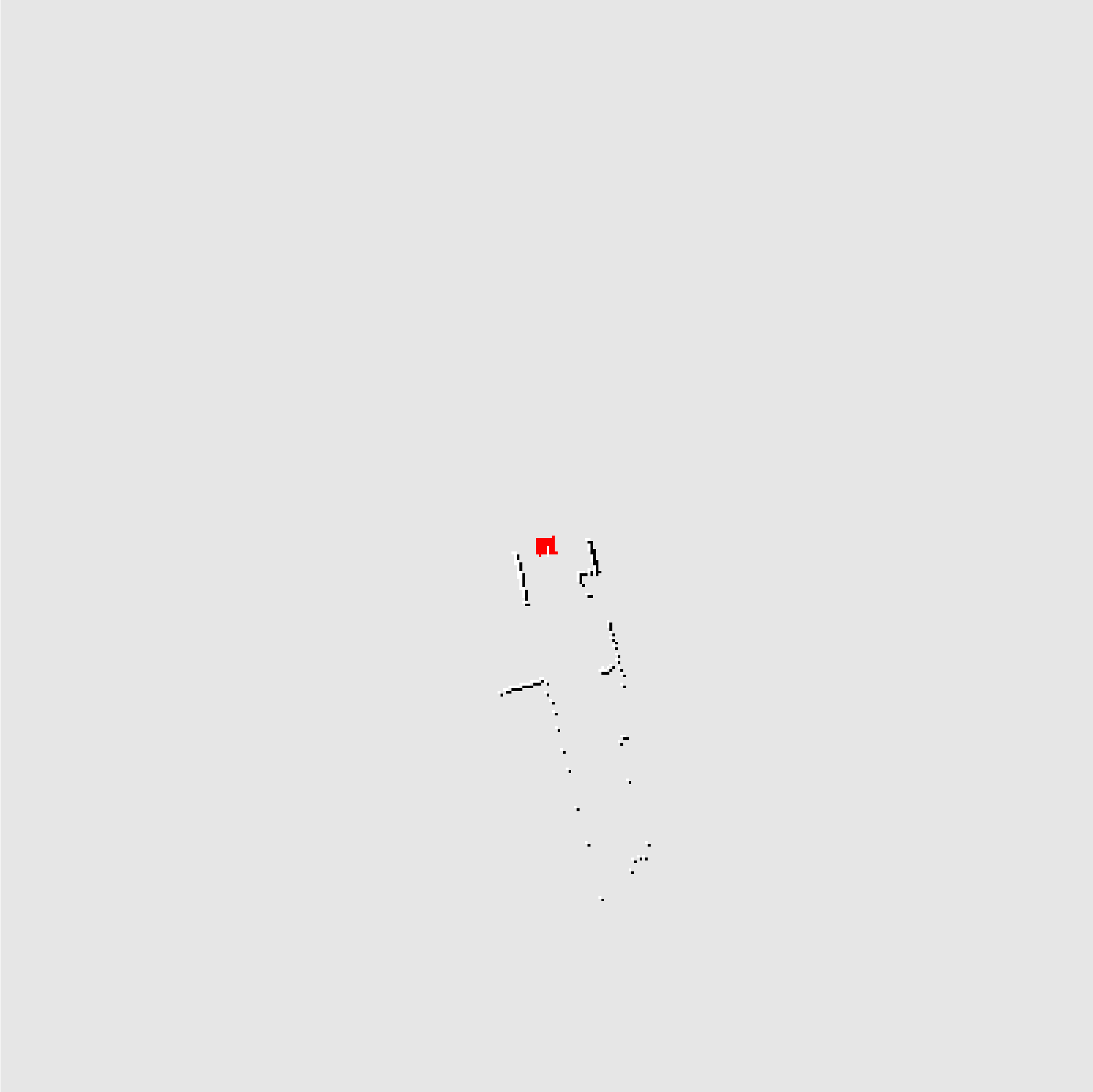


Robot Mapping

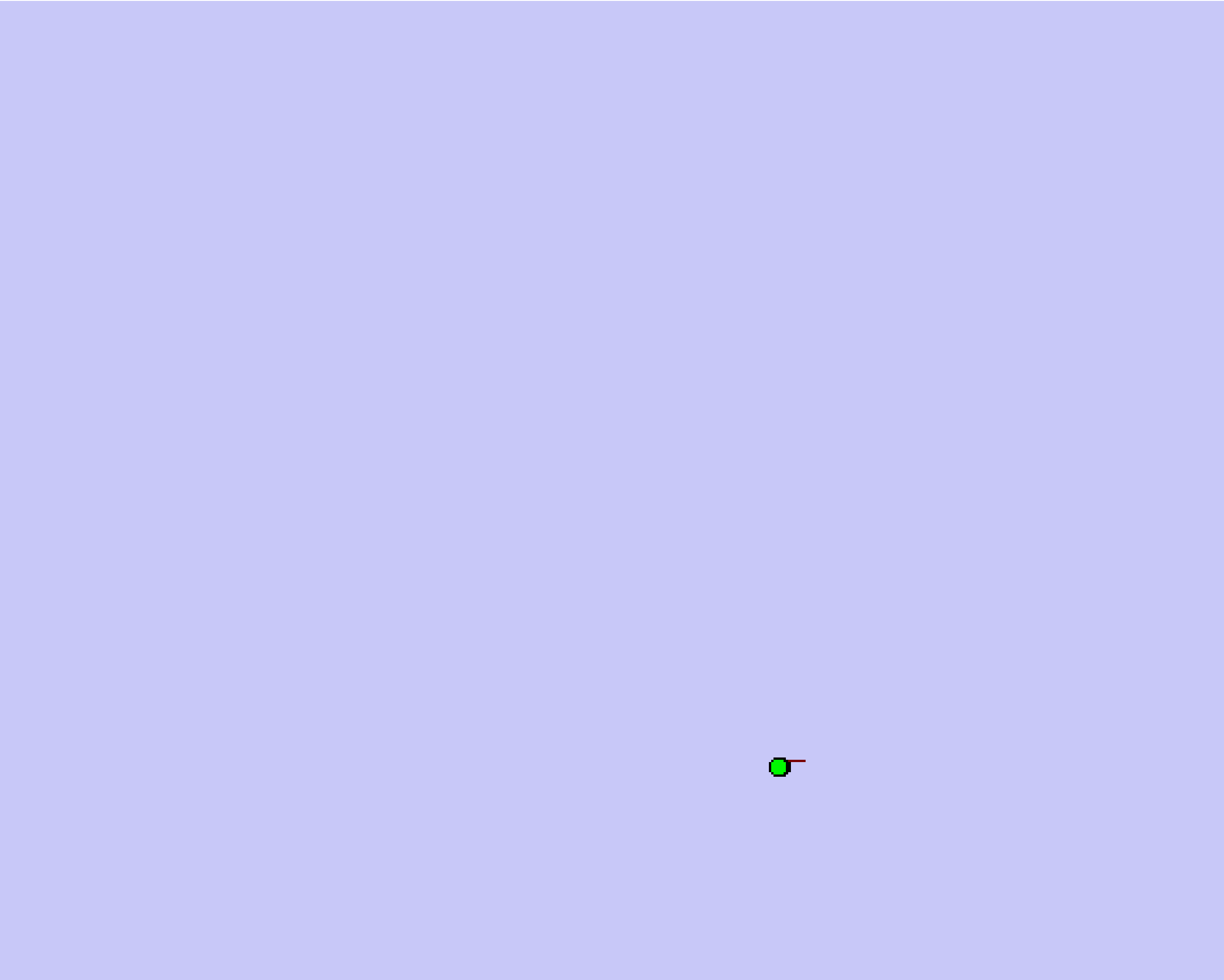
- SLAM: Simultaneous Localization And Mapping
  - We do not know the map or our location
  - State consists of position AND map!
  - Main techniques: Kalman filtering (Gaussian HMMs) and particle methods



Particle Filter SLAM – Video 1



Particle Filter SLAM – Video 2



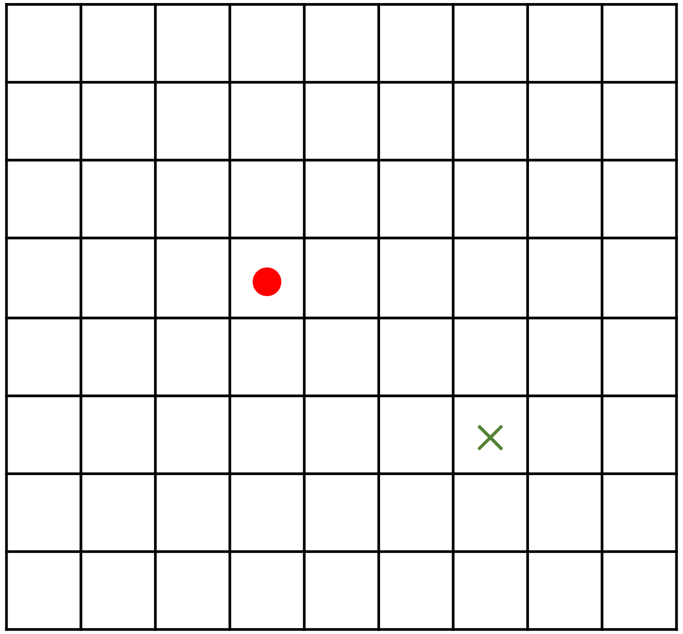
Particle Filtering

Localization: https://www.youtube.com/watch?v=NrzmH_yerBU&ab_channel=MATLAB

SLAM: https://www.youtube.com/watch?v=saVZtgPyyJQ&ab_channel=MATLAB

Some Failure Modes of Particle Filtering

Too few particles



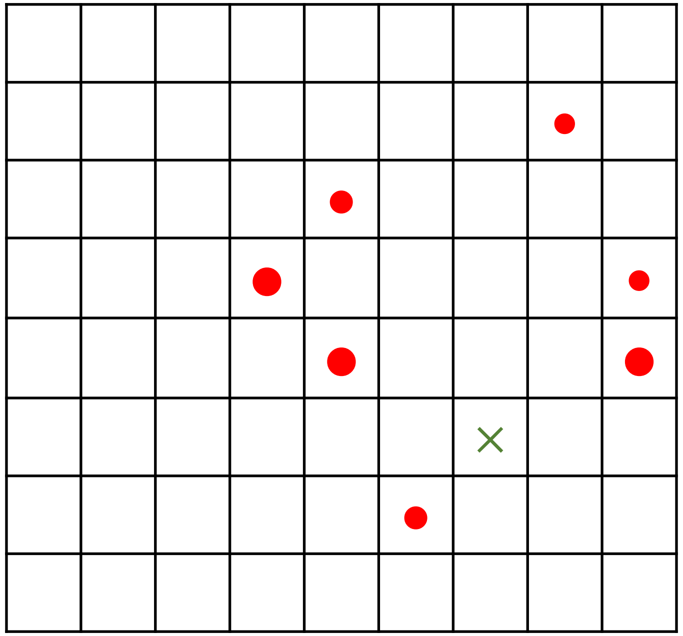
● Particle

× True location

→ The particle has to be dense enough to cover the true state

Some Failure Modes of Particle Filtering

Moderate number of particles but very static state transition



● Particle

× True location

Suppose every state always transitions to itself.

→ All particles and the true location will never move.

→ After several rounds of re-sampling, particles will accumulate to a single position.