Markov Chains and Hidden Markov Models

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Fall 2023

Soleymani

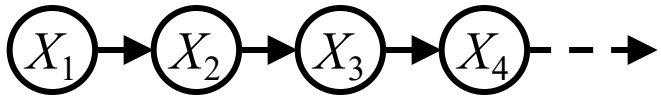
Slides are based on Klein and Abdeel, CS188, UC Berkeley.

Reasoning over Time or Space

- 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 (or space) into our models

Markov Models (aka Markov chain/process)

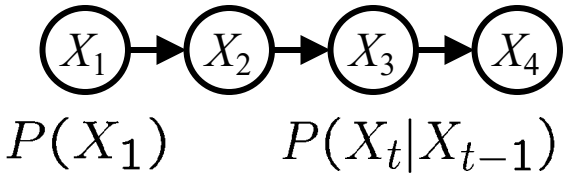
- Value of X at a given time is called the state (usually discrete, finite)



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

- The transition model $P(X_t|X_{t-1})$ specifies how the state evolves over time
- Stationarity assumption: transition probabilities the same at all times
- Same as MDP transition model, but no choice of action
- 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

Joint Distribution of a Markov Model



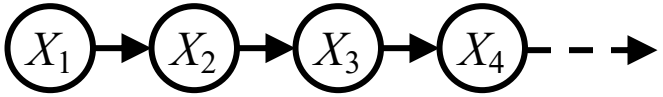
- Joint distribution:

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_2)P(X_4|X_3)$$

- More generally:

$$P(X_1, X_2, \ldots, X_T) = P(X_1)P(X_2|X_1)P(X_3|X_2)\ldots P(X_T|X_{T-1})$$
$$= P(X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})$$

Chain Rule and Markov Models



- From the chain rule, every joint distribution over $X_1, X_2, \ldots, X_T$ can be written as:

$$P(X_1, X_2, \ldots, X_T) = P(X_1) \prod_{t=2}^{T} P(X_t | X_1, X_2, \ldots, X_{t-1})$$

- Assuming that for all *t*:

$$X_t \perp\!\!\!\perp X_1, \ldots, X_{t-2} \mid X_{t-1}$$

gives us the expression posited on the earlier slide:

$$P(X_1, X_2, \ldots, X_T) = P(X_1) \prod_{t=2}^{T} P(X_t | X_{t-1})$$

Markov Models

- Explicit assumption for all t: $X_t \perp\!\!\!\perp X_1, \ldots, X_{t-2} \mid X_{t-1}$
- Consequence, joint distribution can be written as:

$$P(X_1, X_2, \ldots, X_T) = P(X_1)P(X_2|X_1)P(X_3|X_2)\ldots P(X_T|X_{T-1})$$
$$= P(X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})$$

- Implied conditional independencies:
  - Past variables independent of future variables given the present

i.e., if $t_1 < t_2 < t_3$ or $t_1 > t_2 > t_3$ then: $X_{t_1} \perp\!\!\!\perp X_{t_3} \mid X_{t_2}$

- Additional explicit assumption: $P(X_t \mid X_{t-1})$ is the same for all t

Conditional Independence

- Basic conditional independence:
  - Past and future independent of the present
  - Each time step only depends on the previous
  - This is called the (first order) Markov property

- Note that the chain is just a (growable) BN
  - We can always use generic BN reasoning on it if we truncate the chain at a fixed length

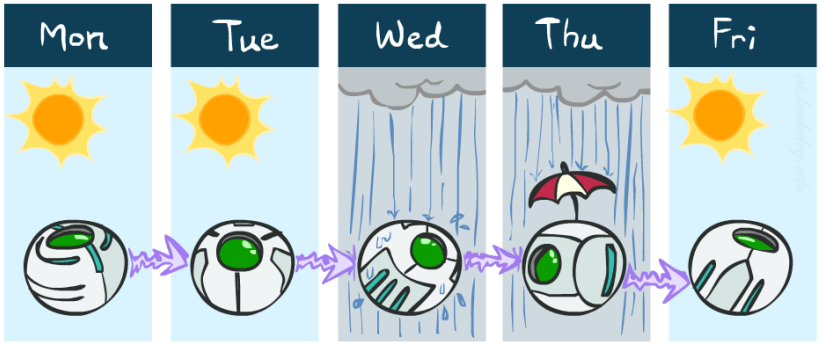
Example Markov Chain: Weather

- States: X = {rain, sun}

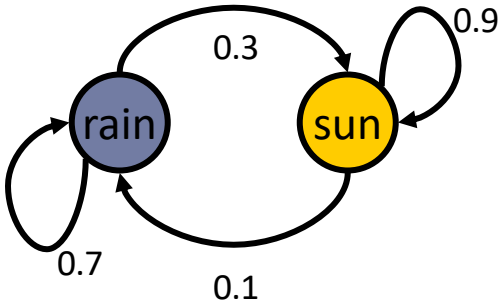
- Initial distribution: 1.0 sun

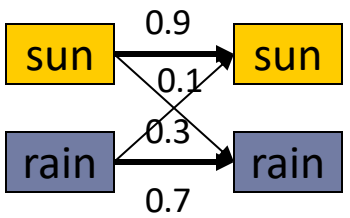
- CPT $P(X_t \mid X_{t-1})$:

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



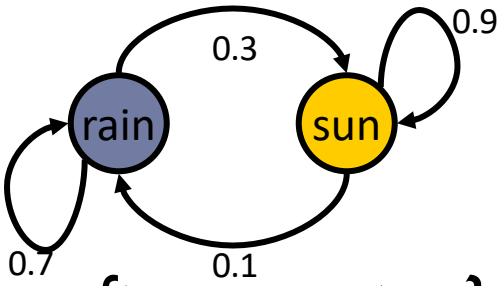
Two new ways of representing the same CPT





Example Markov Chain: Weather

- Initial distribution: 1.0 sun



- What is the probability distribution after one step?

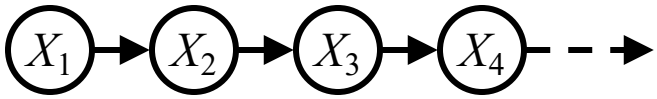
$$P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain})$$

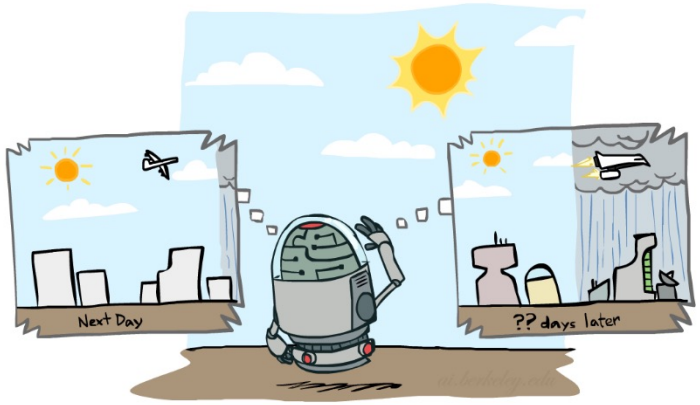
$$0.9 \cdot 1.0 + 0.3 \cdot 0.0 = 0.9$$

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

Mini-Forward Algorithm

- Question: What's P(X) on some day t?





$$P(x_1) = \text{known}$$

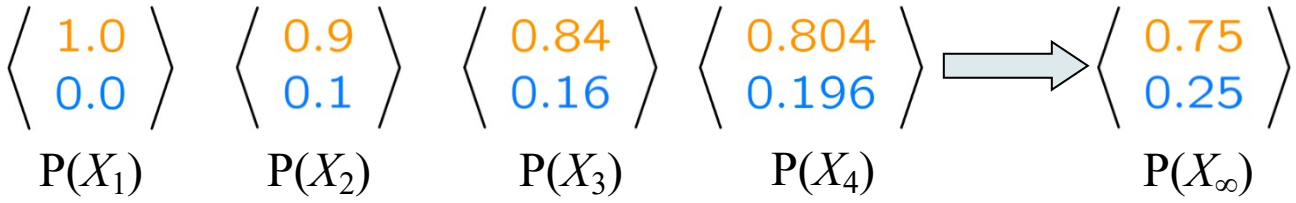
$$P(x_t) = \sum_{x_{t-1}} P(x_{t-1}, x_t)$$

$$= \sum_{x_{t-1}} P(x_t \mid x_{t-1}) P(x_{t-1})$$

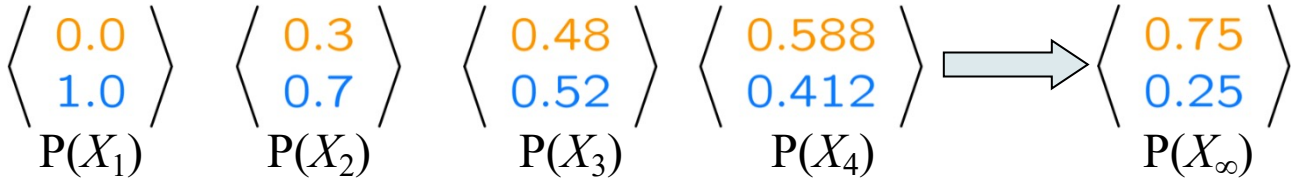
Forward simulation

Example Run of Mini-Forward Algorithm

- From initial observation of sun



- From initial observation of rain



- From yet another initial distribution $P(X_1)$:

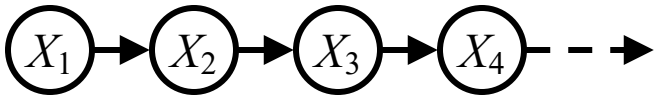
Stationary Distributions

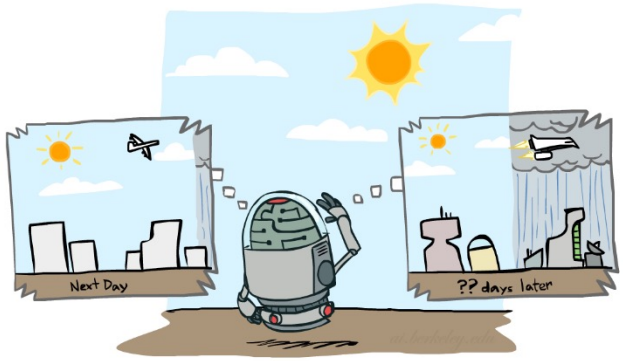
- For most chains:
  - Influence of the initial distribution gets less and less over time.
  - The distribution we end up in is independent of the initial distribution
- Stationary distribution:
  - The distribution we end up with is called the stationary distribution P_∞ of the chain
  - It satisfies

$$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$$

Example: Stationary Distributions

- Question: What's P(X) at time t = infinity?





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

$$P_\infty(sun) = 0.9P_\infty(sun) + 0.3P_\infty(rain)$$
$$P_\infty(rain) = 0.1P_\infty(sun) + 0.7P_\infty(rain)$$

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

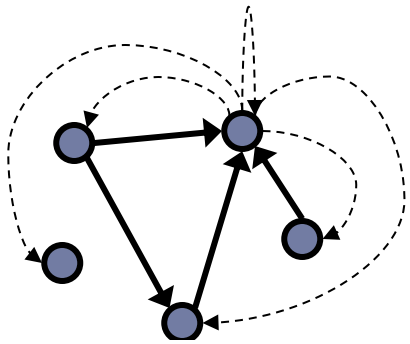
Also: $P_\infty(sun) + P_\infty(rain) = 1$

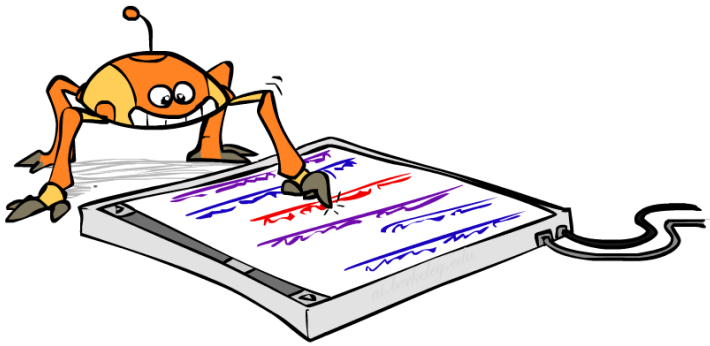
$$P_\infty(sun) = 3/4$$
$$P_\infty(rain) = 1/4$$

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

Application of Stationary Distribution: Web Link Analysis

- PageRank over a web graph
  - Each web page is a state
  - Initial distribution: uniform over pages
  - Transitions:
    - With prob. c, uniform jump to a random page (dotted lines, not all shown)
    - With prob. 1-c, follow a random outlink (solid lines)





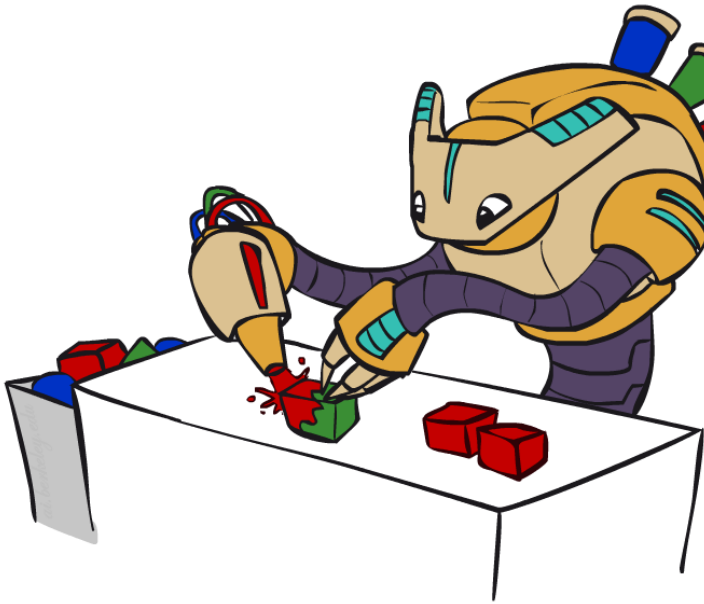
- Stationary distribution
  - Will spend more time on highly reachable pages
  - E.g. many ways to get to the Acrobat Reader download page
  - Somewhat robust to link spam
  - Google 1.0 returned the set of pages containing all your keywords in decreasing rank, now all search engines use link analysis along with many other factors (rank actually getting less important over time)

Application of Stationary Distributions: Gibbs Sampling*

- Each joint instantiation over all hidden and query variables is a state: $\{X_1, \ldots, X_n\} = H \cup Q$
- Transitions:
  - With probability 1/n resample variable X_j according to

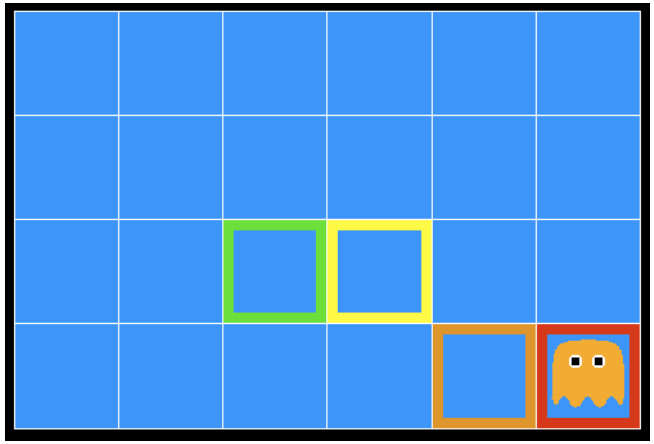
$$P(X_j \mid x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n, e_1, \ldots, e_m)$$

- Stationary distribution:
  - Conditional distribution $P(X_1, X_2, \ldots, X_n \mid e_1, \ldots, e_m)$
  - Means that when running Gibbs sampling long enough we get a sample from the desired distribution
  - Requires some proof to show this is true!



Inference in Ghostbusters

- A ghost is in the grid somewhere
- Sensor readings tell how close a square is to the ghost
  - On the ghost: red
  - 1 or 2 away: orange
  - 3 or 4 away: yellow
  - 5+ away: green



- Sensors are noisy, but we know P(Color | Distance)

P(red | 3)	P(orange | 3)	P(yellow | 3)	P(green | 3)
0.05	0.15	0.5	0.3

Video of Demo Ghostbusters Basic Dynamics

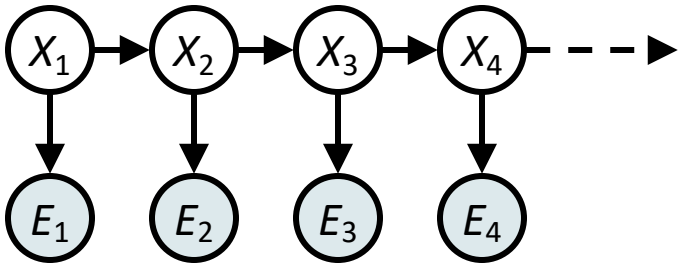
Video of Demo Ghostbusters Circular Dynamics

Video of Demo Ghostbusters Whirlpool Dynamics

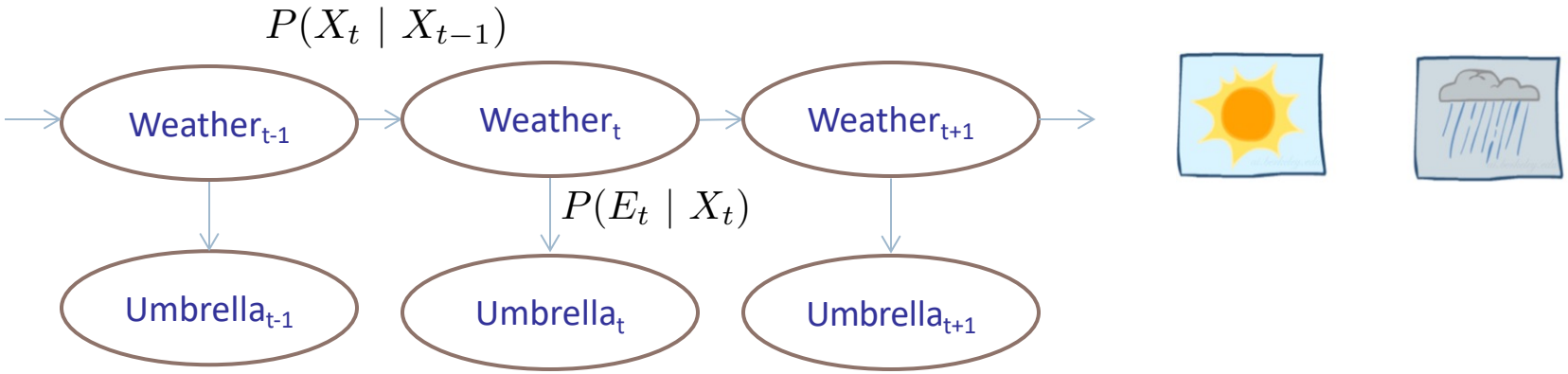
Hidden Markov Models

Hidden Markov Models

- Markov chains not so useful for most agents
  - Need observations to update your beliefs
- 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



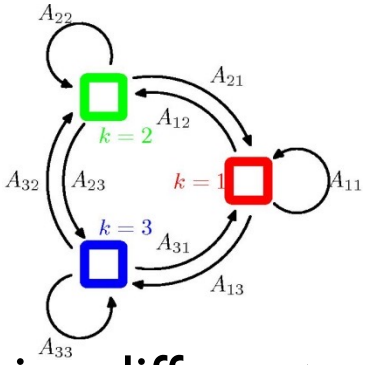
- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t \mid X_{t-1})$
  - Emissions: $P(E_t \mid X_t)$

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

HMM: Probabilistic Model

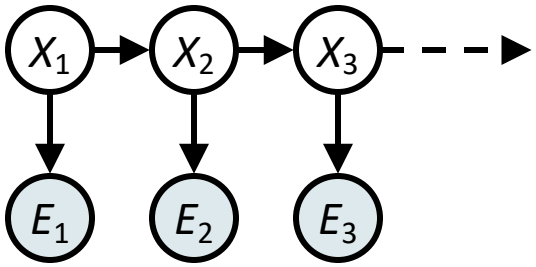
- **Transitional probabilities**: transition probabilities between states
  - $A_{ij} \equiv P(X_t = j | X_{t-1} = i)$



- **Initial state distribution:** start probabilities in different states
  - $\pi_i \equiv P(X_1 = i)$

- **Observation model**: Emission probabilities associated with each state
  - $P(E_t | X_t)$

Joint Distribution of an HMM



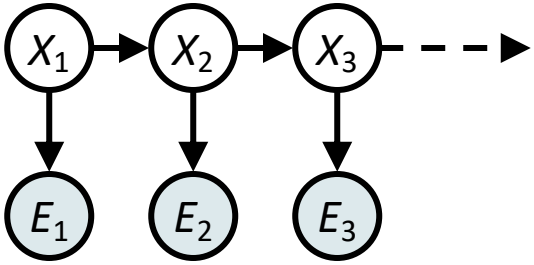
- Joint distribution:

$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1)P(E_2|X_2)P(X_3|X_2)P(E_3|X_3)$$

- More generally:

$$P(X_1, E_1, \ldots, X_T, E_T) = P(X_1)P(E_1|X_1)\prod_{t=2}^{T} P(X_t|X_{t-1})P(E_t|X_t)$$

Conditional Independencies



- State independent of all past states and all past evidence given the previous state, i.e.:

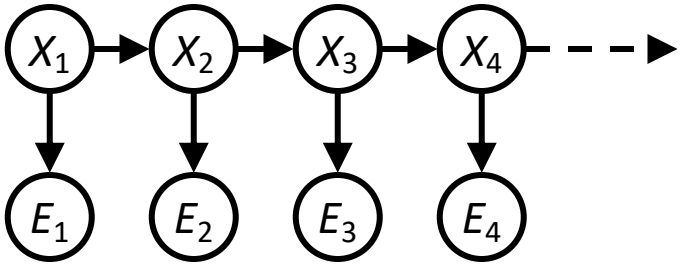
$$X_t \perp\!\!\!\perp X_1, E_1, \ldots, X_{t-2}, E_{t-2}, E_{t-1} \mid X_{t-1}$$

- Evidence is independent of all past states and all past evidence given the current state, i.e.:

$$E_t \perp\!\!\!\perp X_1, E_1, \ldots, X_{t-2}, E_{t-2}, X_{t-1}, E_{t-1} \mid X_t$$

Conditional Independence

- HMMs have two important independence properties:
  - Markov hidden process: future depends on past via the present
  - Current observation independent of all else given current state



- Quiz: does this mean that evidence variables are guaranteed to be independent?
  - [No, they tend to correlated by the hidden state]

Real HMM Examples

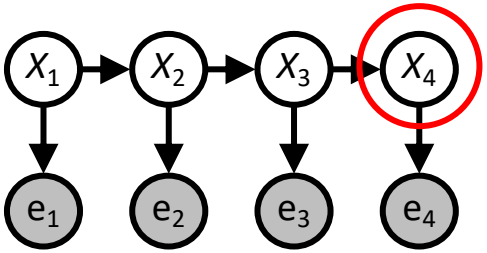
- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options
- Robot tracking:
  - Observations 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

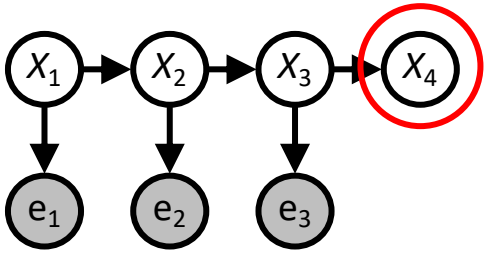
- ***Filtering***: $P(X_t|e_{1:t})$
  - ***belief state***—input to the decision process of a rational agent
- ***Prediction***: $P(X_{t+k}|e_{1:t})$ for $k > 0$
  - evaluation of possible action sequences; like filtering without the evidence
- ***Smoothing***: $P(X_k|e_{1:t})$ for $0 \leq k < t$
  - 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

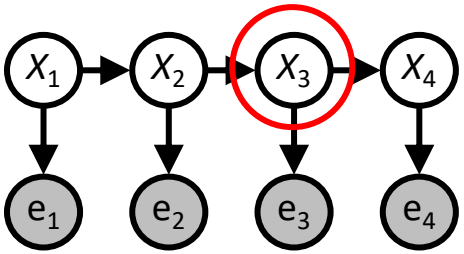
Filtering: P(Xt|e1:t)



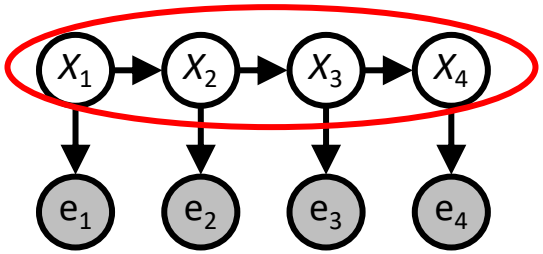
Prediction: P(Xt+k|e1:t)



Smoothing: P(Xk|e1:t), k<t



Explanation: P(X1:t|e1:t)



Filtering / Monitoring

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

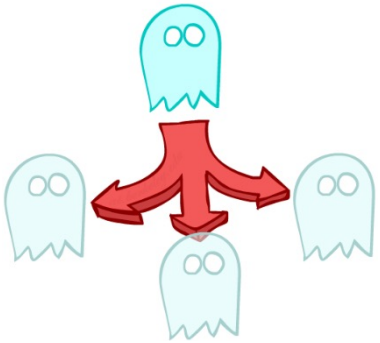
- We start with $\boldsymbol{f}_0$ in an initial setting, usually uniform

- Filtering is a fundamental task in engineering and science

- The Kalman filter (continuous variables, linear dynamics, Gaussian noise) was invented in 1960 and used for trajectory estimation in the Apollo program; core ideas used by Gauss for planetary observations; ***788,000*** papers on Google Scholar

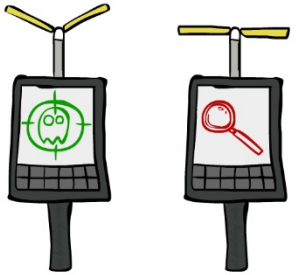
Example: Ghostbusters HMM

- $P(X_1)$ = uniform
- $P(X|X')$ = usually move clockwise, but sometimes move in a random direction or stay in place
- $P(R_{ij}|X)$ = same sensor model as before: red means close, green means far away.



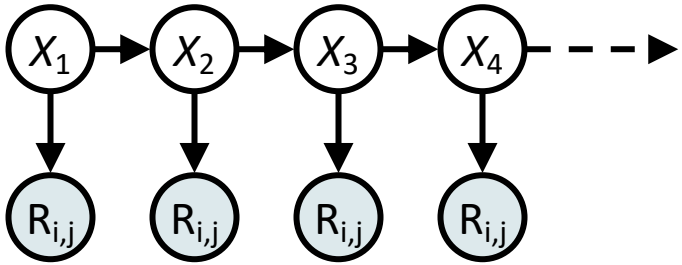
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$P(X_1)$



1/6	1/6	1/2
0	1/6	0
0	0	0

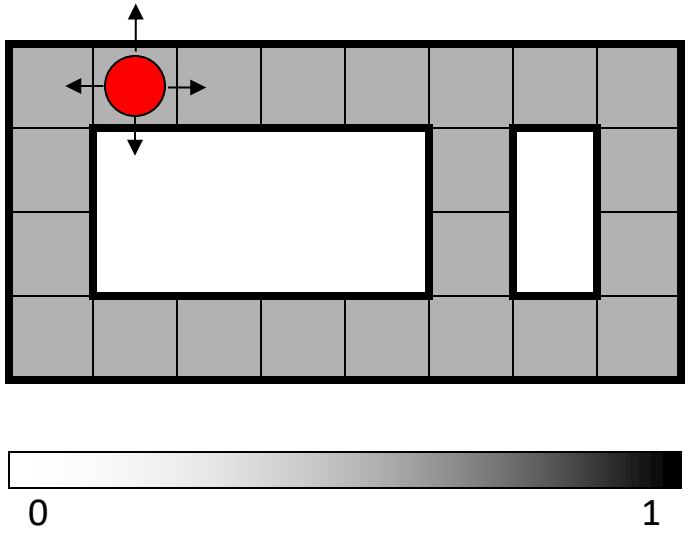
$P(X|X'=<1,2>)$

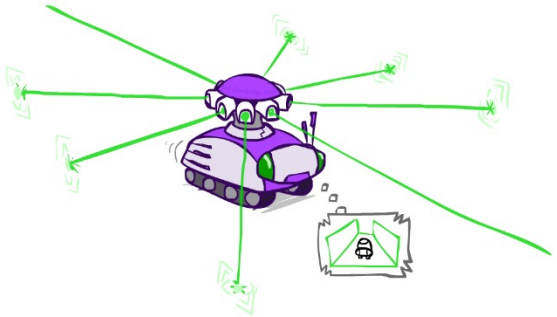


Video of Demo Ghostbusters – Circular Dynamics -- HMM

Example: Robot Localization

Example from
Michael Pfeiffer



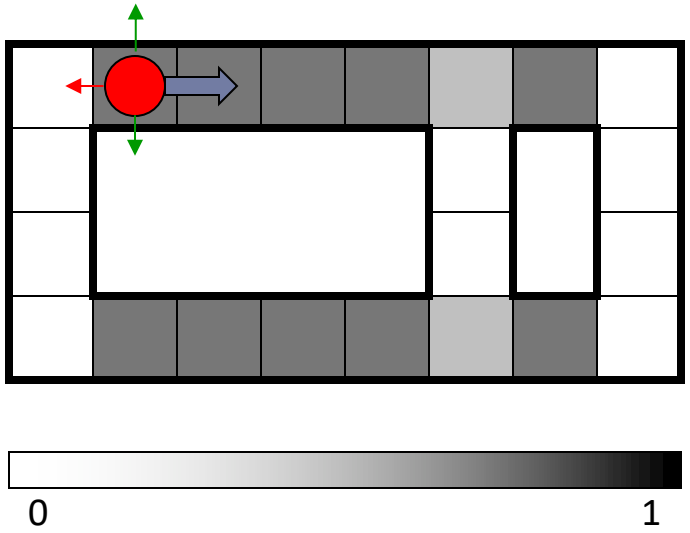


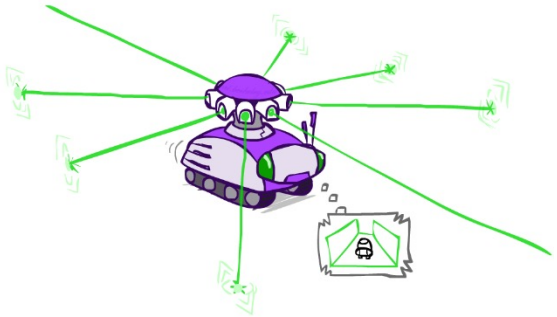
t=0

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

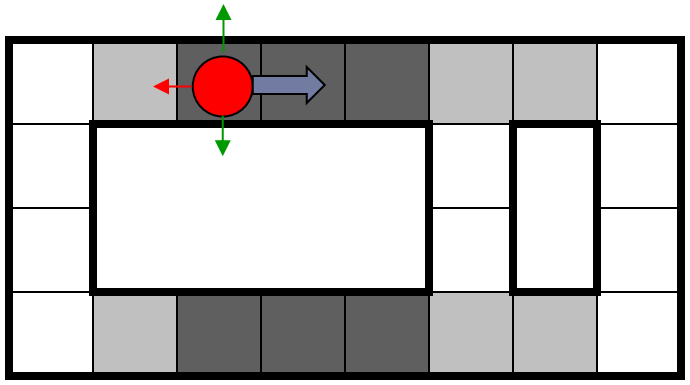


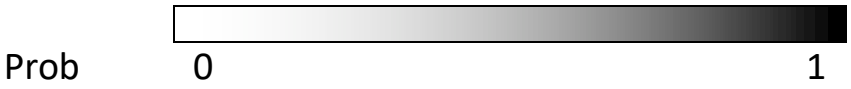


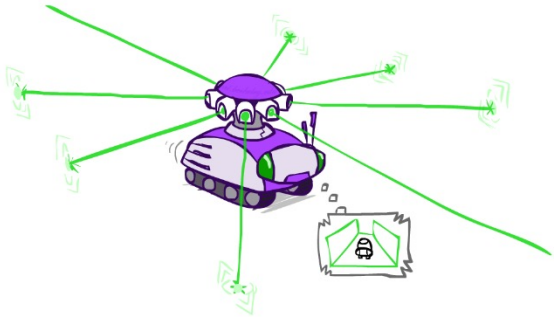
t=1

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

Example: Robot Localization

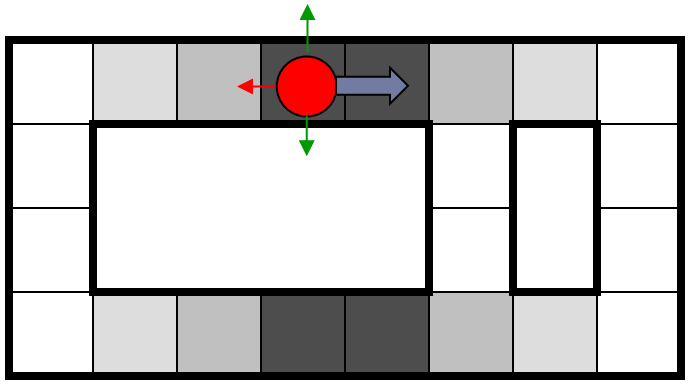


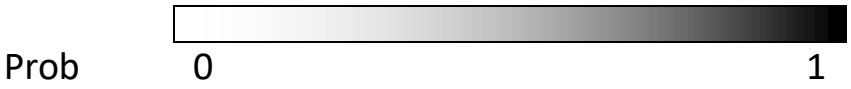


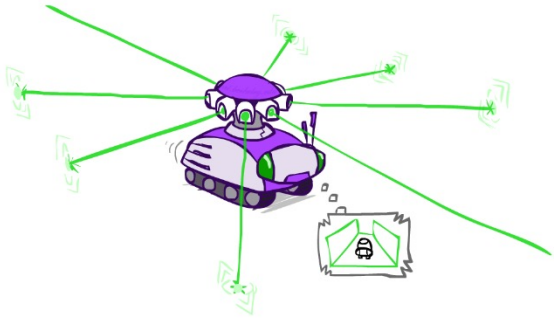


t=2

Example: Robot Localization

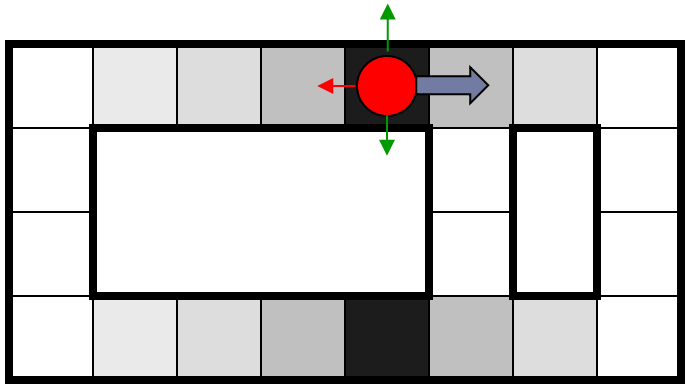


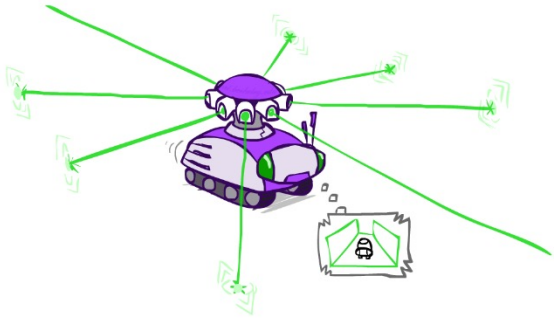




t=3

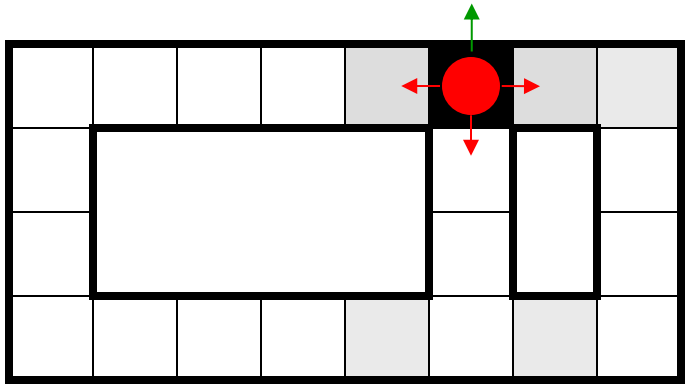
Example: Robot Localization

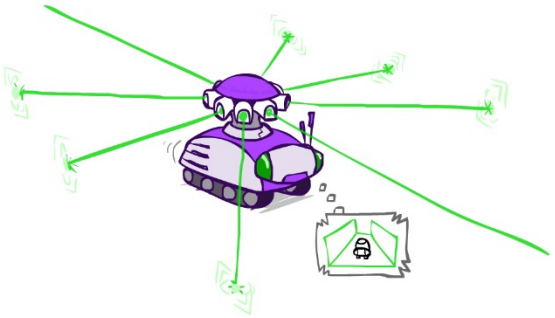




t=4

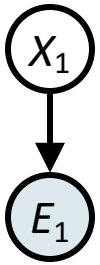
Example: Robot Localization

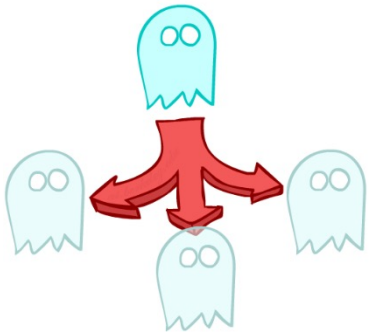


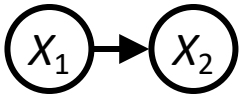


t=5

Inference: Base Cases







$$P(X_1|e_1)$$

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

$$P(X_2)$$

$$\begin{aligned} P(x_2) &= \sum_{x_1} P(x_1, x_2) \\ &= \sum_{x_1} P(x_1)P(x_2|x_1) \end{aligned}$$

Filtering Algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = g(e_{t+1}, P(X_t|e_{1:t}))$

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

$= \alpha\, P(e_{t+1}|X_{t+1}, e_{1:t})\, P(X_{t+1}|\, e_{1:t})$ Bayes' rule

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

$= \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(x_t \mid e_{1:t})\, P(X_{t+1} \mid x_t, e_{1:t})$ Condition on X_t

Normalize Update Predict

Filtering Algorithm

- Aim: devise a ***recursive filtering*** algorithm of the form
  - $P(X_{t+1}|e_{1:t+1}) = g(e_{t+1}, P(X_t|e_{1:t}))$

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

$= \alpha\, P(e_{t+1}|X_{t+1}, e_{1:t})\, P(X_{t+1}| e_{1:t})$ Bayes' rule

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

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

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

Filtering Algorithm

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

- $\boldsymbol{b}_{t+1} = FORWARD(\boldsymbol{b}_t, e_{t+1})$ $\boldsymbol{b}_{t+1} = P(X_{t+1}|e_{1:t+1})$
- Cost per time step: $O(|X|^2)$ where $|X|$ is the number of states
- Time and space costs are constant, independent of t
- $O(|X|^2)$ is infeasible for models with many state variables
- We get to invent really cool approximate filtering algorithms

And the Same Thing in Linear Algebra

- Transition matrix T, observation probability vector O_t
  - Observation vector has state likelihoods for E_t
  - E.g., for U_1 = true, $o_1 = \begin{pmatrix} 0.2 \\ 0.9 \end{pmatrix}$
- Filtering algorithm becomes

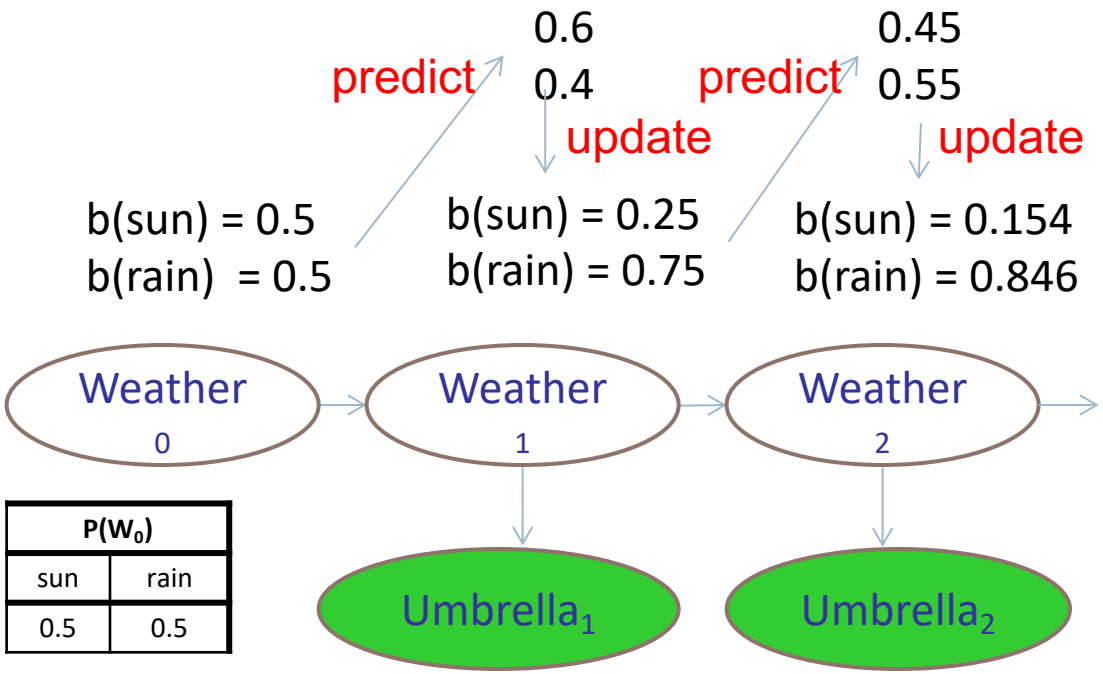
$$\boldsymbol{b'}_t = P(X_{t+1}|e_{1:t}) = T^T \boldsymbol{b}_t$$

$$\boldsymbol{b}_{t+1} = \alpha\, O_{t+1} \odot \boldsymbol{b'}_t$$

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

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

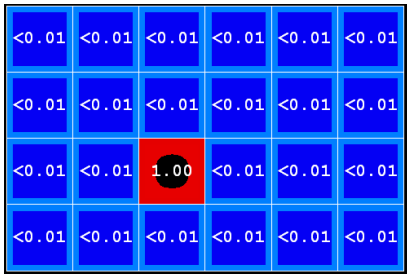
$P(W_0)$	
sun	rain
0.5	0.5

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

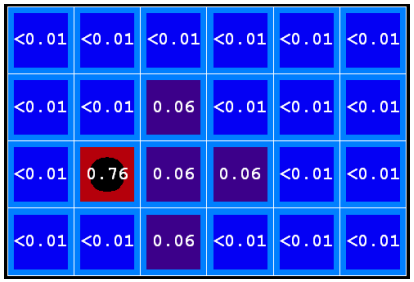
Example: Passage of Time

- As time passes, uncertainty "accumulates"

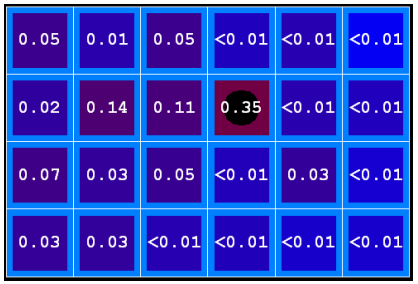
(Transition model: ghosts usually go clockwise)



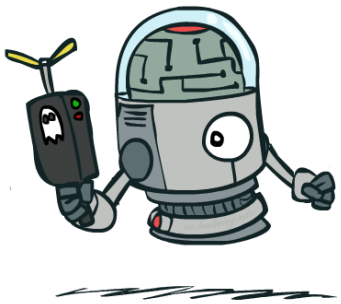
T = 1

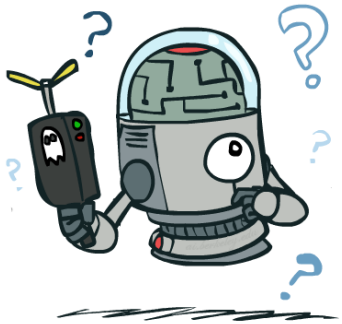


T = 2



T = 5





Example: Observation

- As we get observations, beliefs get reweighted, uncertainty "decreases"

$b'_t = T^\top b_t$

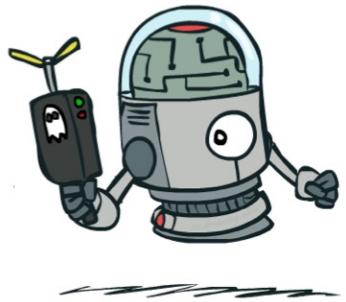
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

$b_{t+1} = \alpha\, O_{t+1} \odot b'_t$

<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



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

Recap: Predict & Update

$b'_t = T^\top b_t$

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

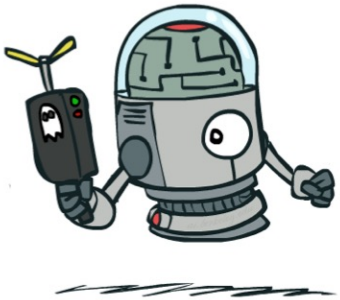
Passage of time
(Before observation)

$b_{t+1} = \alpha\, O_{t+1} \odot b'_t$

<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

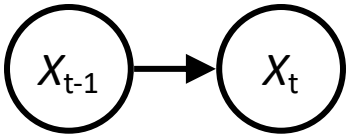
$$b_{t+1} = \alpha\, O_{t+1} \odot b'_t$$



Online Belief Updates

- Every time step, we start with current P(X | evidence)
- We update for time:

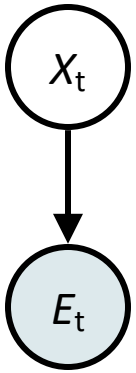
$$P(X_{t+1}|e_{1:t}) = \sum_{x_t} P(x_t \mid e_{1:t})\, P(X_{t+1} \mid x_t)$$



$$\boldsymbol{b}'_t = T^T \boldsymbol{b}_t$$

- We update for evidence:

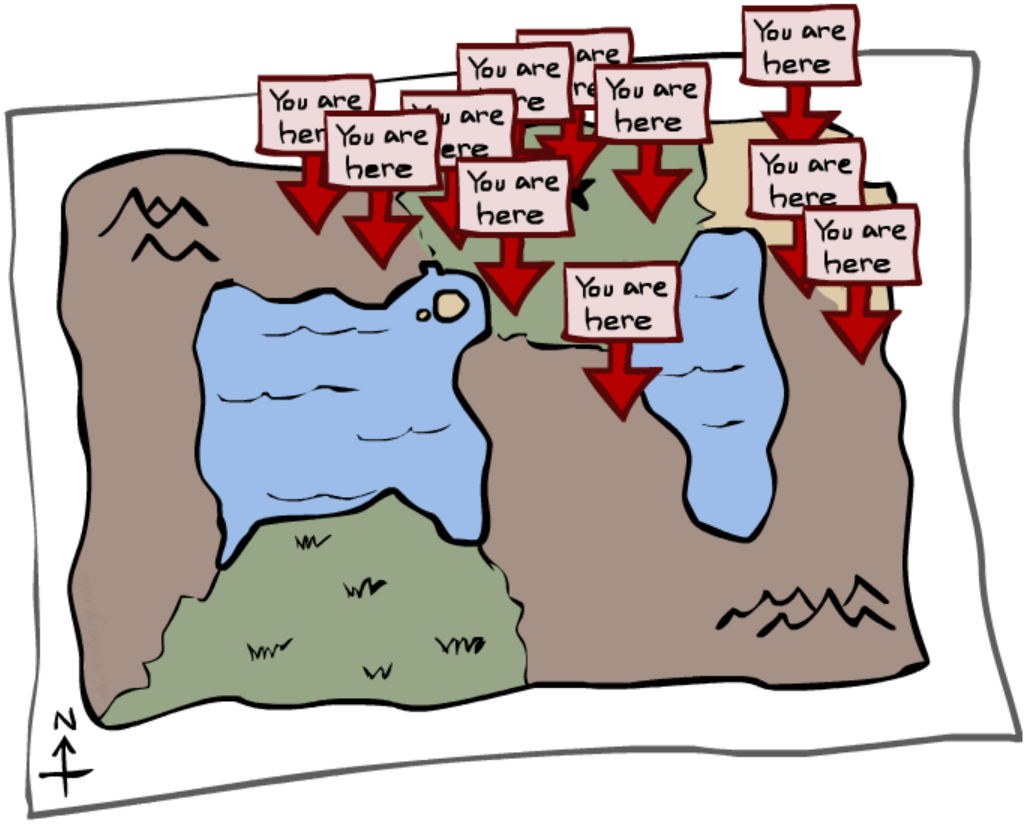
$$P(X_{t+1}|e_{1:t+1}) = \alpha\, P(e_{t+1}|X_{t+1})\, P(X_{t+1}|e_{1:t})$$



$$\boldsymbol{b}_{t+1} = \alpha\, O_{t+1} \odot \boldsymbol{b}'_t$$

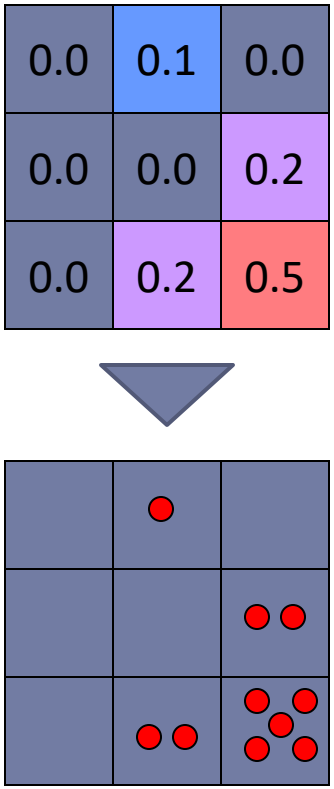
- The forward algorithm does both at once (and doesn't normalize until end)

Particle Filtering



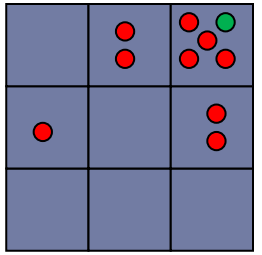
Particle Filtering

- 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
- Usually we want a ***low-dimensional*** marginal
- 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: Prediction Step

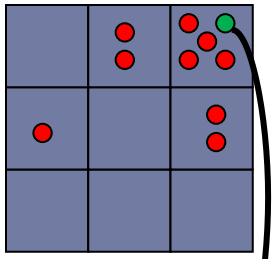
- Each particle is moved by sampling its next position from the transition model

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

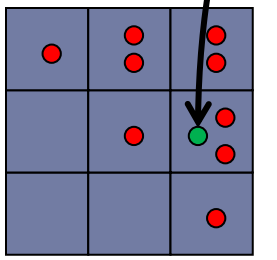
  - This is like prior sampling – samples' frequencies reflect the transition probabilities
  - Here, most samples move clockwise, but some move in another direction or stay in place

- 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)



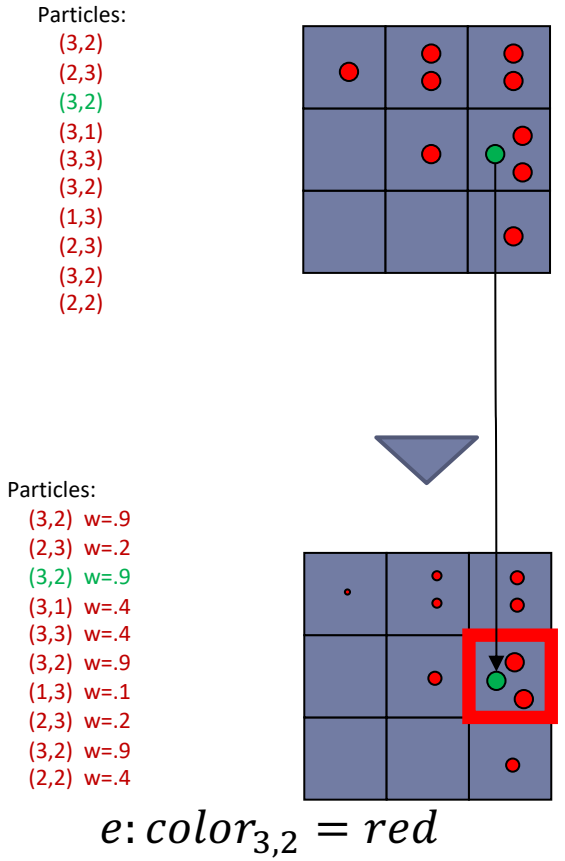
Particle Filtering: Update Step

- 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))

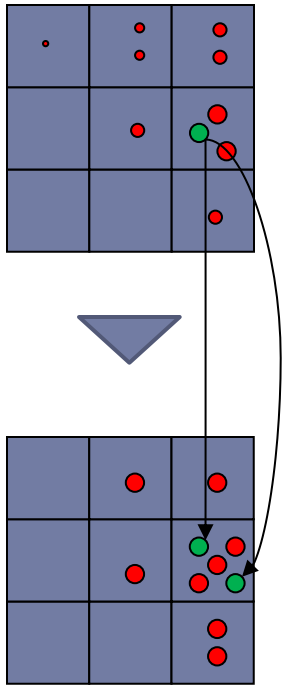


Particle Filtering: Resample

- 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

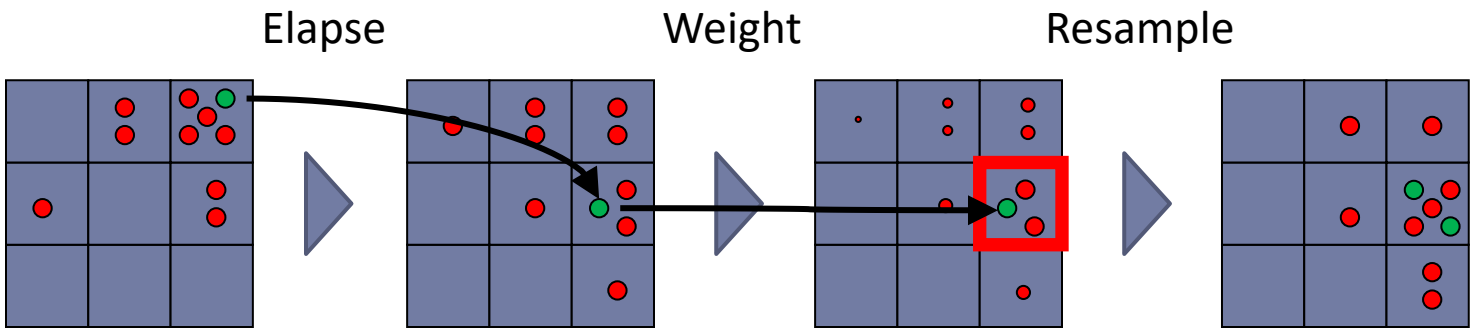
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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



Particle Filtering: Summary

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

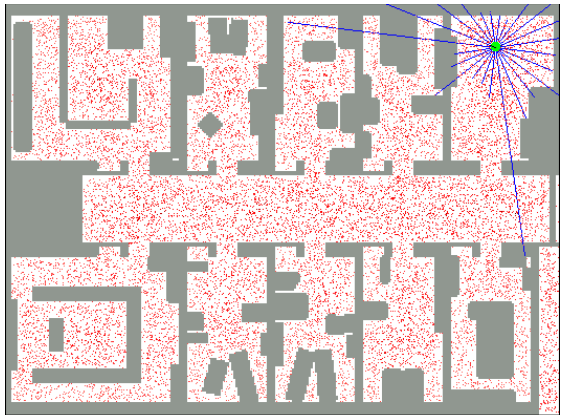


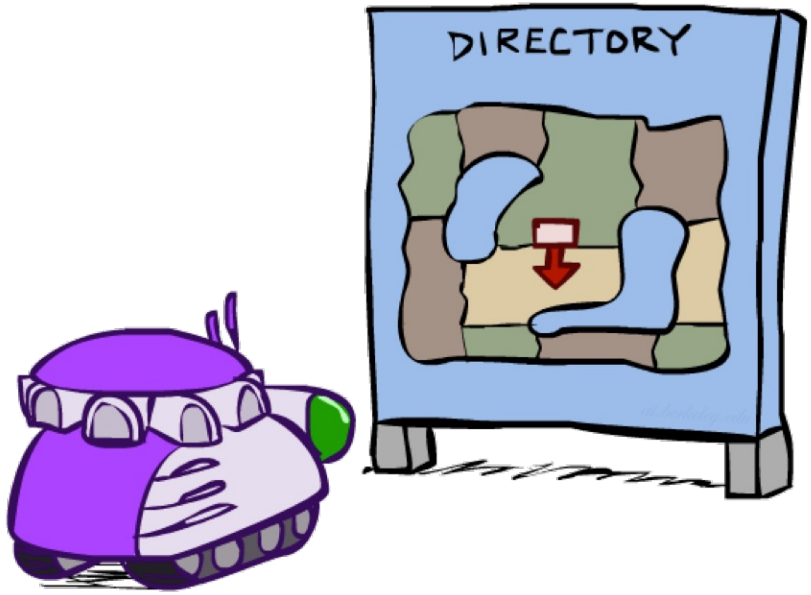
Particles:	Particles:	Particles:	(New) Particles:
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(2,3)	(2,3)	(2,3) w=.2	(2,2)
(3,3)	(3,2)	(3,2) w=.9	(3,2)
(3,2)	(3,1)	(3,1) w=.4	(2,3)
(3,3)	(3,3)	(3,3) w=.4	(3,3)
(3,2)	(3,2)	(3,2) w=.9	(3,2)
(1,2)	(1,3)	(1,3) w=.1	(1,3)
(3,3)	(2,3)	(2,3) w=.2	(2,3)
(3,3)	(3,2)	(3,2) w=.9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

Video of Demo – Moderate Number of Particles

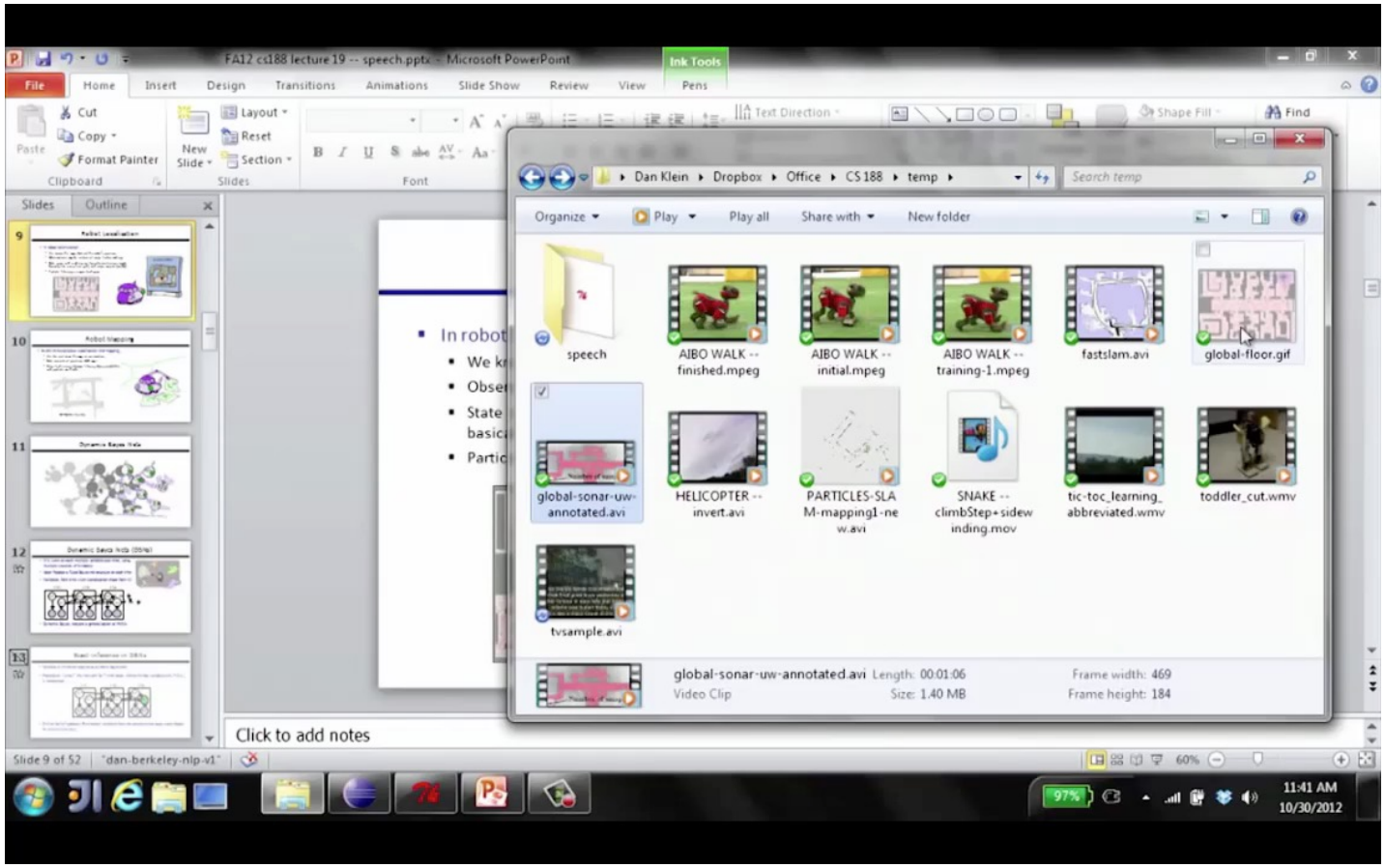
Robot Localization

- 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

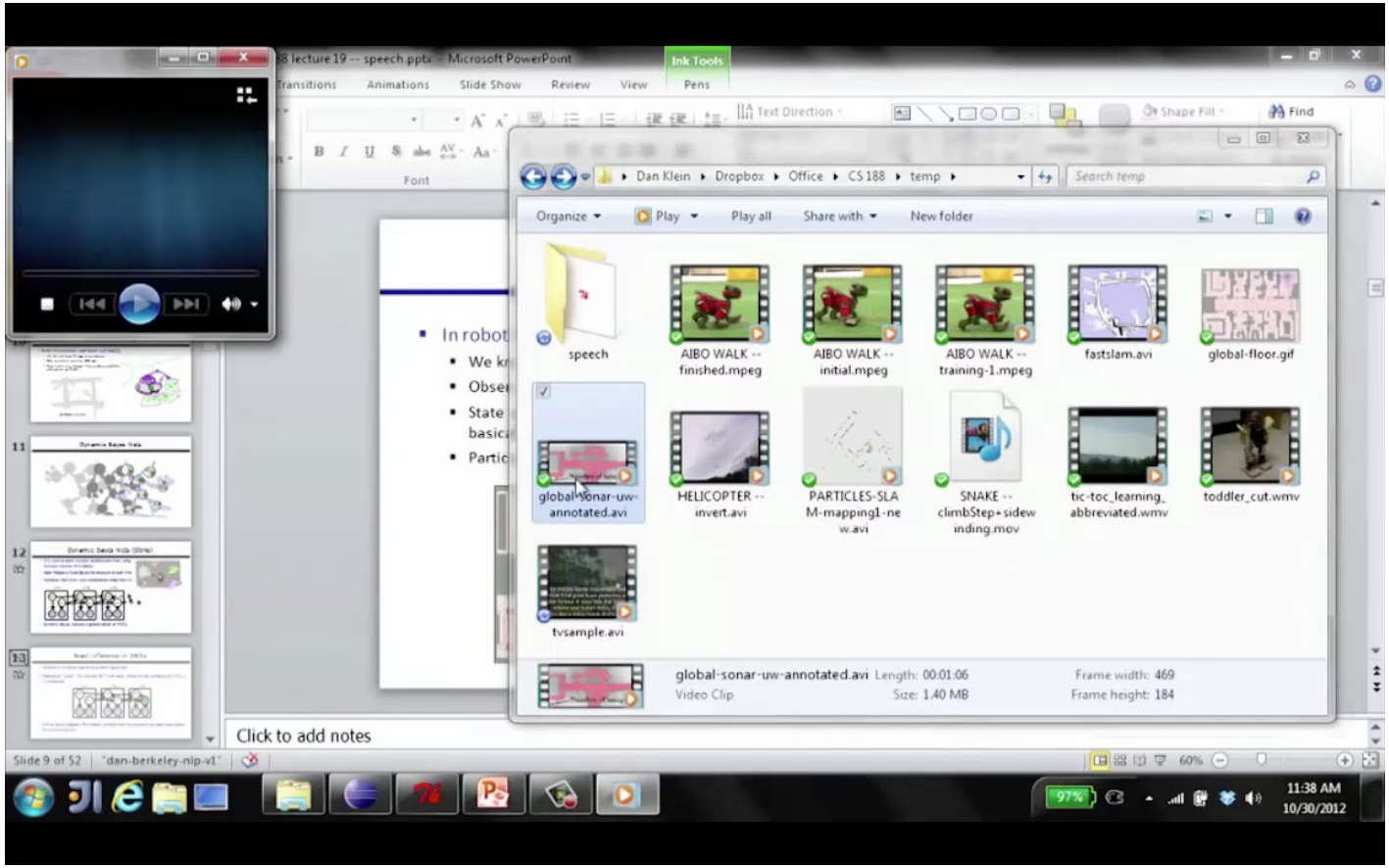




Particle Filter Localization (Laser)

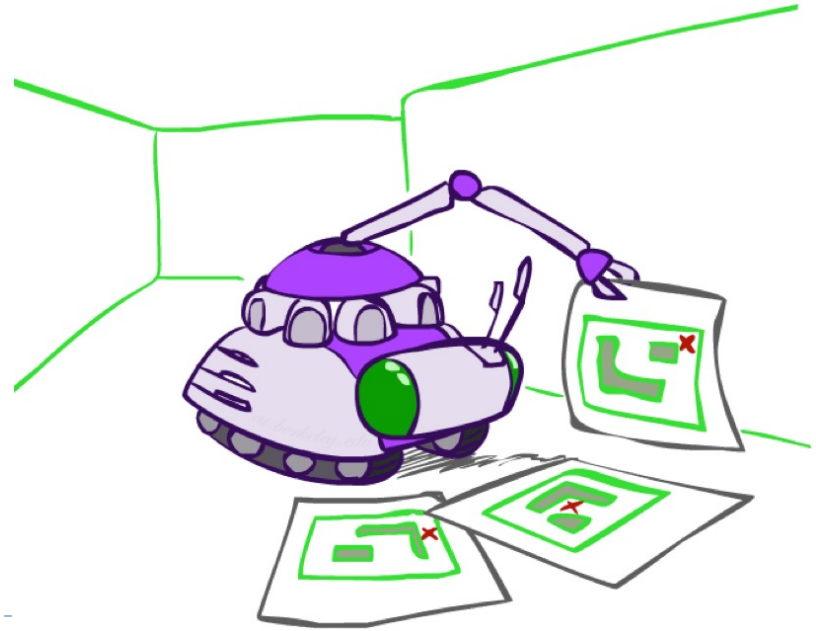


Particle Filter Localization (Sonar)

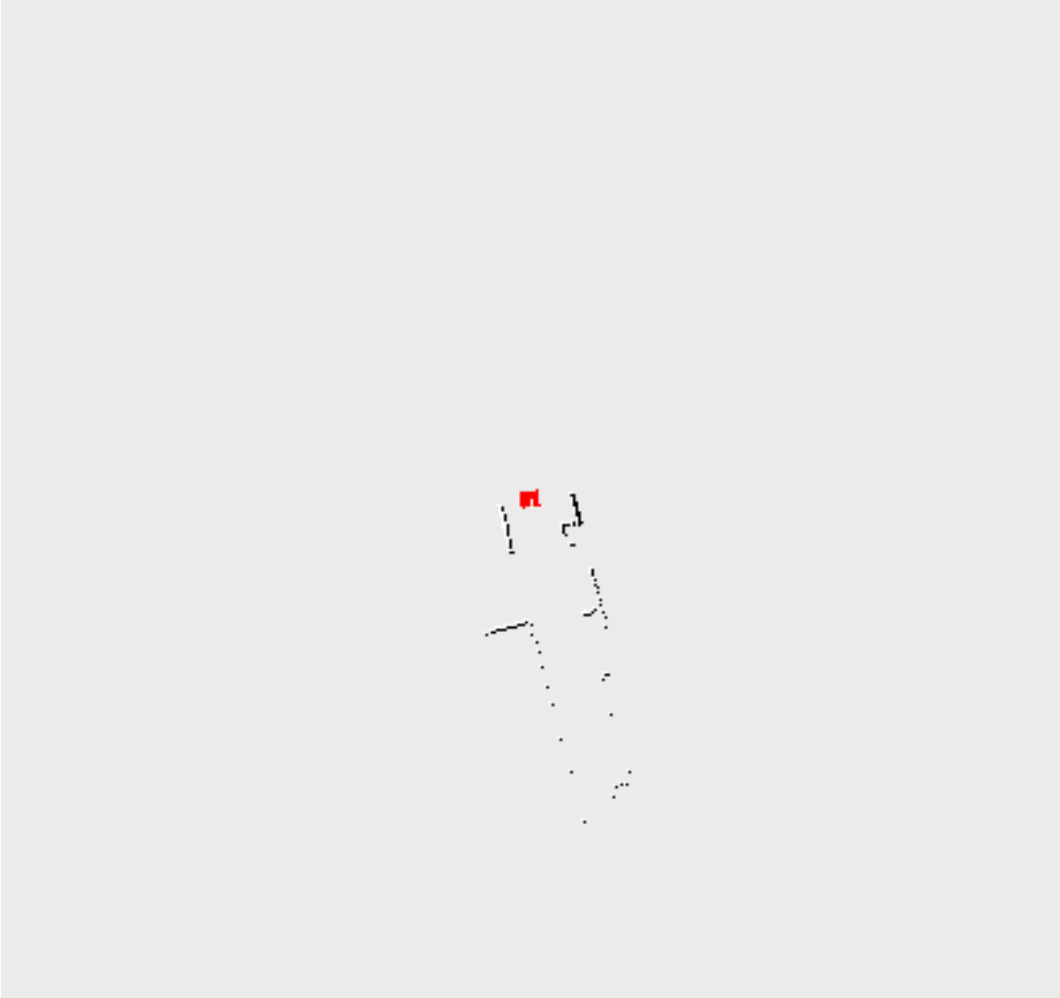


Robot Mapping

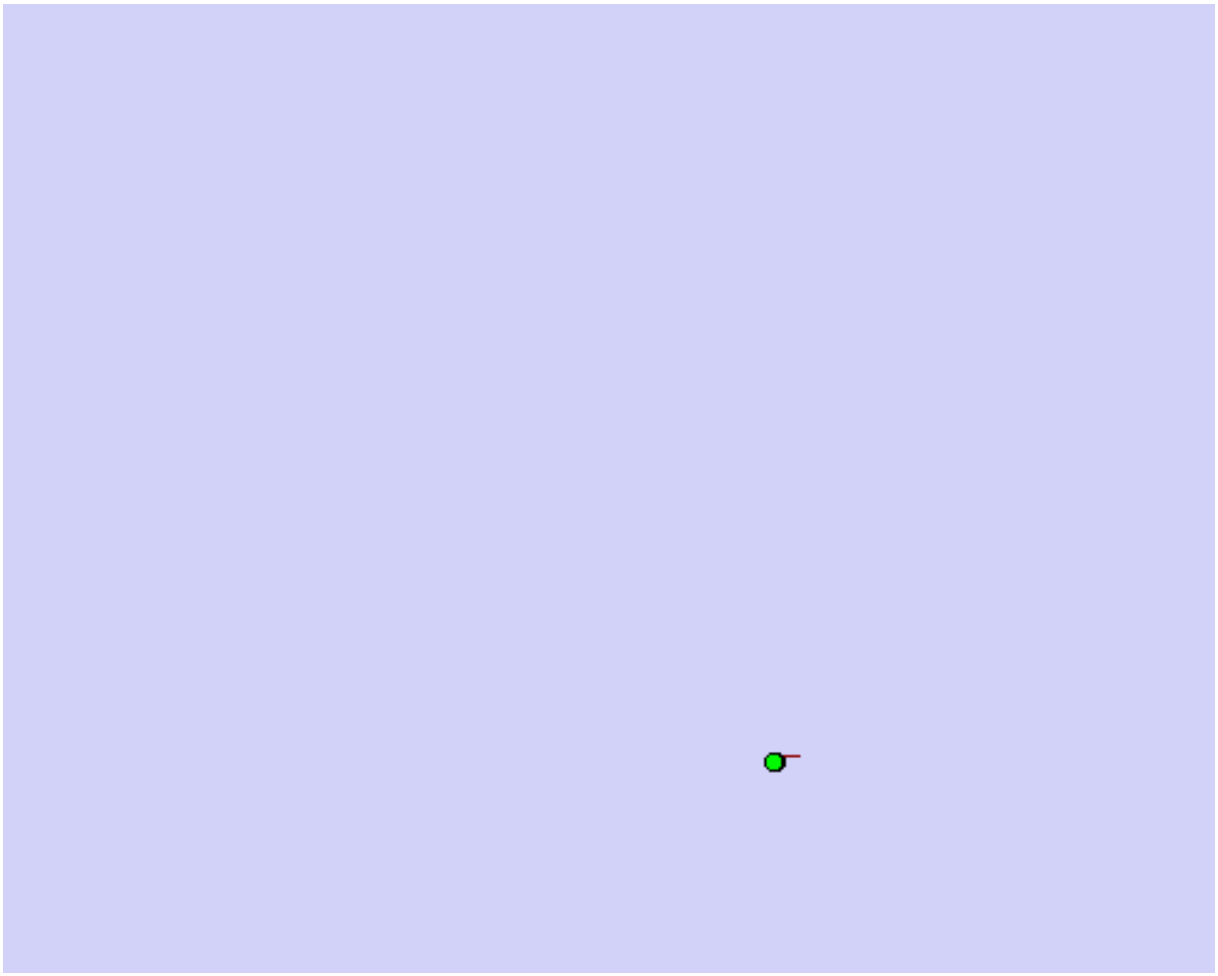
- SLAM: Simultaneous Localization And Mapping
  - Robot does not know map or location
  - Localization: Determine state $x_t^{(j)}$ consists of position+orientation
  - (Each map usually inferred exactly given sampled position+orientation sequence: RBPF)



Particle Filter SLAM – Video 1



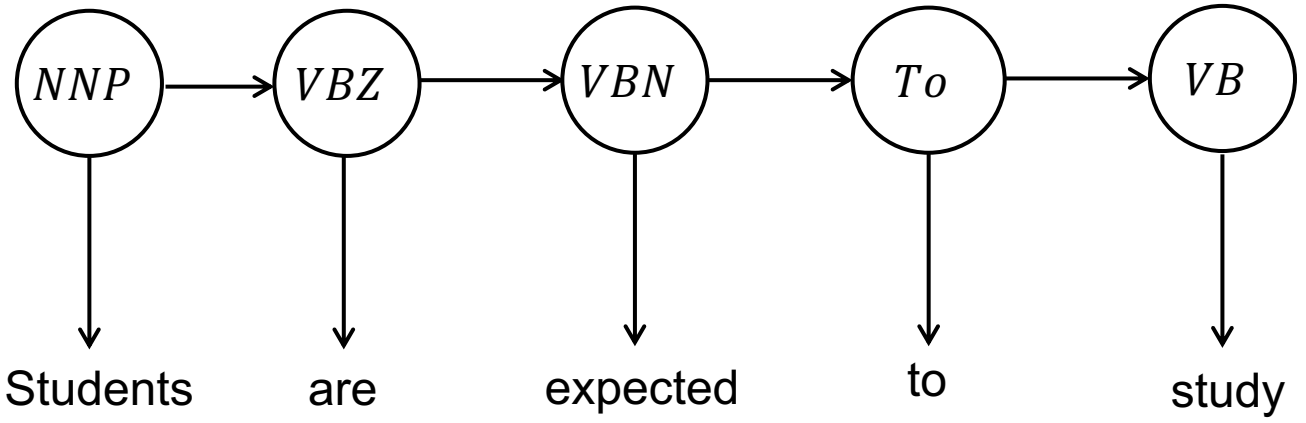
Particle Filter SLAM – Video 2



HMM examples

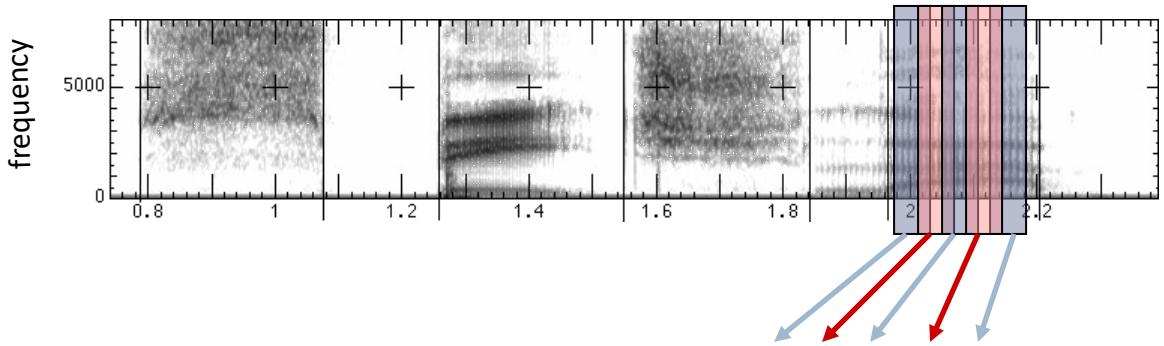
- Some applications of HMM
  - Speech recognition, NLP, activity recognition

- Part-of-speech-tagging



Acoustic Feature Sequence

- Time slices are translated into acoustic feature vectors (~39 real numbers per slice)



…….e_{12} e_{13} e_{14} e_{15} e_{16}…….

- These are the observations E, now we need the hidden states X

Speech State Space

- HMM Specification
  - P(E|X) encodes which acoustic vectors are appropriate for each phoneme (each kind of sound)
  - P(X|X') encodes how sounds can be strung together

- State Space
  - We will have one state for each sound in each word
  - Mostly, states advance sound by sound
  - Build a little state graph for each word and chain them together to form the state space X

Decoding

- Finding the words given the acoustics is an HMM inference problem
- Which state sequence $x_{1:T}$ is most likely given the evidence $e_{1:T}$?

$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T}) \;\; = \arg\max_{x_{1:T}} P(x_{1:T}, e_{1:T})$$

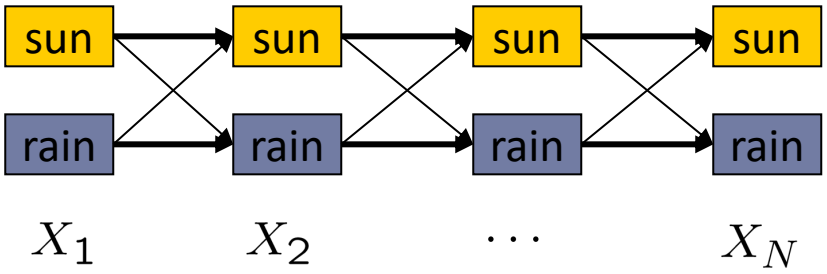
- From the sequence x, we can simply read off the words

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***: $\arg\max_{x_{1:t}} P(x_{1:t} \mid e_{1:t})$
  - speech recognition, decoding with a noisy channel

Forward / Viterbi Algorithms



Forward Algorithm (Sum)

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

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

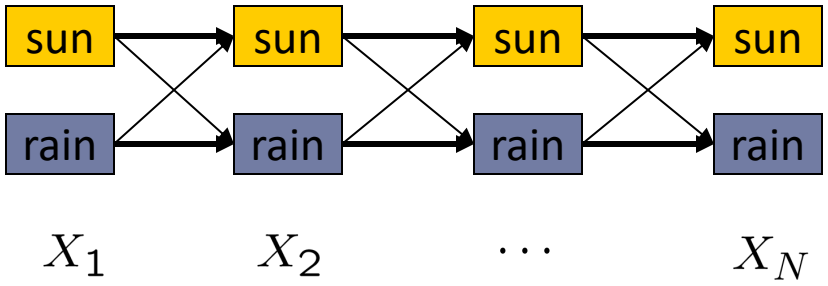
Viterbi Algorithm (Max)

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

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

State Trellis

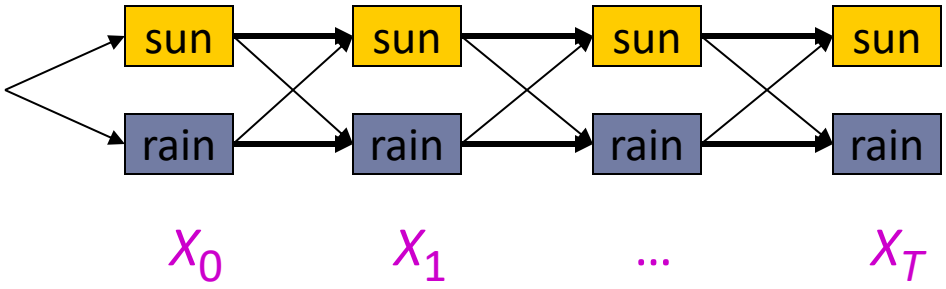
- State trellis: graph of states and transitions over time



- 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

Most Likely Explanation = Most Probable Path

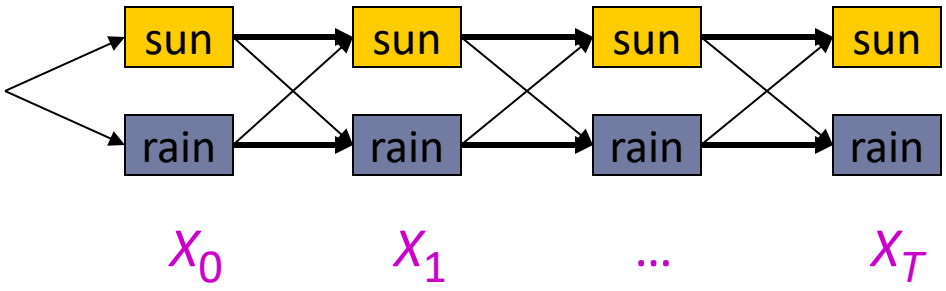
- ***State trellis***: graph of states and transitions over time



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

- 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 $P(x_0)$)
- The ***product*** of weights on a path is proportional to that state sequence's probability
- Forward algorithm computes sums of paths, ***Viterbi algorithm*** 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

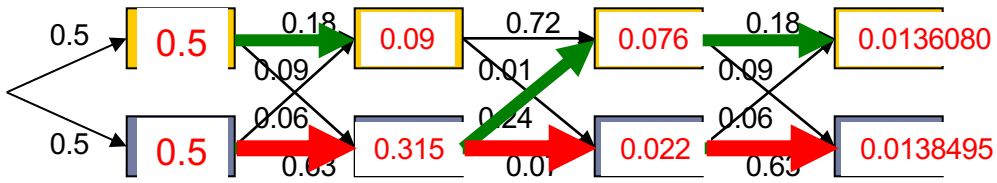
$$\boldsymbol{f}_{t+1} = \text{FORWARD}(\boldsymbol{f}_t , e_{t+1})$$
$$= \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}| x_t)\, \boldsymbol{f}_t$$

Viterbi Algorithm (max)

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

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

Viterbi Algorithm Contd.



X_0 X_1 X_2 X_T

U_1=true U_2=false U_3=true

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

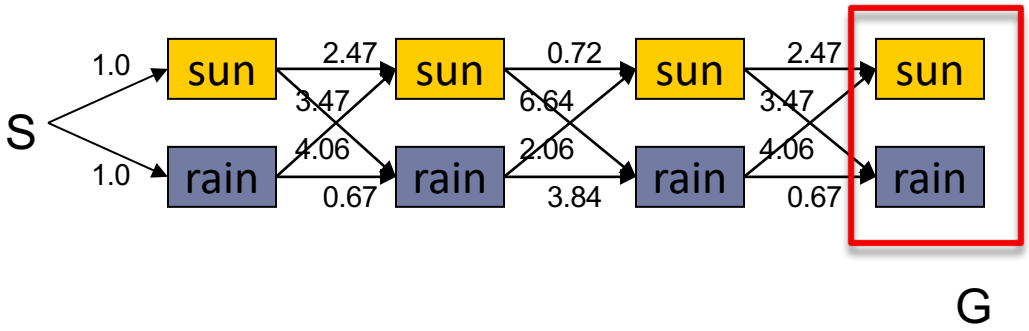
W_t	$P(U_t \mid W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

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

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

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

Viterbi in Negative Log Space



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

argmax of product of probabilities
= argmin of sum of negative log probabilities
= minimum-cost path

Viterbi is essentially breadth-first graph search

What about A*?