CS 188: Artificial Intelligence

Hidden Markov Models

Instructor: Anca Dragan --- University of California, Berkeley

[These slides were created by Dan Klein, Pieter Abbeel, and Anca. http://ai.berkeley.edu.]

Probability Recap

- Conditional probability $P(x|y) = \frac{P(x,y)}{P(y)}$
- Product rule $P(x,y) = P(x|y)P(y)$
- Chain rule $$\begin{aligned} P(X_1, X_2, \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots \\ &= \prod_{i=1}^{n} P(X_i|X_1, \ldots, X_{i-1}) \end{aligned}$$
- X, Y independent if and only if: $\forall x, y : P(x,y) = P(x)P(y)$
- X and Y are conditionally independent given Z if and only if:

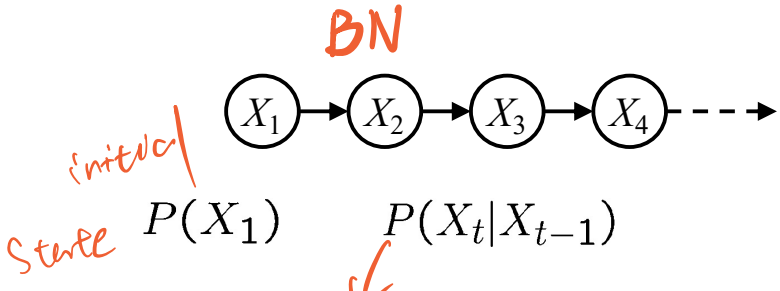
$$\forall x, y, z : P(x,y|z) = P(x|z)P(y|z)$$

Reasoning over Time or Space

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

Markov Models

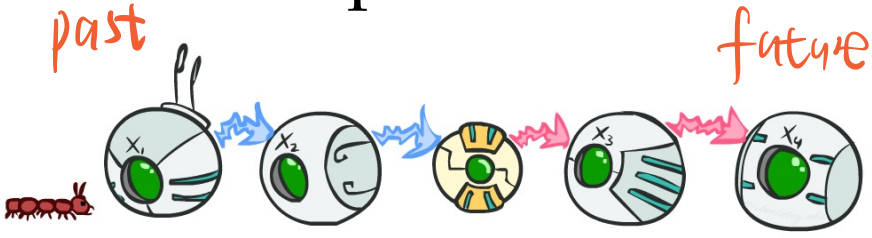
- Value of X at a given time is called the state



$$P(X_t) = ?$$

- Parameters: called transition probabilities or dynamics, specify how the state evolves over time (also, initial state probabilities)
- Stationarity assumption: transition probabilities the same at all times
- Same as MDP transition model, but no choice of action
- A (growable) BN: We can always use generic BN reasoning on it if we truncate the chain at a fixed length

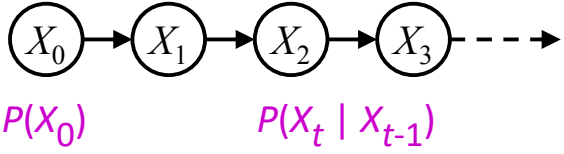
Markov Assumption: Conditional Independence



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

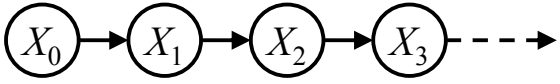
Markov Models (aka Markov chain/process)

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



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

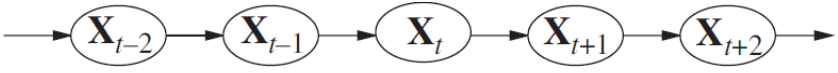
Quiz: are Markov models a special case of Bayes nets?



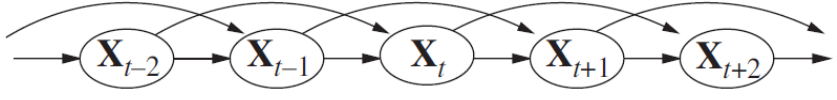
- Yes and no!
- Yes:
  - Directed acyclic graph, joint = product of conditionals
- No:
  - Infinitely many variables (unless we truncate)
  - Repetition of transition model not part of standard Bayes net syntax

截断

Markov Assumption: Conditional Independence

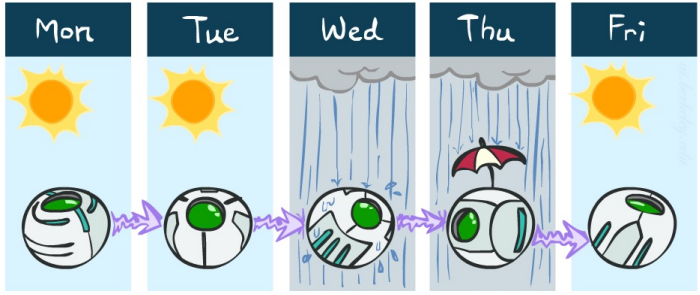


- ***Markov*** assumption: X_{t+1}, … are independent of X_0,…, X_{t-1} given X_t
  - Past and future independent given the present
  - Each time step only depends on the previous
- This is a ***first-order*** Markov model
- A kth-order model allows dependencies on k earlier steps



Example Markov Chain: Weather

- States: X = {rain, sun}

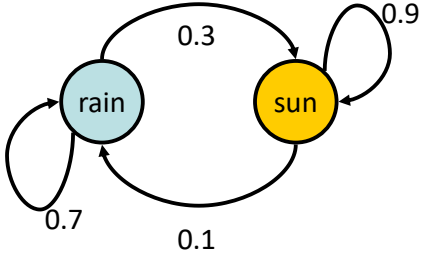


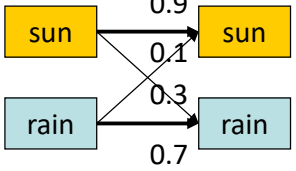
- Initial distribution: 1.0 sun

- 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

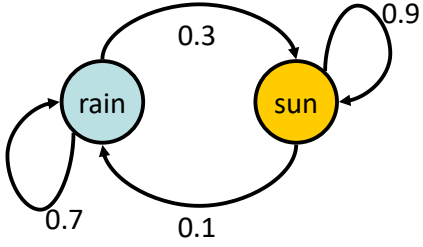




+1 step

Example Markov Chain: Weather

- Initial distribution: 1.0 sun



- What is the probability distribution after one step?

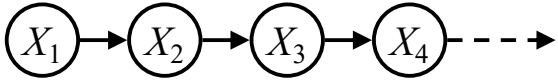
$$P(X_2 = sun) = \sum_{x_1} P(x_1, X_2 = sun) = \sum_{x_1} P(X_2 = sun|x_1)P(x_1)$$

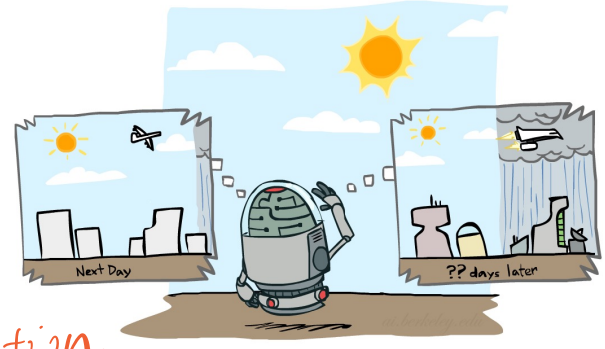
$$P(X_2 = \text{sun}) = P(X_2 = \text{sun}|X_1 = \text{sun})P(X_1 = \text{sun}) + P(X_2 = \text{sun}|X_1 = \text{rain})P(X_1 = \text{rain})$$

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

Mini-Forward Algorithm

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





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

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

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

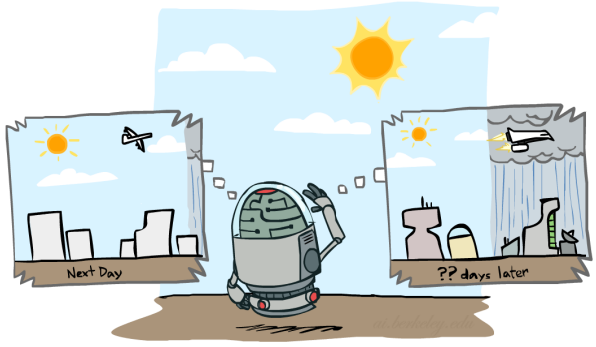
update equation.

Forward simulation

Weather prediction

- 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



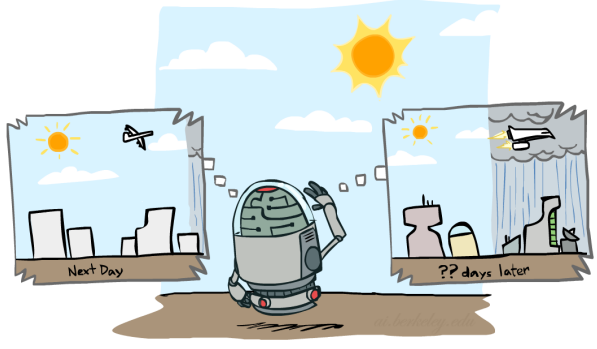
- What is the weather like at time 1?
  - $P(X_1) = \sum_{x_0} P(X_1, X_0=x_0)$
  - $= \sum_{x_0} P(X_0=x_0)\, P(X_1 \mid X_0=x_0)$
  - $= 0.5\langle 0.9,0.1\rangle + 0.5\langle 0.3,0.7\rangle = \langle 0.6,0.4\rangle$

T, F ...

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

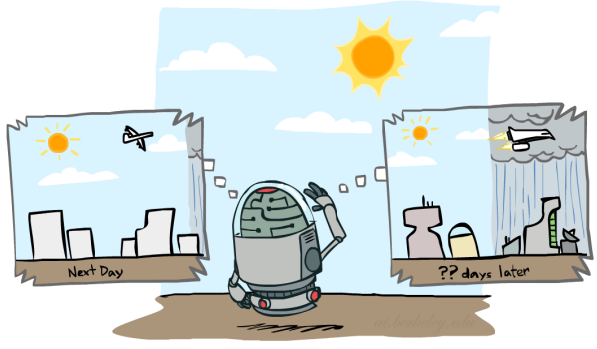


- What is the weather like at time 2?
  - $P(X_2) = \sum_{x_1} P(X_2, X_1=x_1)$
  - $= \sum_{x_1} P(X_1=x_1)\, P(X_2 \mid X_1=x_1)$
  - $= 0.6\langle 0.9,0.1\rangle + 0.4\langle 0.3,0.7\rangle = \langle 0.66,0.34\rangle$

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

Forward algorithm (simple form)

- What is the state at time t (given an initial distribution $P(X_0)$)?
  - $P(X_t) = \sum_{x_{t-1}} P(X_t, X_{t-1}=x_{t-1})$
  - $= \sum_{x_{t-1}} P(X_{t-1}=x_{t-1})\, P(X_t \mid X_{t-1}=x_{t-1})$

Probability from previous iteration

Transition model

- Iterate this update starting at $t=0$

Example Run of Mini-Forward Algorithm

- From initial observation of sun

sunny start

$$\begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} \quad \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} \quad \begin{bmatrix} 0.84 \\ 0.16 \end{bmatrix} \quad \begin{bmatrix} 0.804 \\ 0.196 \end{bmatrix} \rightarrow \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

$$P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_4) \quad P(X_\infty)$$

- From initial observation of rain

rainy start

$$\begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix} \quad \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \quad \begin{bmatrix} 0.48 \\ 0.52 \end{bmatrix} \quad \begin{bmatrix} 0.588 \\ 0.412 \end{bmatrix} \rightarrow \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

$$P(X_1) \quad P(X_2) \quad P(X_3) \quad P(X_4) \quad P(X_\infty)$$

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

"Decay"

$$\begin{bmatrix} p \\ 1-p \end{bmatrix} \quad \ldots \rightarrow \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

$$P(X_1) \quad P(X_\infty)$$

[Demo: L13D1,2,3]

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

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

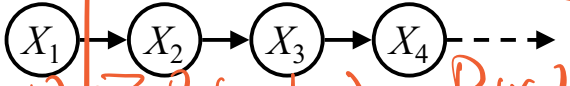
$P(x') = \sum_x P(x'|x) \cdot P(x)$

$\sum P(x') = 1$

Solve it to find ∞

Example: Stationary Distributions

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



$$P(sun) \quad \Sigma P(sun|x) \cdot P(x)$$

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

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

$$P(rain) \quad \Sigma P(rain|x) \cdot P(x)$$

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

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

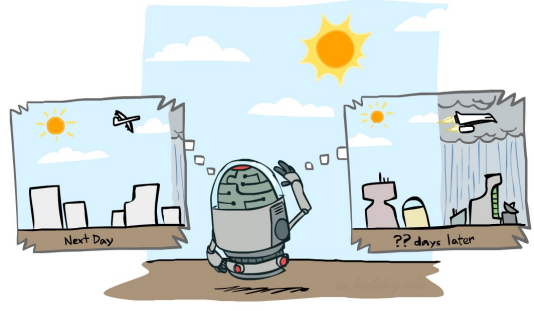
$$P_\infty(sun) = 3P_\infty(rain)$$

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

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

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

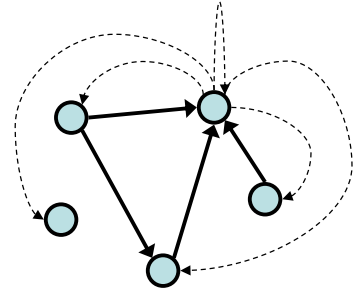
$$P_\infty(rain) = 1/4$$



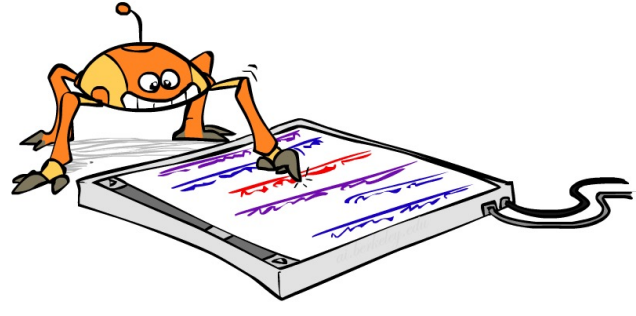
X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7

Application of Stationary Distribution: Web Link Analysis

- PageRank over a web graph
  - Each web page is a possible value of 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)

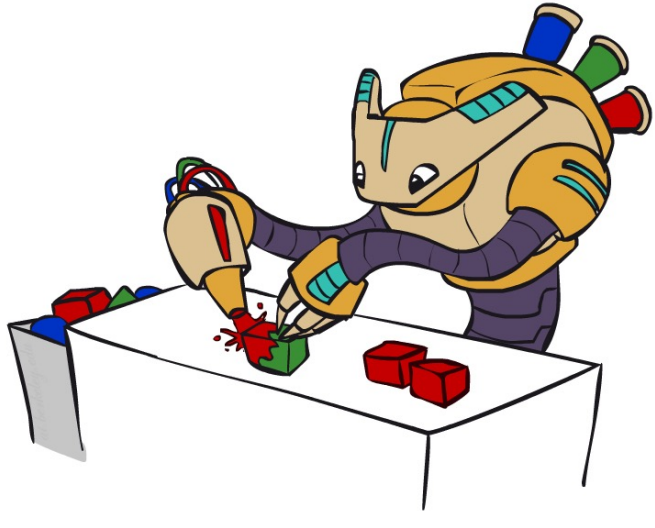


Application of Stationary Distributions: Gibbs Sampling*

- 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!

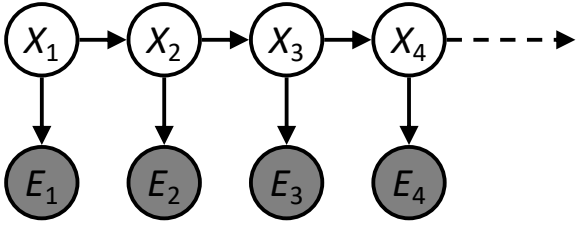


Hidden Markov Models

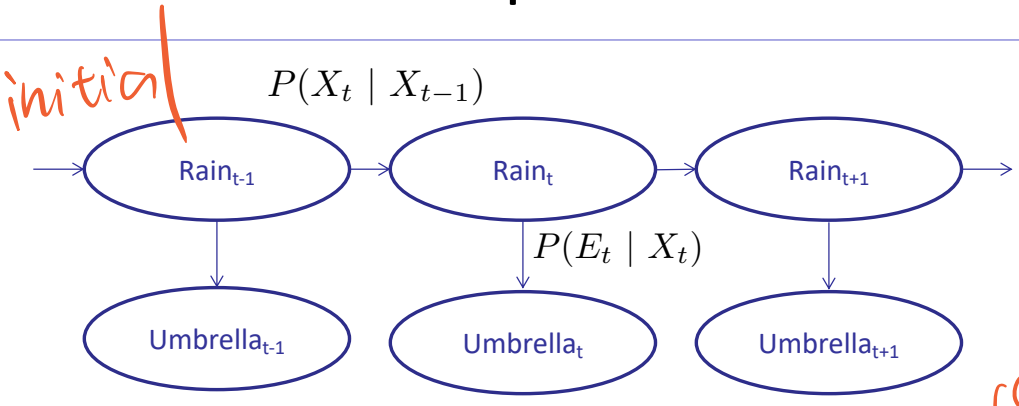
HMM

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 outputs (effects) at each time step



Example: Weather HMM



(state)

Markov

Emit part

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

R_{t-1}	R_t	$P(R_t \mid R_{t-1})$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

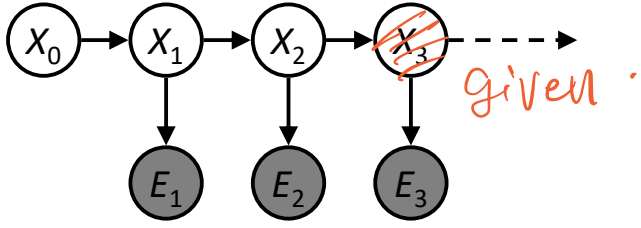
R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

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:

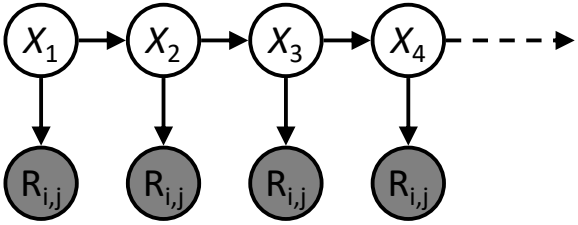
$$P(X_0, X_1, E_1, \ldots, X_T, E_T) = P(X_0) \prod_{t=1:T} P(X_t \mid X_{t-1}) P(E_t \mid X_t)$$

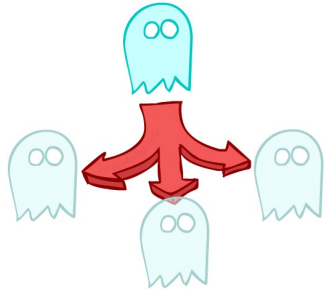
- Independence in HMM
  - Future states are independent of the past given the present
  - Current evidence is independent of everything else given the current state



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

$P(X_1)$

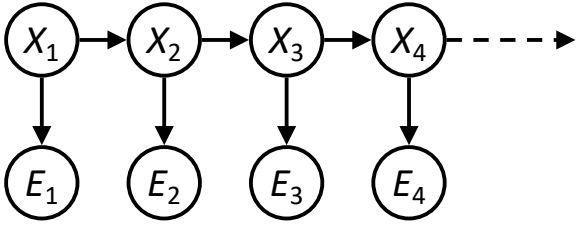
1/6	1/6	1/2
0	1/6	0
0	0	0

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

[Demo: Ghostbusters – Circular Dynamics – HMM (L14D2)]

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



E_1 inde of E_4 given X_2.

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

Real HMM Examples

- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)
- 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

Inference tasks

- Useful notation: $X_{a:b} = X_a, X_{a+1}, \ldots, X_b$
- ***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***: $\arg\max_{x_{0:t}} P(x_{0:t} \mid e_{1:t})$
  - Ex: speech recognition, decoding with a noisy channel

Filtering / Monitoring

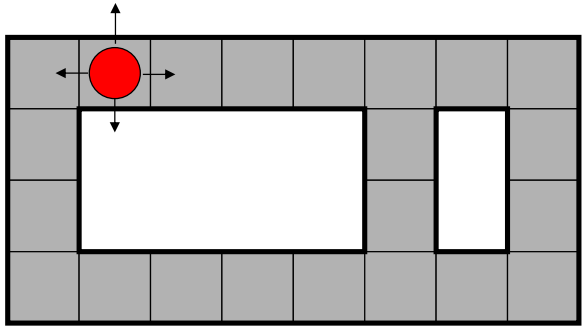
- Filtering, or monitoring, is the task of tracking the distribution $B_t(X) = P_t(X_t \mid e_1, \ldots, e_t)$ (the belief state) over time

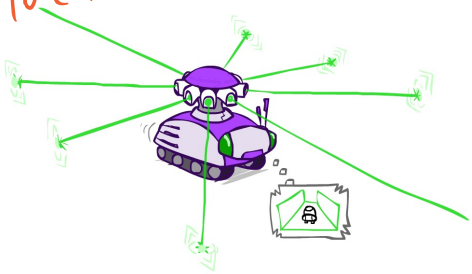
evidence

- We start with $B_1(X)$ in an initial setting, usually uniform
- As time passes, or we get observations, we update B(X)
- The Kalman filter was invented in the 60's and first implemented as a method of trajectory estimation for the Apollo program

Example: Robot Localization

Example from Michael Pfeiffer





Hidden state: robot location

Prob 0 1

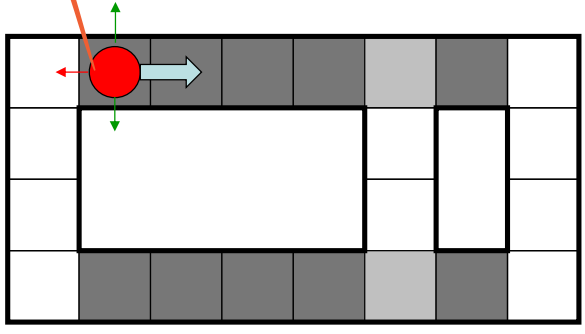
t=0

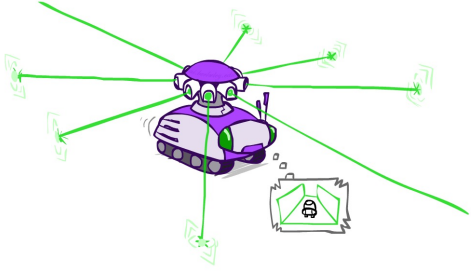
4 bits wall / no wall for 4 direction

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

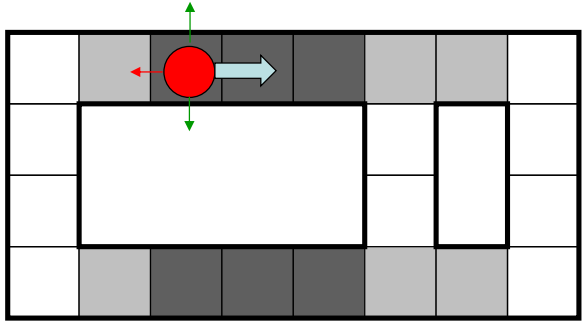


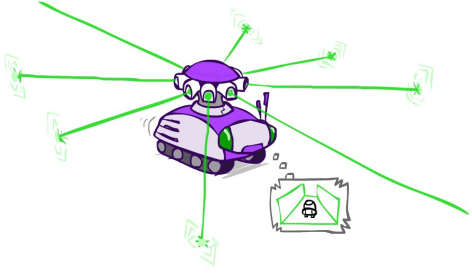


t=1

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

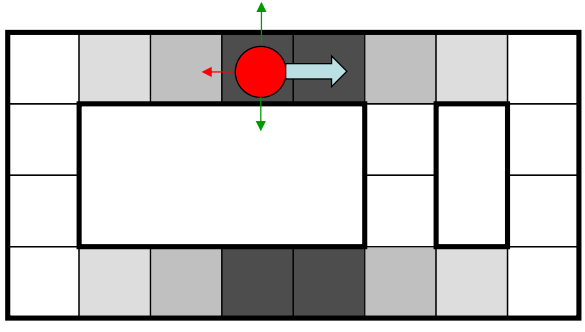
Example: Robot Localization

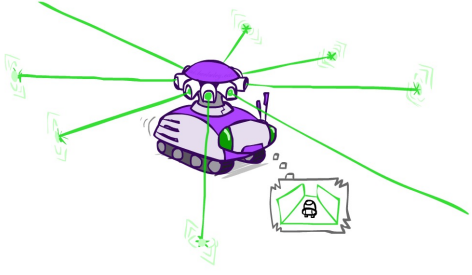




t=2

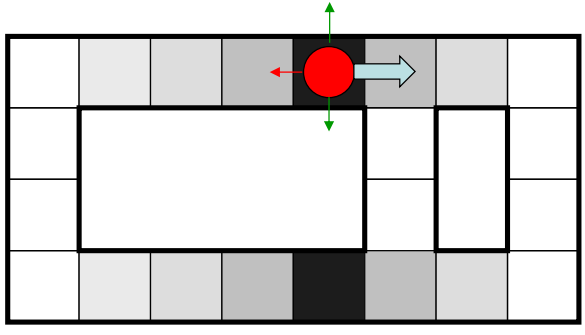
Example: Robot Localization

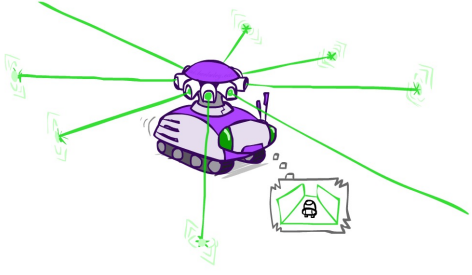




t=3

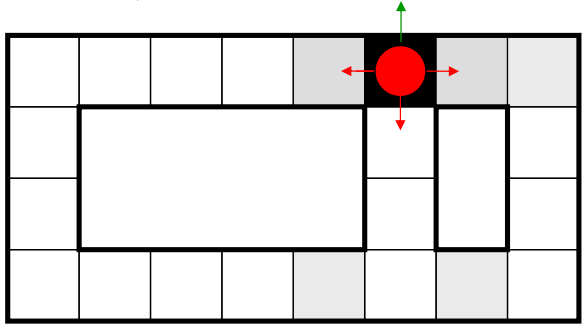
Example: Robot Localization

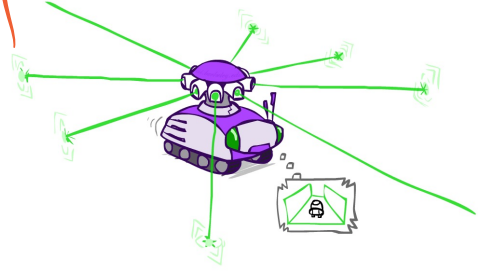




t=4

Example: Robot Localization





Prob 0 1

t=5

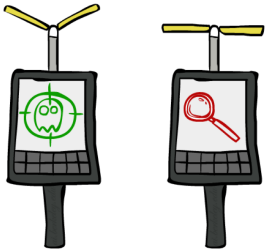
Inference: Find State Given Evidence

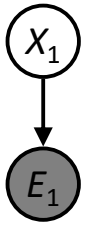
- We are given evidence at each time and want to know

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

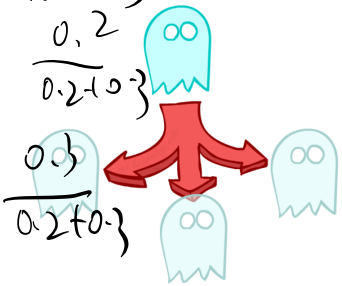
- Idea: start with $P(X_1)$ and derive B_t in terms of B_{t-1}
  - equivalently, derive B_{t+1} in terms of B_t

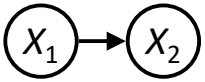
Inference: Base Cases





X_1	$\propto P(X_1 \mid e_1)$
0	0.2
1	0.3





$$P(X_1|e_1)$$

$$P(x_1|e_1) = P(x_1, e_1)/P(e_1)$$
$$\propto_{X_1} P(x_1, e_1) = P(e_1, x_1)$$
$$= P(x_1)P(e_1|x_1)$$

mean: when consider only X as variable propotion.

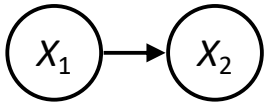
$$P(X_2)$$

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

Passage of Time

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

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



- Then, after one time step passes:

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

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

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

- Or compactly:

$$B'(X_{t+1}) = \sum_{x_t} P(X'|x_t) B(x_t)$$

- Basic idea: beliefs get "pushed" through the transitions
  - With the "B" notation, we have to be careful about what time step t the belief is about, and what evidence it includes

Example: Passage of Time

- As time passes, uncertainty "accumulates"

(Transition model: ghosts usually go clockwise)

<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	1.00	<0.01	<0.01	<0.01
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

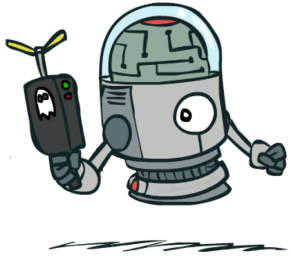
T = 1

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

T = 2

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

T = 5



Observation

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

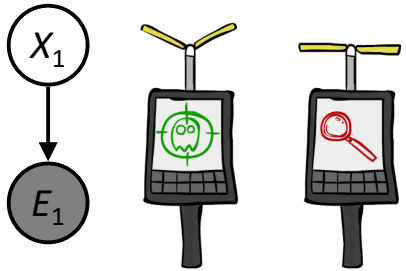
$$B'(X_{t+1}) = P(X_{t+1}|e_{1:t})$$

- Then, after evidence comes in:

$$\begin{aligned} P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t}) \\ &\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|e_{1:t}, X_{t+1})P(X_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \end{aligned}$$

- Or, compactly:

$$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1}|X_{t+1})B'(X_{t+1})$$

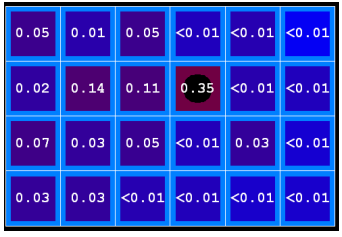


- Basic idea: beliefs "reweighted" by likelihood of evidence
- Unlike passage of time, we have to renormalize

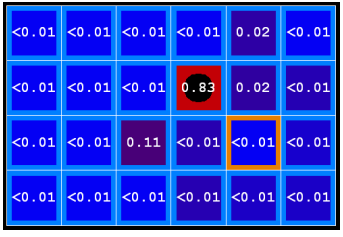
(new) Example: Observation

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

More known
More accurate

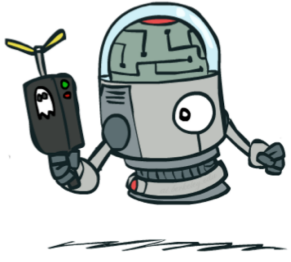


Before observation

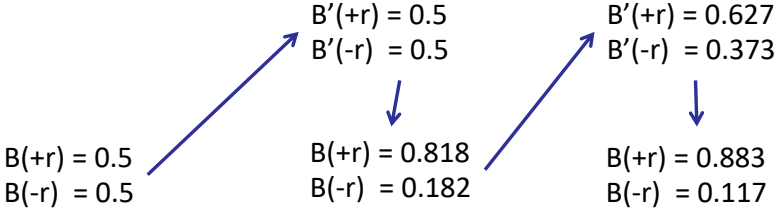


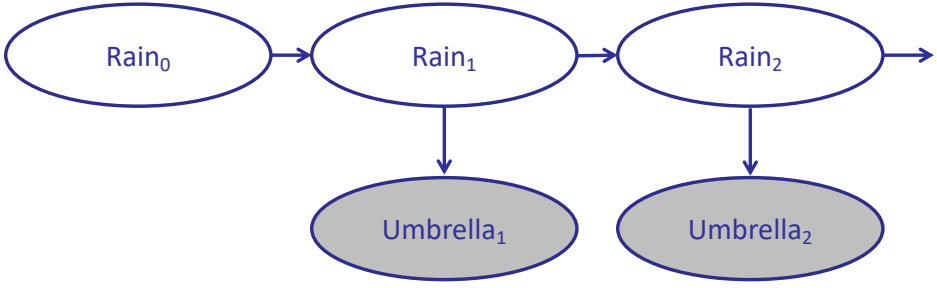
After observation

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



Example: Weather HMM





R_t	R_{t+1}	$P(R_{t+1} \mid R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

The Forward Algorithm

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

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

- We can derive the following updates

$$P(x_t|e_{1:t}) \propto_X P(x_t, e_{1:t})$$

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

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

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

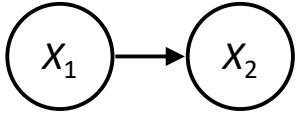
We can normalize as we go if we want to have P(x|e) at each time step, or just once at the end…

Online Belief Updates

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

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

trans.

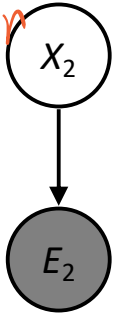


- We update for evidence:

$$P(x_t|e_{1:t}) \propto_X P(x_t|e_{1:t-1}) \cdot P(e_t|x_t)$$

Can do time update again and again until find evidence

need to normal



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

Filtering algorithm

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

Apply Bayes' rule

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

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

Bayes' Rule

- Two ways to factor a joint distribution over two variables:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

- Dividing, we get:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

- Why is this at all helpful?
  - Lets us build one conditional from its reverse
  - Often one conditional is tricky but the other one is simple
  - Foundation of many systems we'll see later

- In the running for most important AI equation!

Filtering algorithm

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

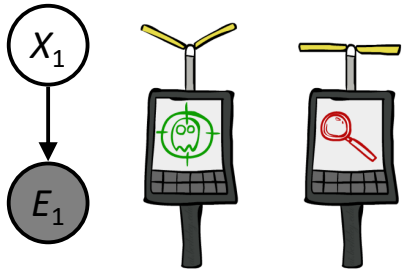
- $P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{1:t}, e_{t+1})$
- $= \alpha\, P(e_{t+1} | X_{t+1}, e_{1:t})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1})\, P(X_{t+1} | e_{1:t})$
- $= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} P(x_t | e_{1:t})\, P(X_{t+1} | x_t, e_{1:t})$

Condition on X_t

Observation

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

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



- Then, after evidence comes in:

$$\begin{aligned} P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}, e_{t+1}|e_{1:t})/P(e_{t+1}|e_{1:t}) \\ &\propto_{X_{t+1}} P(X_{t+1}, e_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|e_{1:t}, X_{t+1})P(X_{t+1}|e_{1:t}) \\ &= P(e_{t+1}|X_{t+1})P(X_{t+1}|e_{1:t}) \end{aligned}$$

- Or, compactly:

$$B(X_{t+1}) \propto_{X_{t+1}} P(e_{t+1}|X_{t+1})B'(X_{t+1})$$

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

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

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

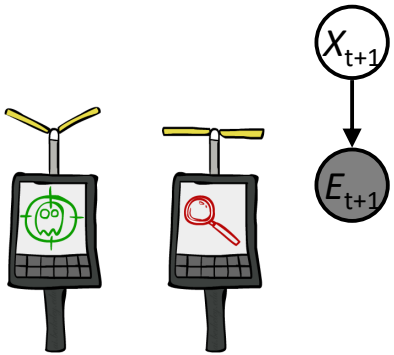
Normalize

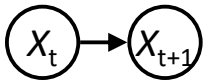
Update

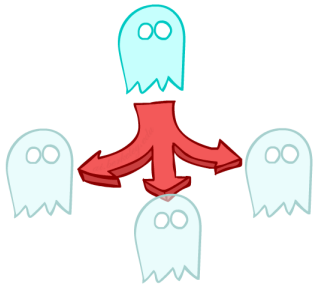
Predict

evidence

(past)







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

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

known

chain law — observe new evidence

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

independent marker

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

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

Inde

Filtering algorithm

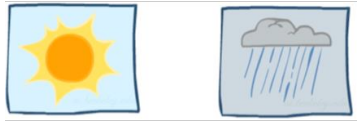
- $P(X_{t+1}|e_{1:t+1}) = \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(x_t \mid e_{1:t})\, P(X_{t+1}|\, x_t)$

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

上一轮的滤波结果

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

Example: Weather HMM



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

Normalize | Update | Predict

predict

0.6
0.4

$$P(s) = 0.5 \times 0.9 + 0.5 \times 0.3 = 0.6$$
$$P(r) = 0.5 \times 0.1 + 0.5 \times 0.7 = 0.4$$

update & normalize

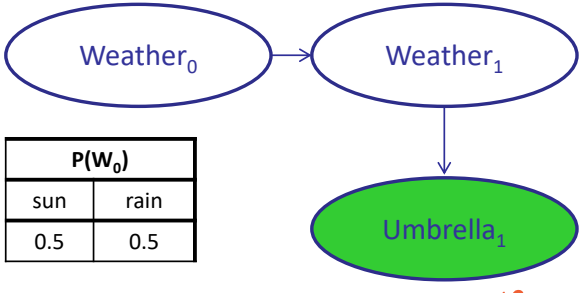
f(sun) = 0.5
f(rain) = 0.5

f(sun) = 0.25
f(rain) = 0.75

$$P(s|u) \propto 0.6 \times 0.2 = 0.12$$
$$P(s|u) \propto 0.4 \times 0.9 = 0.36$$

normalize

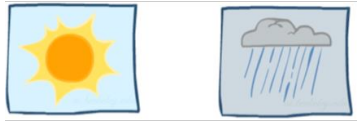
W_{t-1}	$P(W_t \mid W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7



$P(W_0)$	
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: Weather HMM



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

Normalize | Update | Predict

predict

0.6
0.4

f(sun) = 0.5
f(rain) = 0.5

update & normalize

f(sun) = 0.25
f(rain) = 0.75

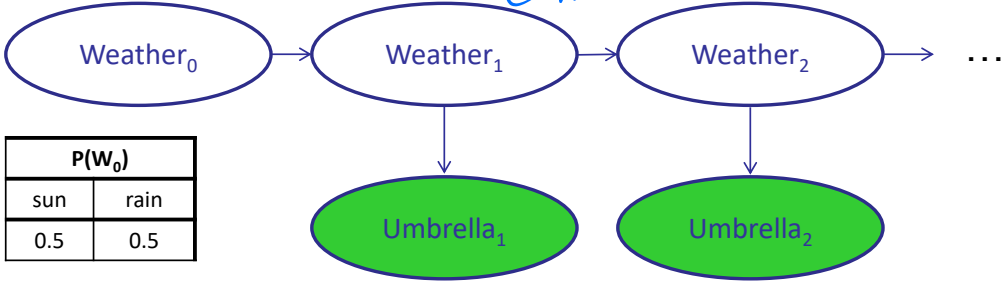
predict

0.45
0.55

update & normalize

f(sun) = 0.154
f(rain) = 0.846

e involved



$P(W_0)$	
sun	rain
0.5	0.5

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

Filtering algorithm

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

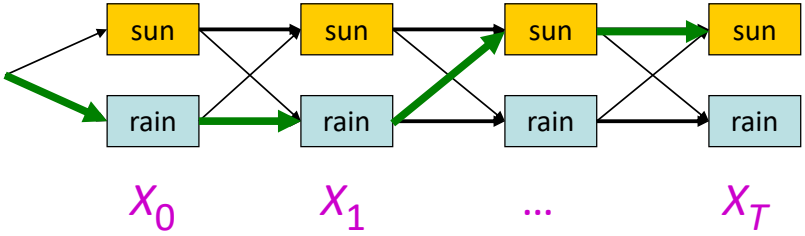
α is a constant. So if we only want to compute $P(x_t \mid e_{1:t})$, then we can skip normalization when computing $P(x_1 \mid e_1)$, $P(x_2 \mid e_{1:2})$, …, $P(x_{t-1} \mid e_{1:t-1})$

find normalization

Can just do

Another view of the algorithm

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



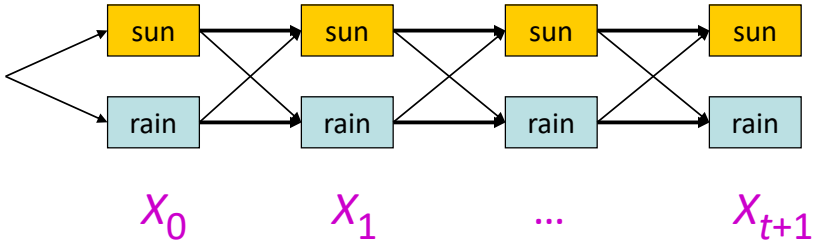
- Each arc represents some transition $x_{t-1} \rightarrow x_t$
- Each arc has weight $P(x_t \mid x_{t-1})\, P(e_t \mid x_t)$ (arcs to initial states have weight $P(x_0)$)
- Each path is a sequence of states
- The ***product*** of weights on a path is proportional to that state sequence's probability

One path: 和全称

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

path filtering

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

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

    $= \alpha\, P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t)\, \boldsymbol{f}_{1:t}[x_t]$

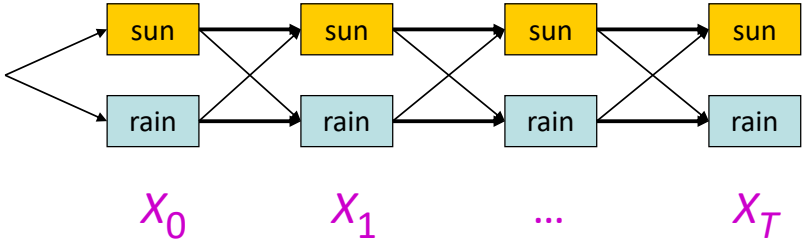
Most Likely Explanation

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***: $\arg\max_{x_{0:t}} P(x_{0:t} \mid e_{1:t})$
  - 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

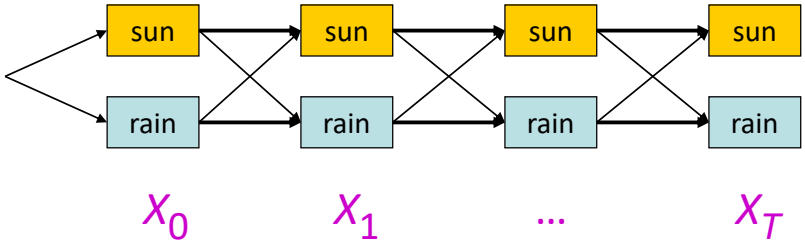
  $P(x_0) \prod_t P(x_t \mid x_{t-1}) P(e_t \mid x_t) = P(x_{0:t}, e_{1:t}) \propto P(x_{0:t} \mid e_{1:t})$

- ***Viterbi algorithm*** computes best paths

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

finding most possible path

Forward / Viterbi algorithms



Viterbi Algorithm (max)

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

$\boldsymbol{m}_{1:t+1} = \text{VITERBI}(\boldsymbol{m}_{1:t}, e_{t+1})$

$= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}|x_t)\, \boldsymbol{m}_{1:t}[x_t]$

Forward Algorithm (sum)

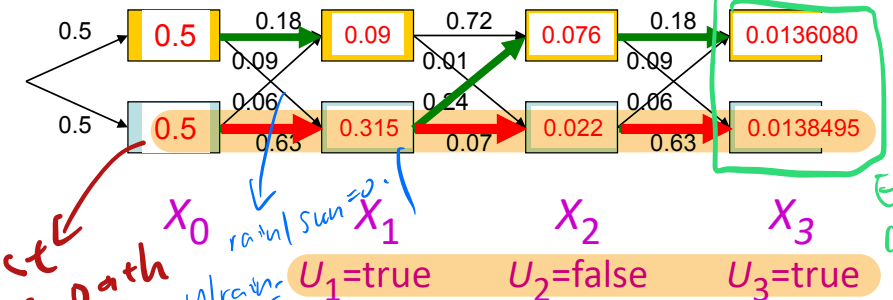
For each state at time t, keep track of the ***total probability of all paths*** to it

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

$= \alpha\, P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t)\, \boldsymbol{f}_{1:t}[x_t]$

Viterbi algorithm contd.

$P(W_0)$	
sun	rain
0.5	0.5



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

- $m_{1:t+1} = P(e_{t+1} \mid X_{t+1}) \max_{x_t} P(X_{t+1} \mid x_t)\, m_{1:t}[x_t]$

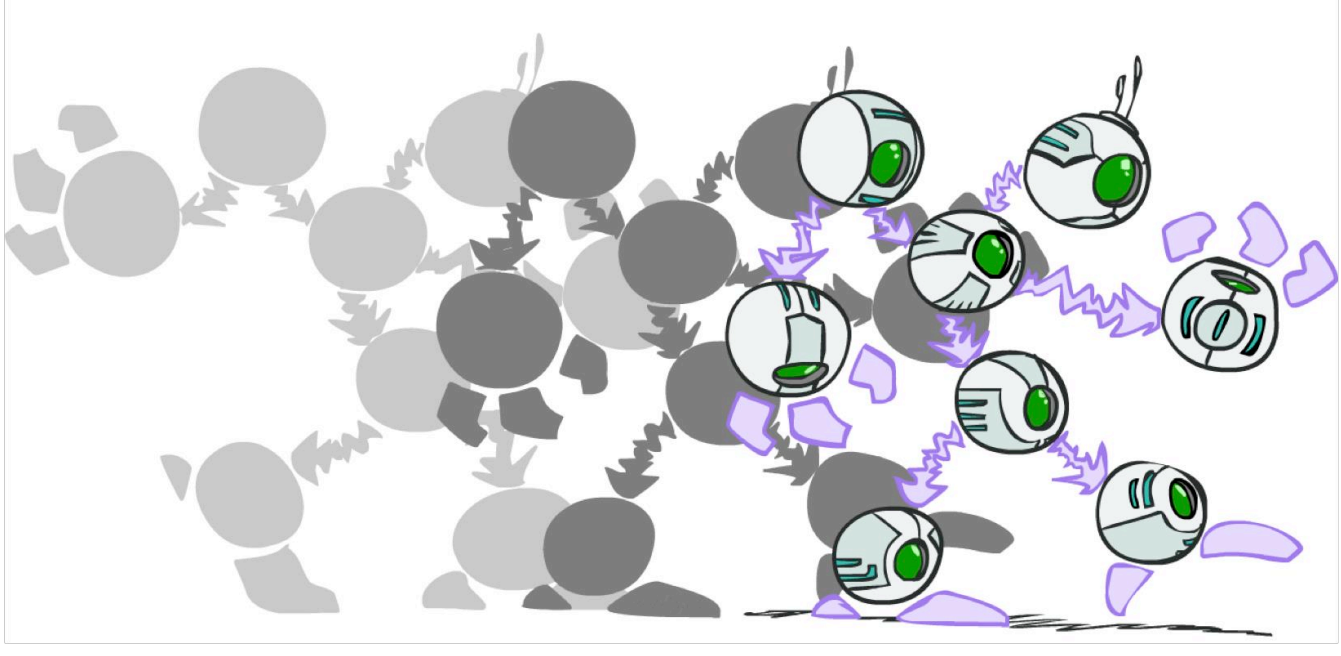
- Time complexity: $O(|X|^2 T)$
- Space complexity: $O(|X| T)$

eg: find most possible path to both sun / rain.

DBN

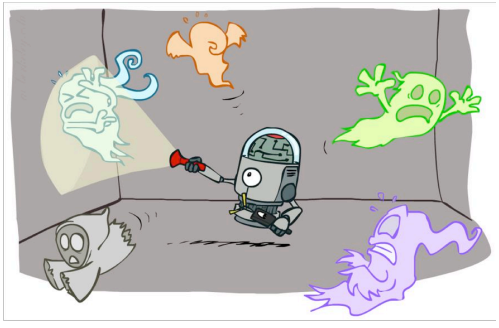
Dynamic Bayes Nets

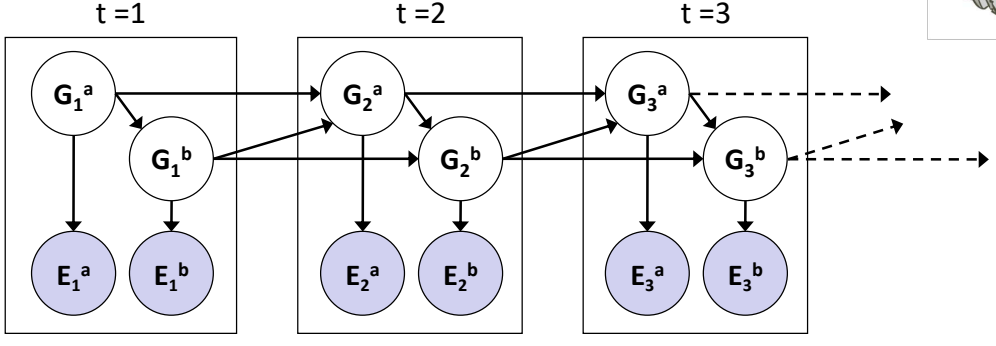
(intro)



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*



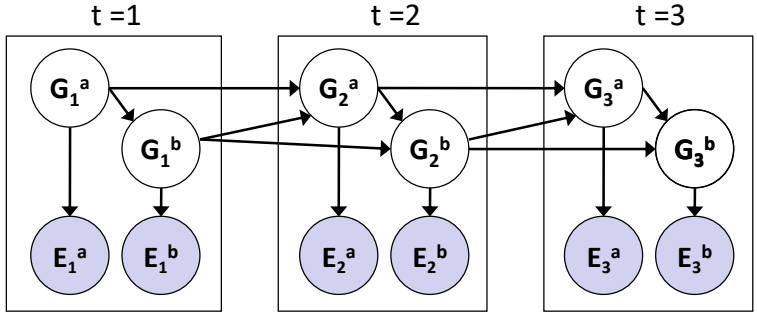


DBNs and HMMs

- Every HMM is a DBN
- Every discrete DBN can be represented by a HMM
  - Each HMM state is Cartesian product of DBN state variables
    - E.g., 3 binary state variables => one state variable with 2^3 possible values
  - Advantage of DBN vs. HMM?
    - Sparse dependencies => exponentially fewer parameters
    - E.g., 20 binary state variables, 2 parents each;
      DBN has 20 x 2^{2+1} = 160 parameters, HMM has 2^{20} x 2^{20} =~ 10^{12} parameters

Exact Inference in DBNs

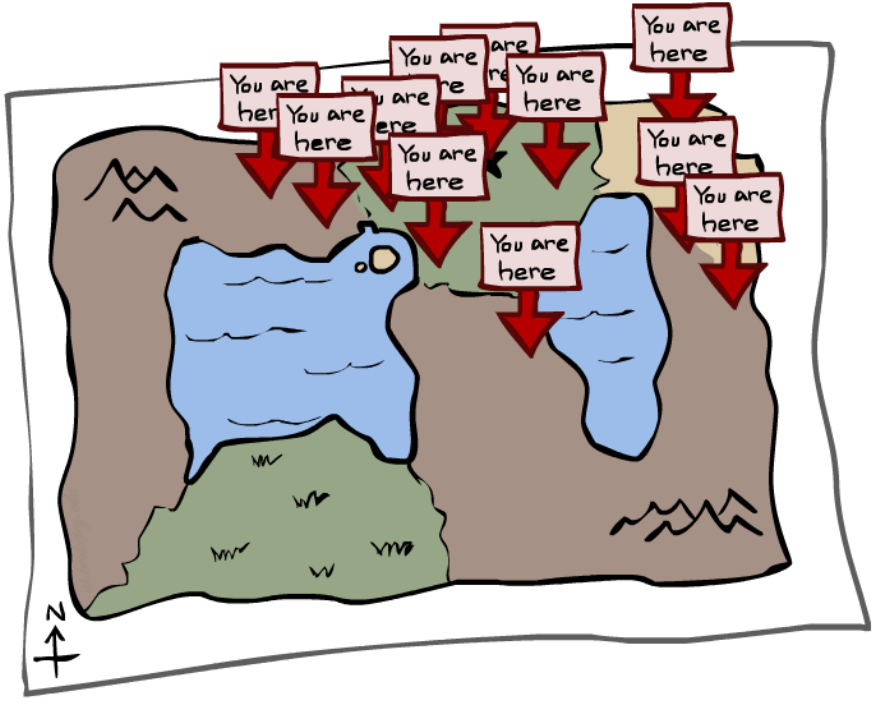
- Variable elimination applies to dynamic Bayes nets
- Offline: "unroll" the network for T time steps, then eliminate variables to find $P(X_T | e_{1:T})$
  - Problem: results in very large BN



- Can we do better?
  - Do we need to unroll for many steps? What is the best variable order of elimination?
- Online: unroll as we go, eliminate all variables from the previous time step

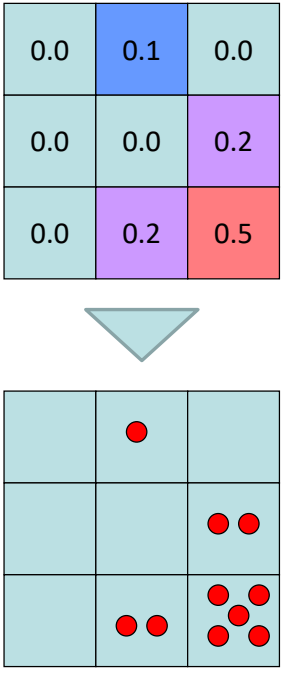
each time

Particle Filtering



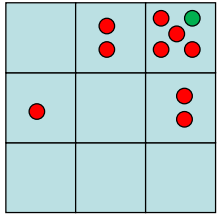
Particle Filtering

- Filtering: approximate solution
- Sometimes |X| is too big to use exact inference
- Solution: approximate inference
  - Track samples of X, not all values
  - 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 << |X|
    - More particles, more accuracy; but a large N would defeat the point.



Representation: Particles

- At first, all particles have a weight of 1

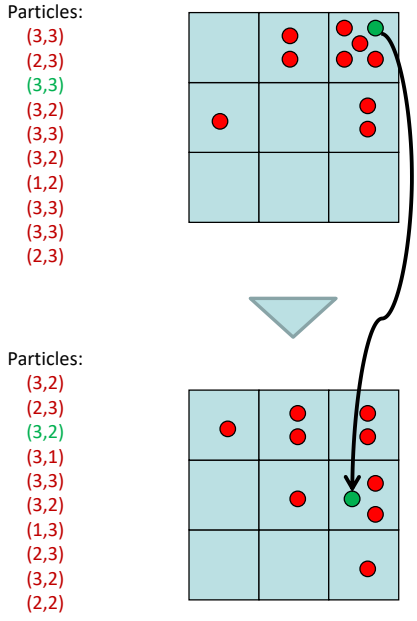


Particles:
(3,3)
(2,3)
(3,3)
(3,2)
(3,3)
(3,2)
(1,2)
(3,3)
(3,3)
(2,3)

Particle Filtering: Propagate forward

- Each particle is moved by sampling its next position from the transition model:
  - $x_{t+1} \sim P(X_{t+1} \mid x_t)$

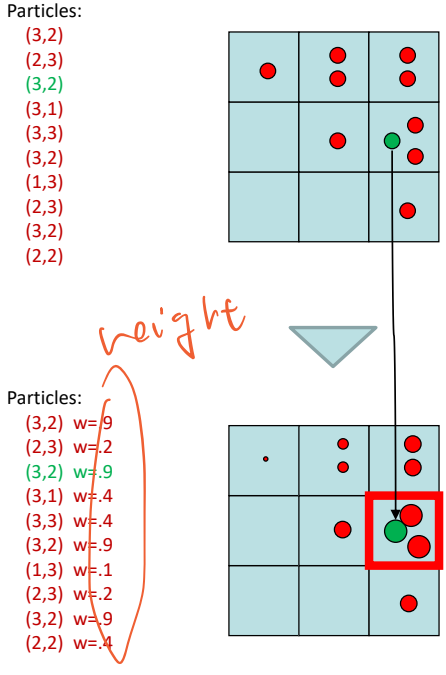
- This captures the passage of time
  - If enough samples, close to exact probabilities (consistent)



Particle Filtering: Observe

- Similar to likelihood weighting, weight samples based on the evidence
  - $W = P(e_t \mid x_t)$
  - 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…

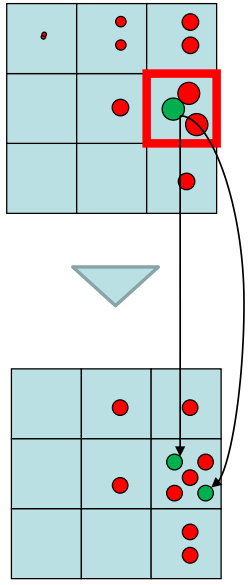


Particle Filtering: Resample

- 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

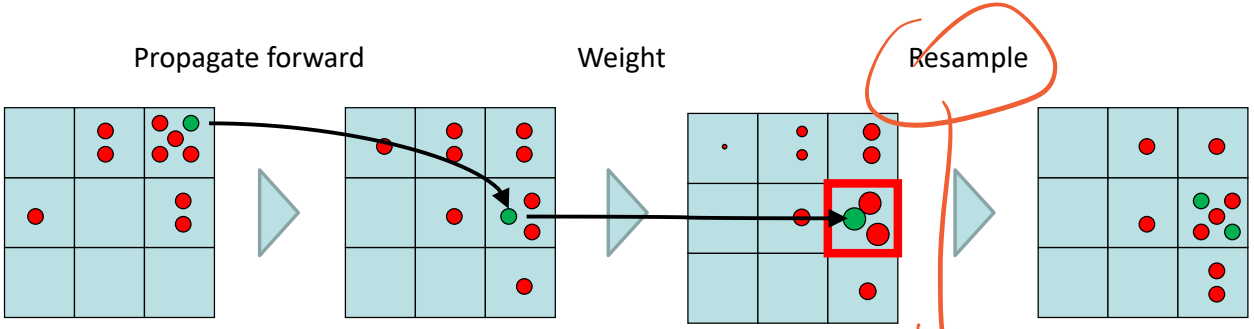
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

(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

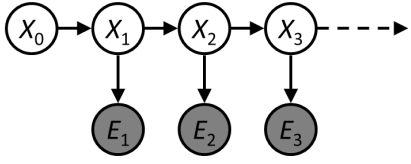
- Particles: track samples of states rather than an explicit distribution

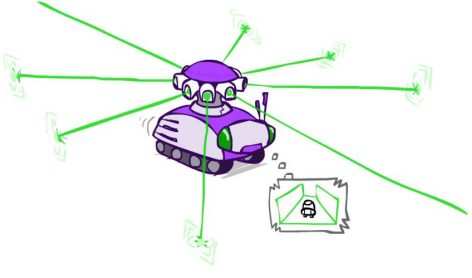


Consistency: see proof in AIMA Ch. 15

Summary

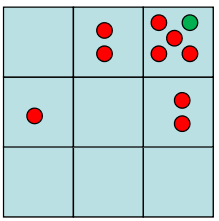
- Probabilistic temporal models
  - Markov model
  - Hidden Markov model
    - Filtering: forward algorithm
    - MLE: Viterbi algorithm
  - Dynamic Bayesian network
  - Approximate inference by particle filtering





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)

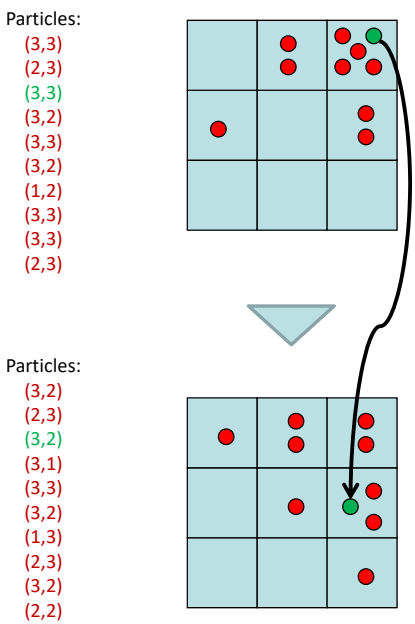
Particle Filtering: Elapse Time

- 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
  - 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)



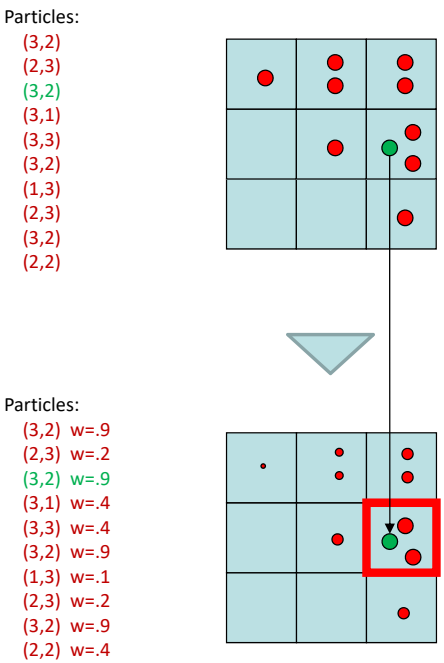
Particle Filtering: Observe

- 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))

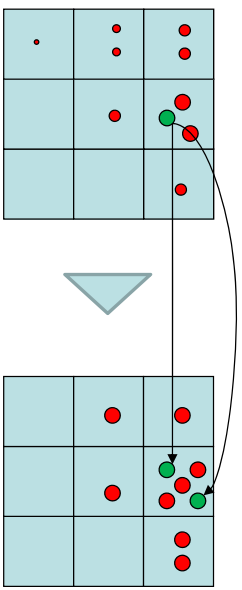


Particle Filtering: Resample

- 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

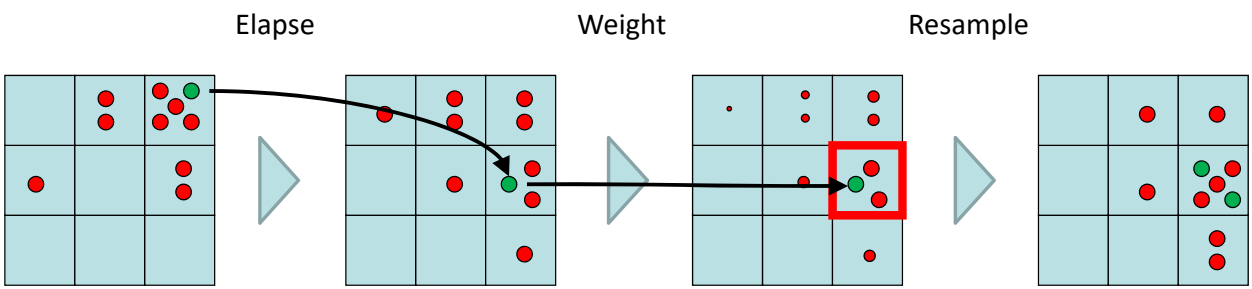
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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



Recap: Particle Filtering

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



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)

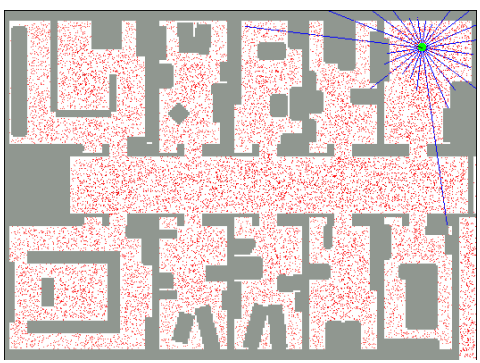
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

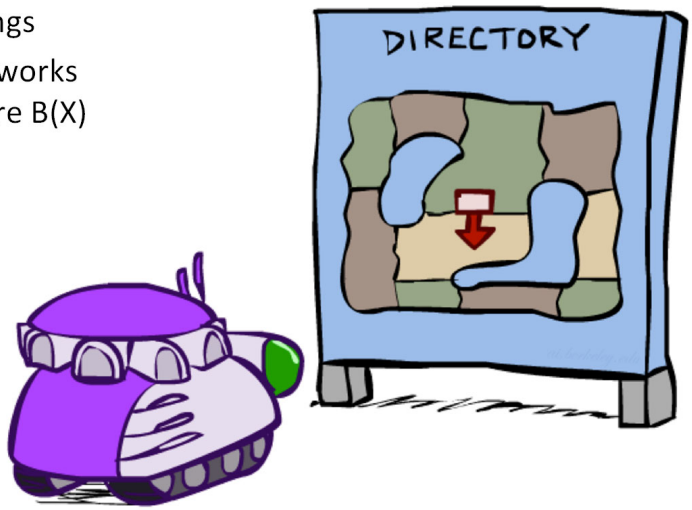
(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)

[Demos: ghostbusters particle filtering (L15D3,4,5)]

Robot Localization

- In robot localization:
  - We know the map, but not the robot's position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store B(X)
  - Particle filtering is a main technique



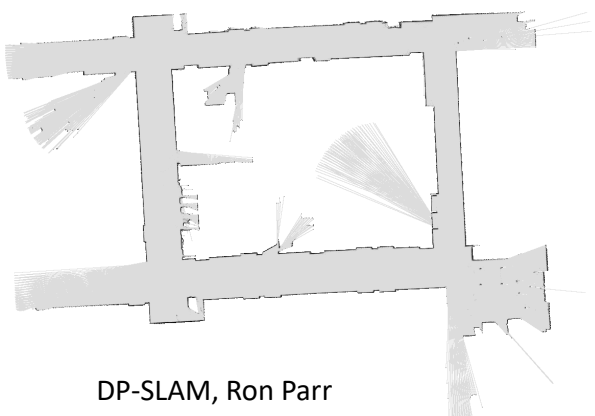


Particle Filter Localization (Sonar)

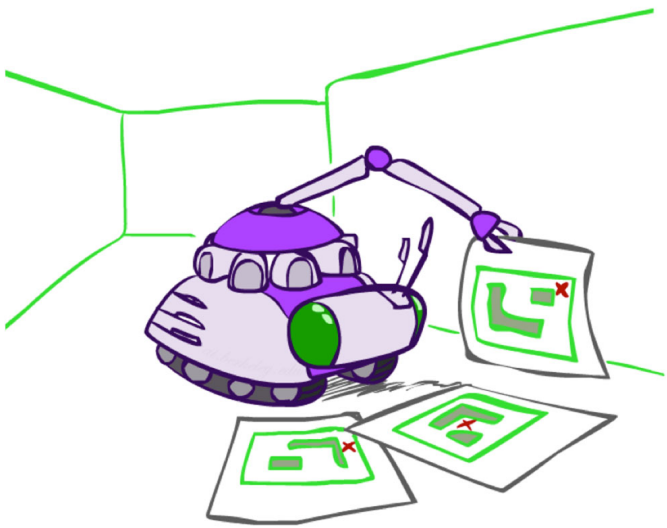
[Video: global-sonar-uw-annotated.avi]

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



[Demo: PARTICLES-SLAM-mapping1-new.avi]