CS 343: Artificial Intelligence

Hidden Markov Models, Particle Filtering, and VPI

Profs. Peter Stone and Yuke Zhu — The University of Texas at Austin

[These slides based on those of Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Good morning colleagues!

- Past due:
  - HW1-5: Search, CSPs, Games, MDP, RL
  - 7 reading responses: AI100 report; 8 Textbook readings
  - P0,1,2: tutorial, Search, Multiagent
  - Midterm
- Upcoming EdX Homeworks
  - HW6: Bayes Nets – due Monday 4/5 at 11:59 pm
  - HW7: Sampling, HMMs, Particle Filters, and VPI – due Monday 4/12 at 11:59 pm
- Upcoming programming projects
  - P3: RL – due Wednesday 3/31 at 11:59pm
  - **P4: Bayes Nets – due Wednesday 4/14 at 11:59pm**
  - P5: Particle Filters – due Wednesday 4/21 at 11:59pm
- Readings: Naive Bayes and Perceptrons – Due Monday 4/5 at 9:30am
- Contest: Capture the flag
  - Qualification due 4/28 (required); Finals 5/3 (extra credit)

Real HMM Examples

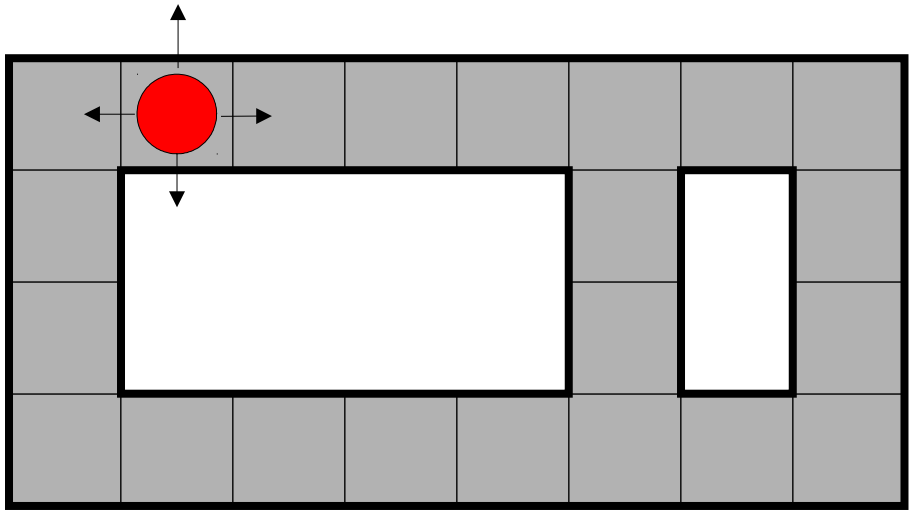
- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)
- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options
- Robot tracking:
  - Observations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

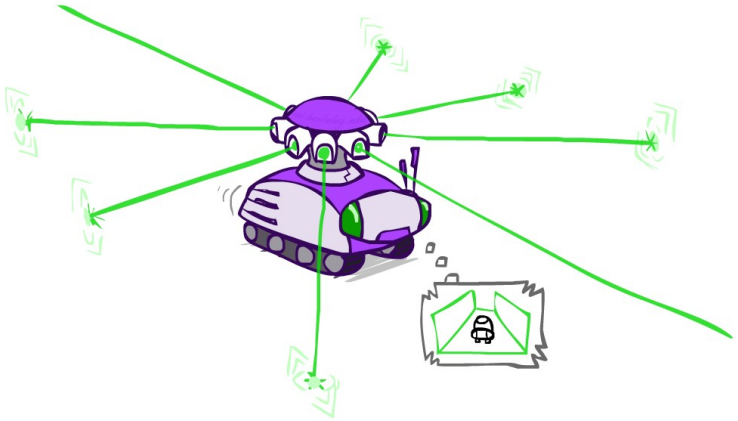
Filtering / Monitoring

- 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
- 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





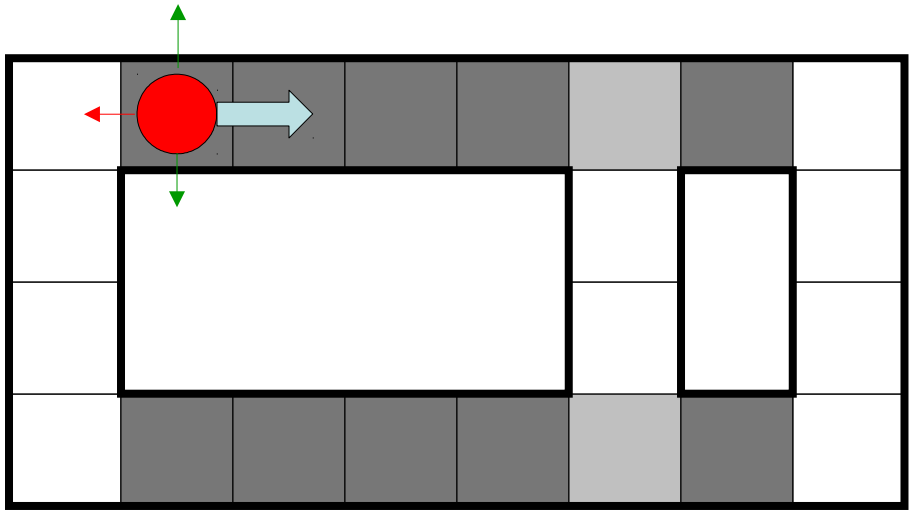
Prob 0 1

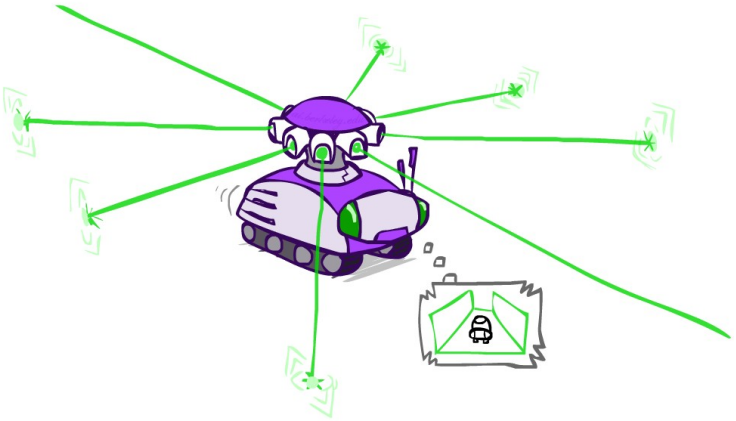
t=0

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



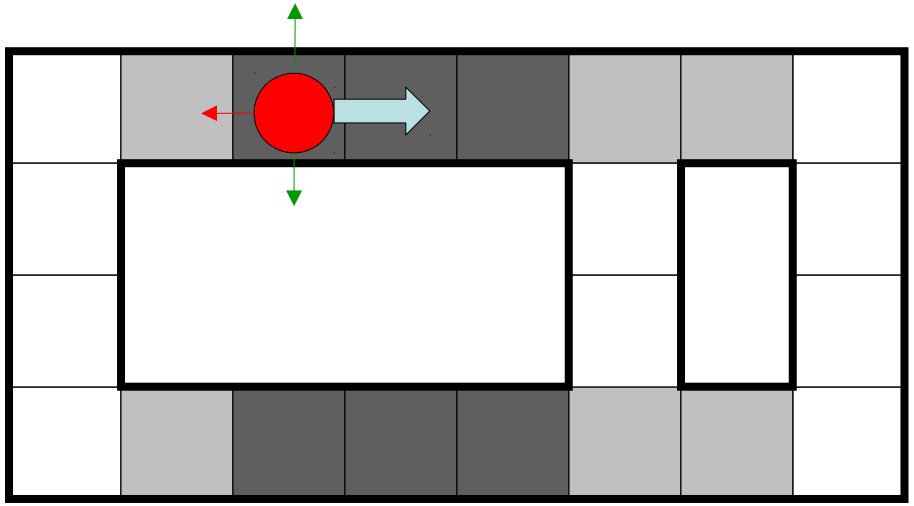


Prob 0 1

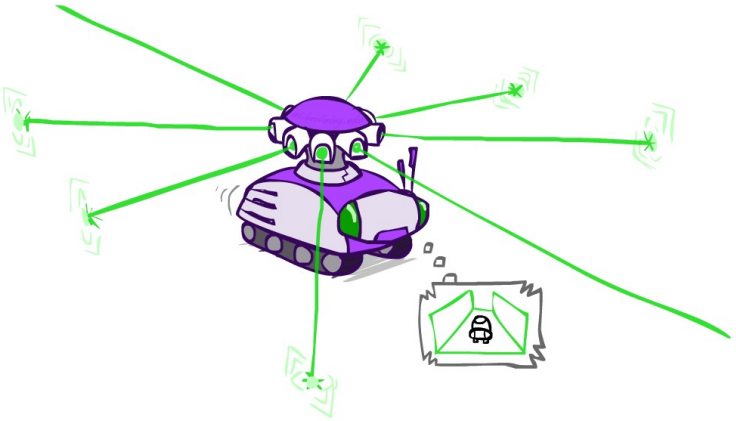
t=1

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

Example: Robot Localization

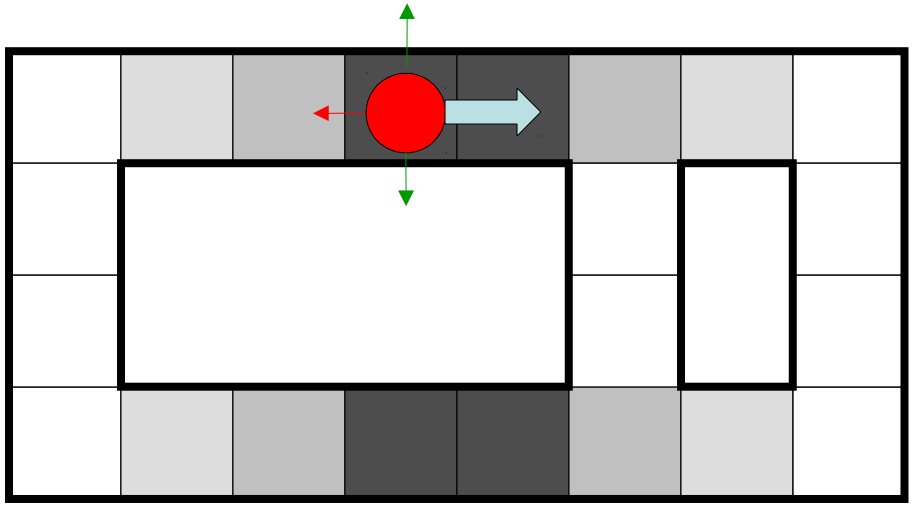


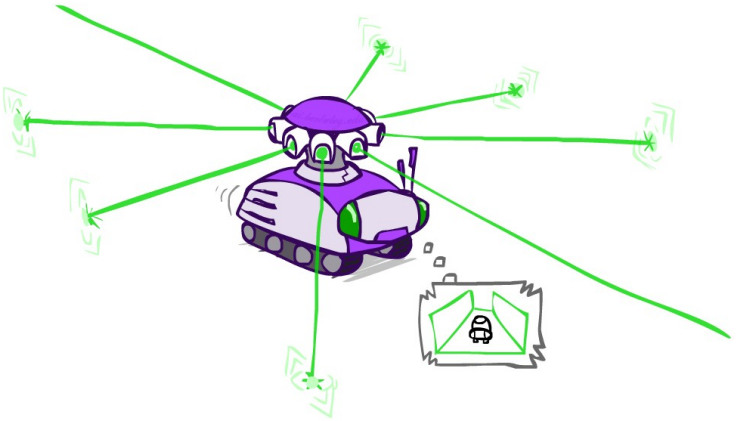
Prob 0 1



t=2

Example: Robot Localization

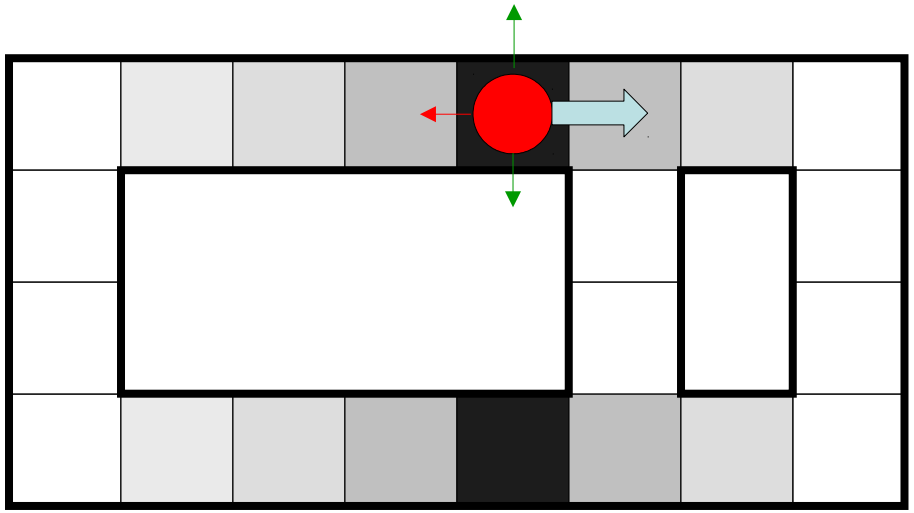




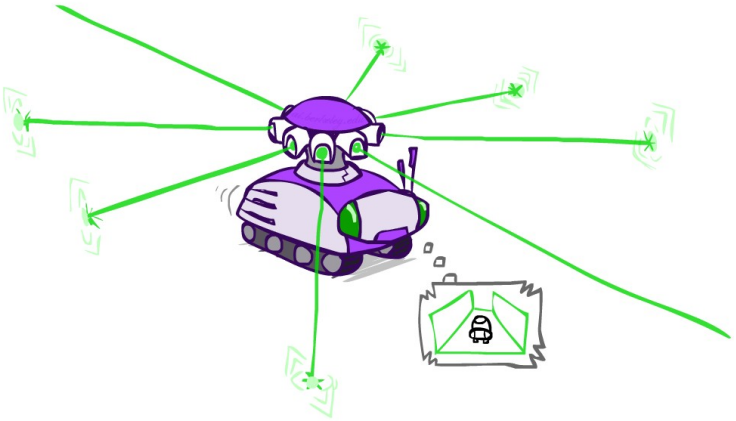
Prob 0 1

t=3

Example: Robot Localization

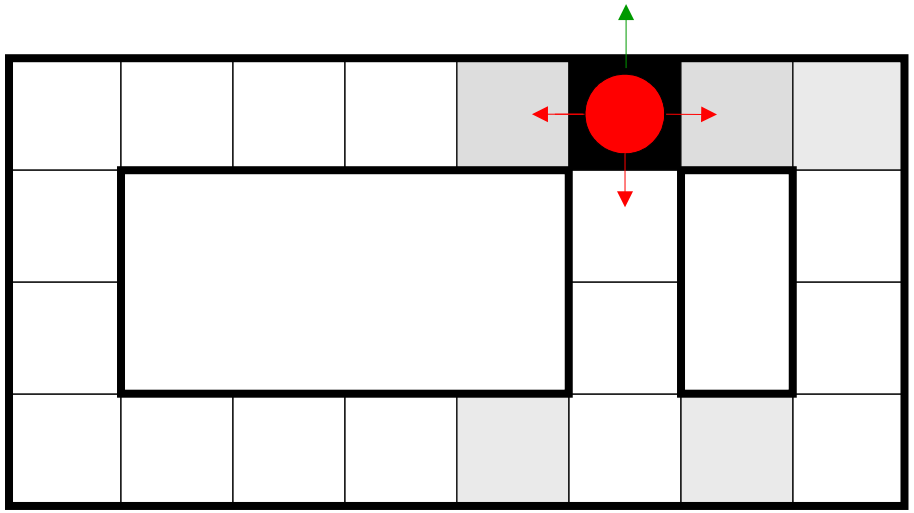


Prob 0 1

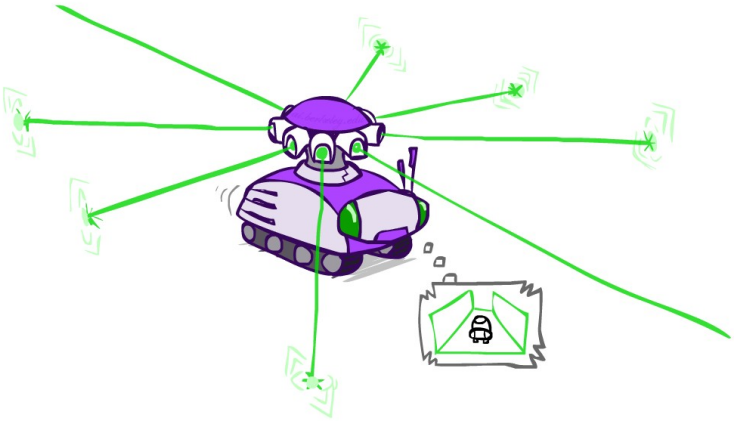


t=4

Example: Robot Localization

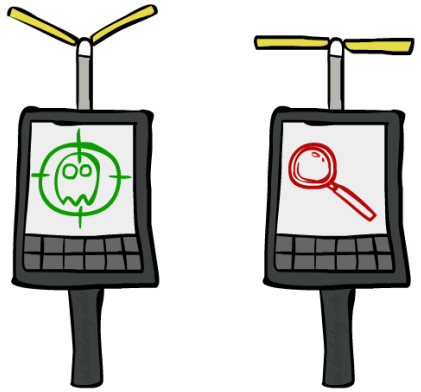


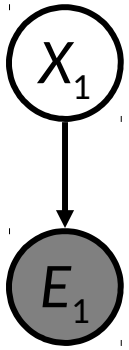
Prob 0 1

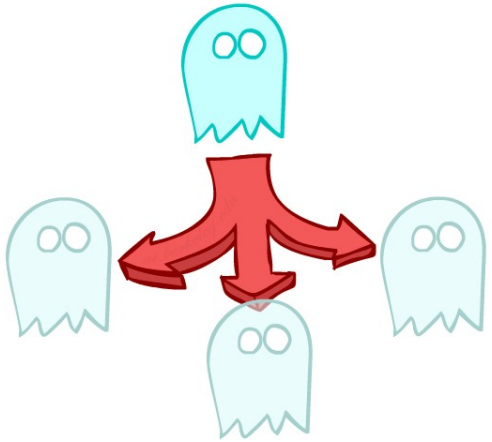


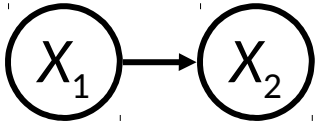
t=5

Inference: Base Cases









$$P(X_1|e_1)$$

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

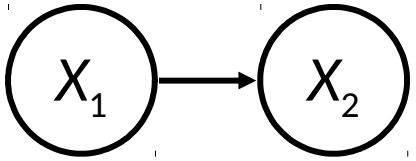
$$P(X_2)$$

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

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

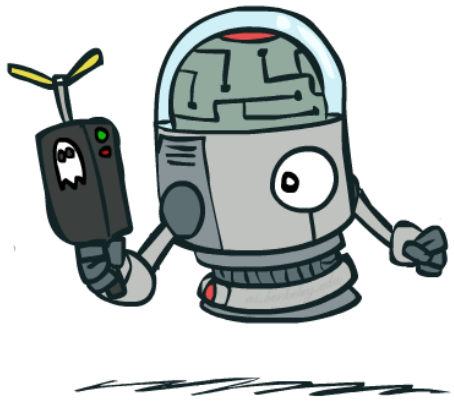
T = 1

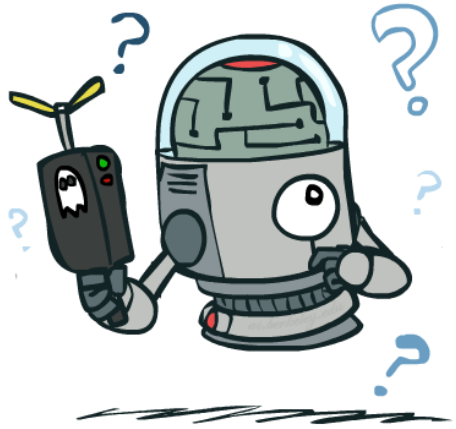
<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01
<0.01	0.76	0.06	0.06	<0.01	<0.01
<0.01	<0.01	0.06	<0.01	<0.01	<0.01

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):

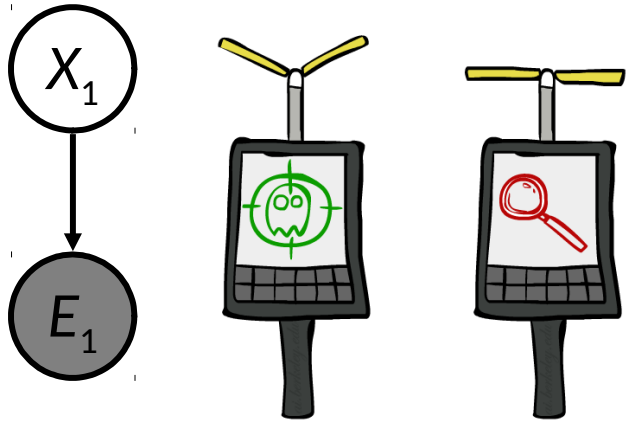
$$B'(X_{t+1}) = P(X_{t+1}|e_{1:t})$$

- 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

Example: Observation

- As we get observations, beliefs get reweighted, uncertainty “decreases”

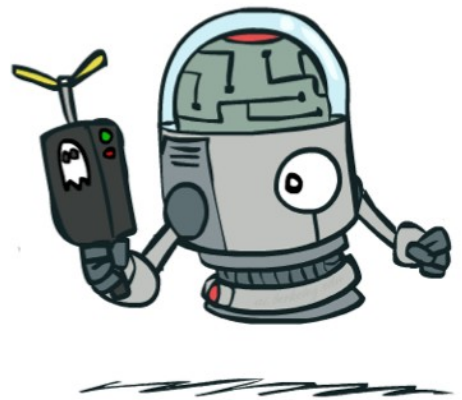
0.05	0.01	0.05	<0.01	<0.01	<0.01
0.02	0.14	0.11	0.35	<0.01	<0.01
0.07	0.03	0.05	<0.01	0.03	<0.01
0.03	0.03	<0.01	<0.01	<0.01	<0.01

Before observation

<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

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



Putting it All Together: 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

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

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

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

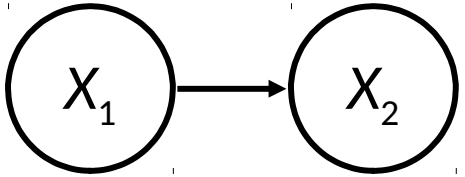
$$= P(e_t|x_t) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}, e_{1:t-1})$$

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

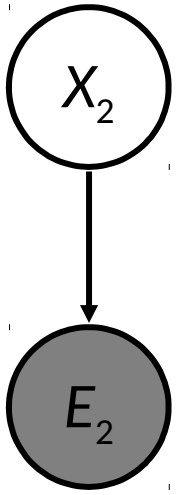
- Every time step, we start with current P(X | evidence)
- We update for time:

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



- We update for evidence:

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



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

Example: Weather HMM

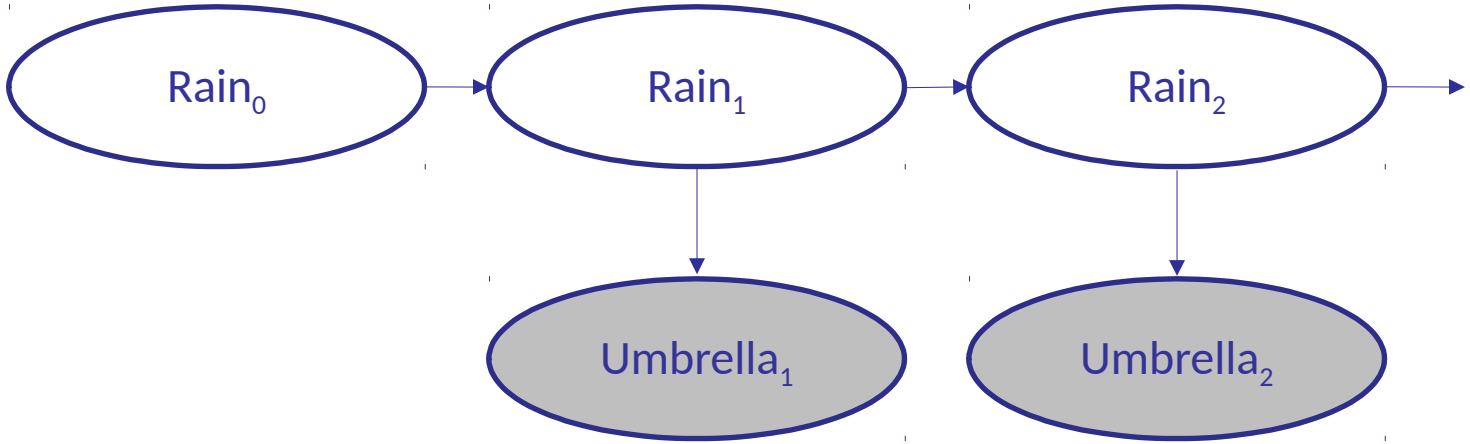
B'(+r) = 0.5
B'(-r) = 0.5

B'(+r) = 0.627
B'(-r) = 0.373

B(+r) = 0.5
B(-r) = 0.5

B(+r) = 0.818
B(-r) = 0.182

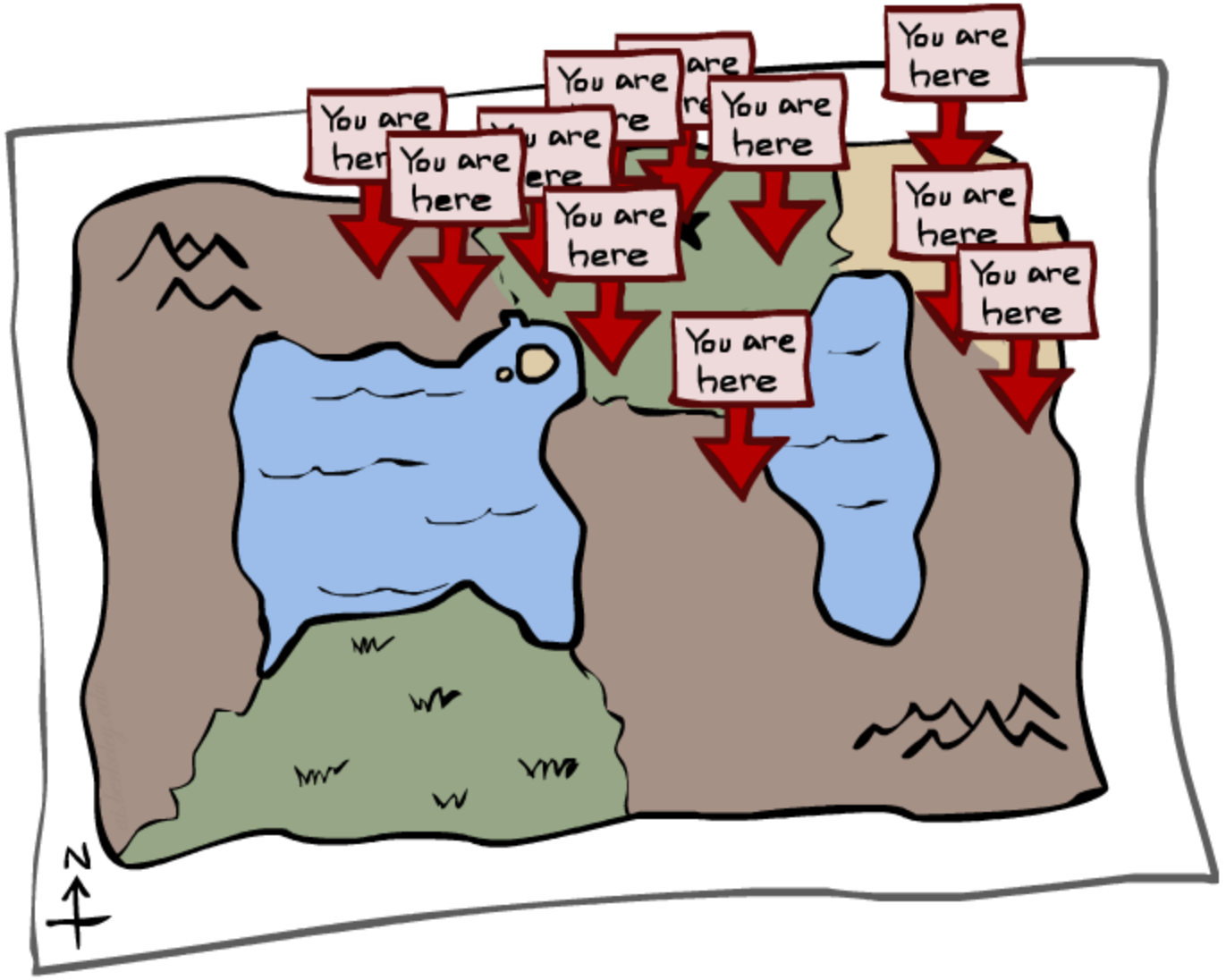
B(+r) = 0.883
B(-r) = 0.117



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

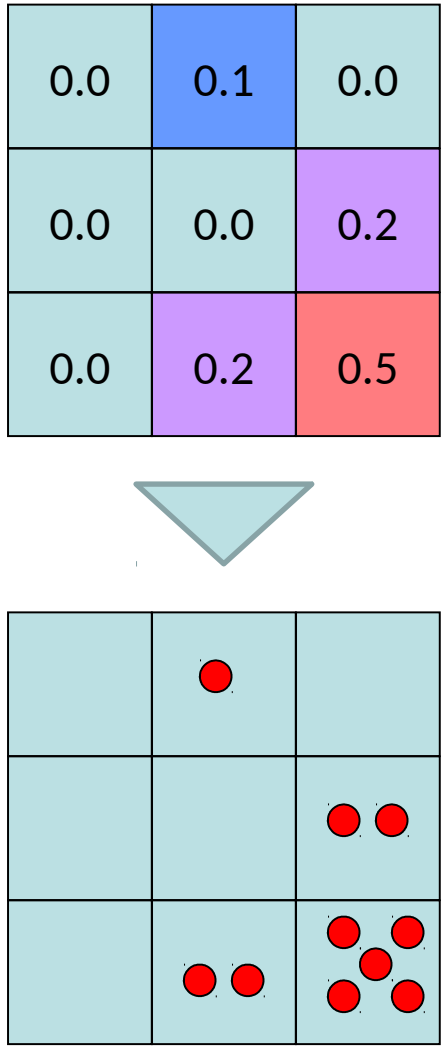
R_t	U_t	$P(U_t \mid R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Particle Filtering



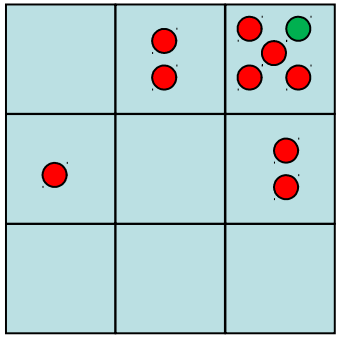
Particle Filtering

- 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| (…but not in project 4)
  - 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
- Particle filtering uses three repeated steps:
  - Elapse time and observe (similar to exact filtering) and resample



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

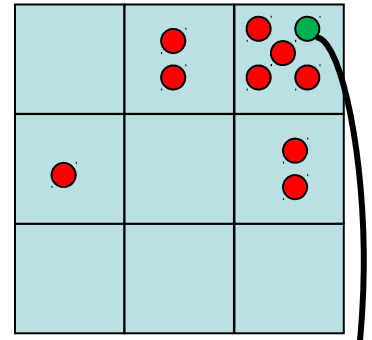
- Each particle is moved by sampling its next position from the transition model

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

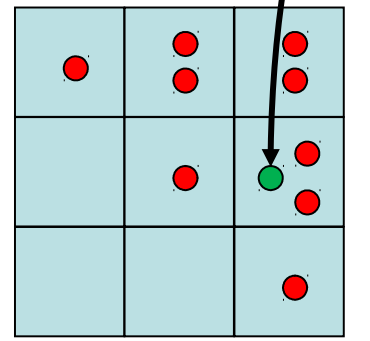
  - Sample frequencies reflect the transition probabilities
  - Here, most samples move clockwise, but some move in another direction or stay in place

- 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: Observe

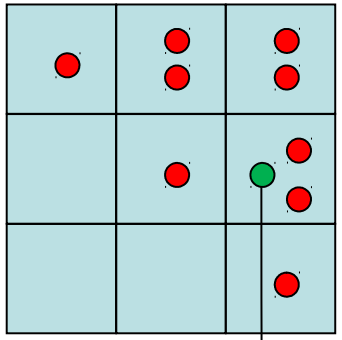
- Slightly trickier:
  - Don't sample observation, fix it
  - Similar to likelihood weighting, downweight samples based on the evidence

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

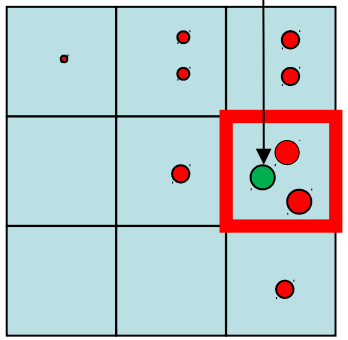
$$B(X) \propto P(e|X)B'(X)$$

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

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



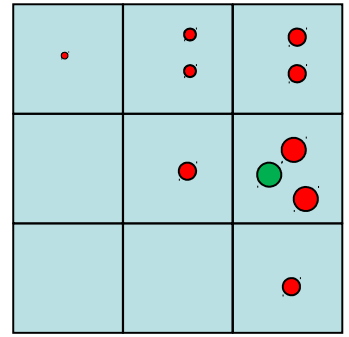
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4



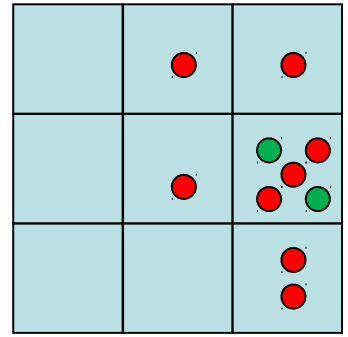
Particle Filtering: Resample

- Rather than tracking weighted samples, we resample
- N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- This essentially renormalizes 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

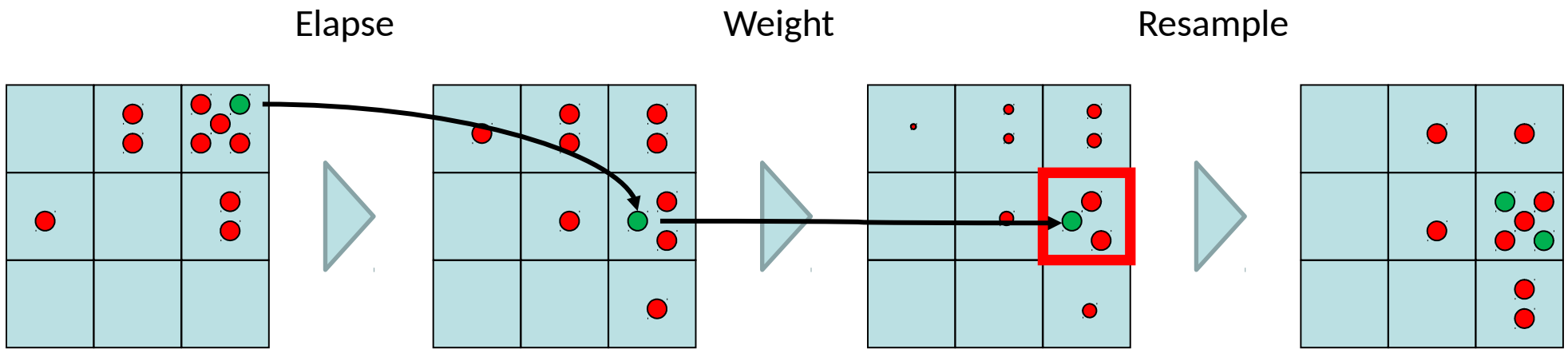


(New) Particles:
(3,2)
(2,2)
(3,2)
(2,3)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(3,2)



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)

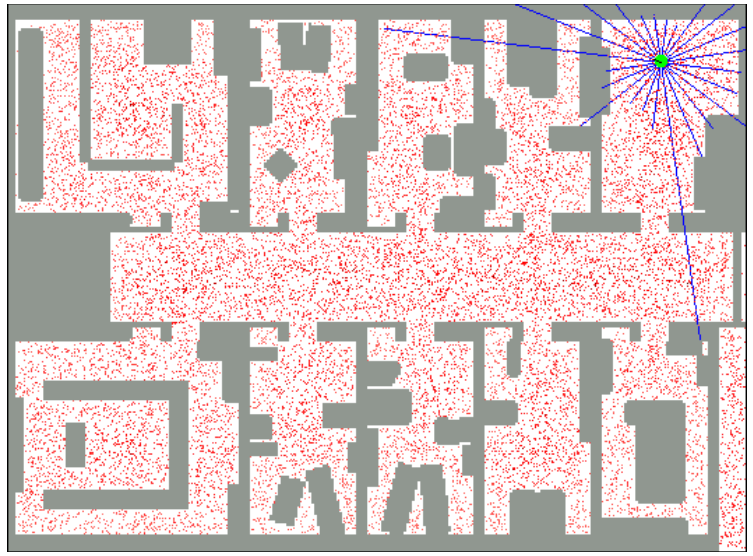
Particles:
(3,2)
(2,3)
(3,2)
(3,1)
(3,3)
(3,2)
(1,3)
(2,3)
(3,2)
(2,2)

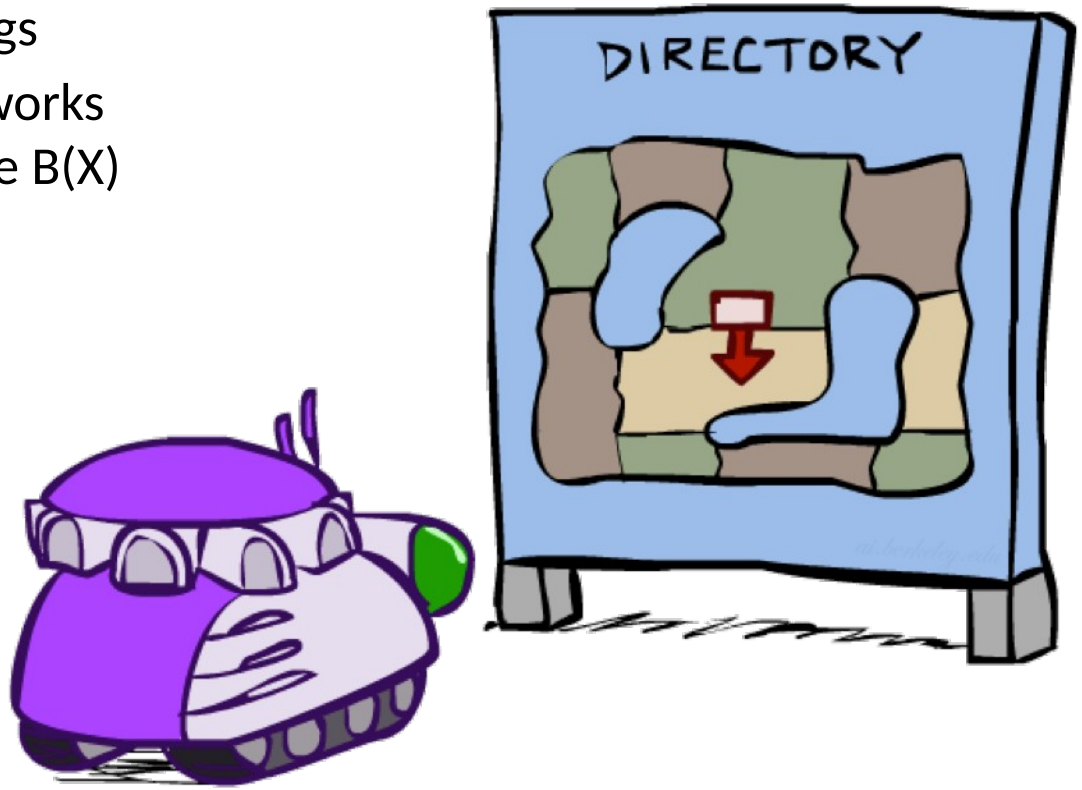
Particles:
(3,2) w=.9
(2,3) w=.2
(3,2) w=.9
(3,1) w=.4
(3,3) w=.4
(3,2) w=.9
(1,3) w=.1
(2,3) w=.2
(3,2) w=.9
(2,2) w=.4

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

Robot Localization

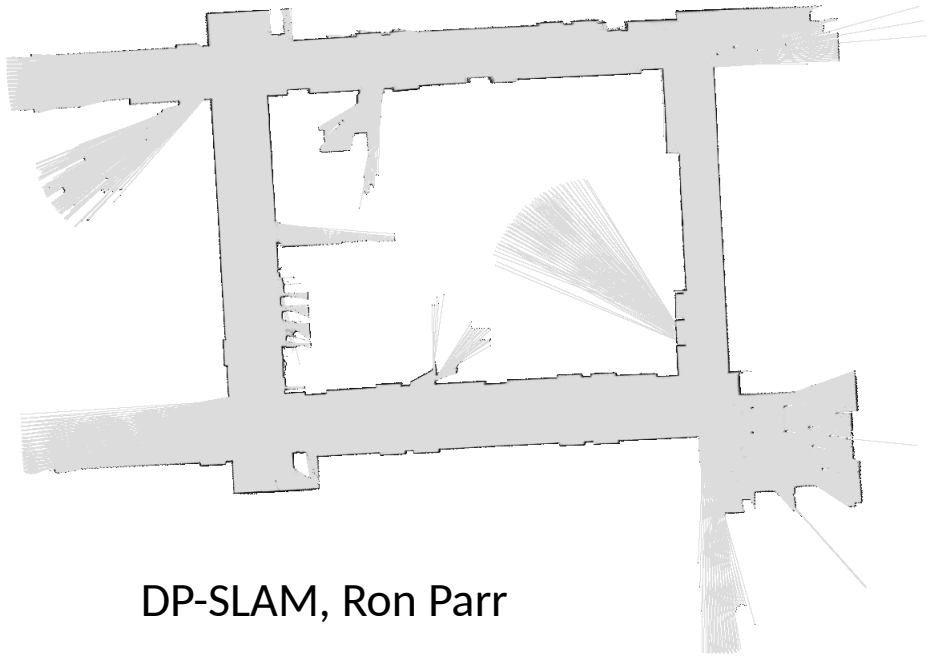
- In robot localization:
  - We know the map, but not the robot's position
  - Observations may be vectors of range finder readings
  - State space and readings are typically continuous (works basically like a very fine grid) and so we cannot store B(X)
  - Particle filtering is a main technique



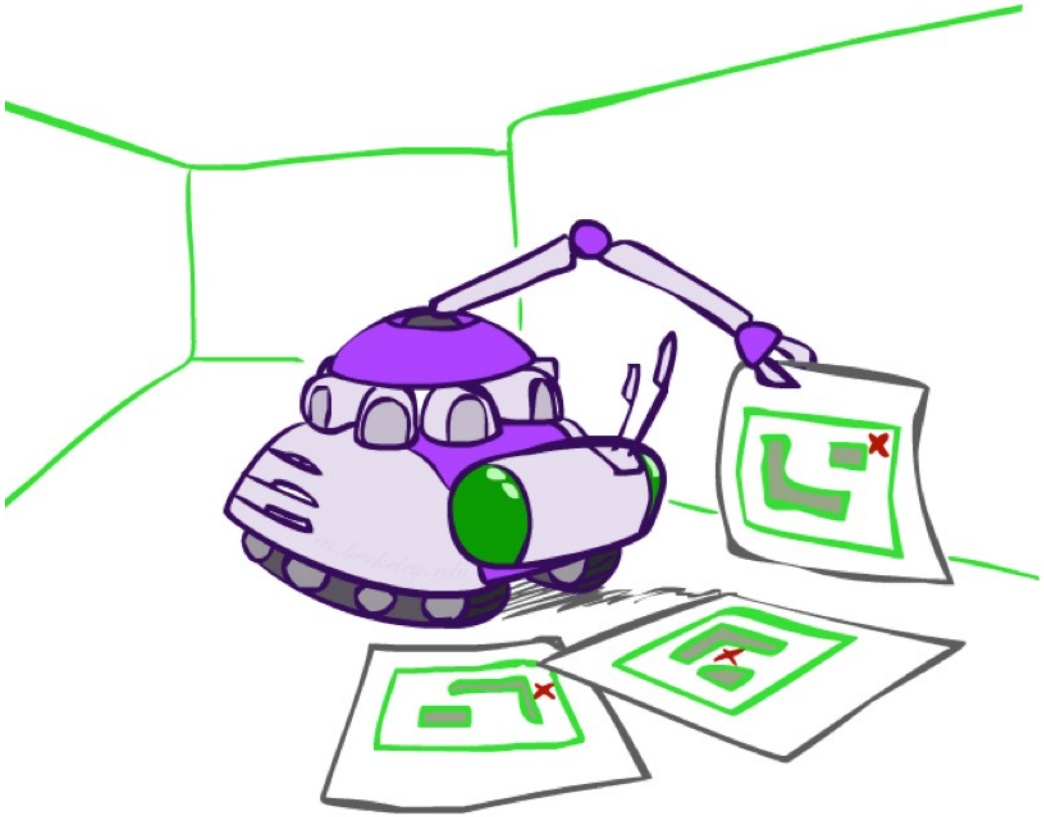


Robot Mapping

- 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

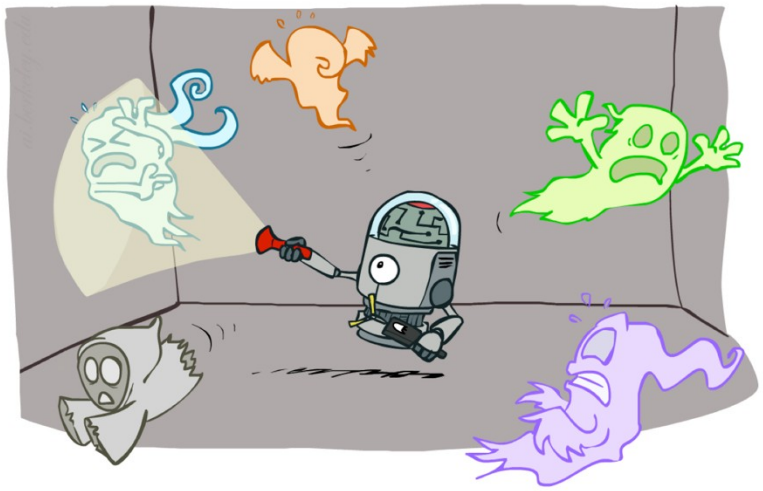


DP-SLAM, Ron Parr

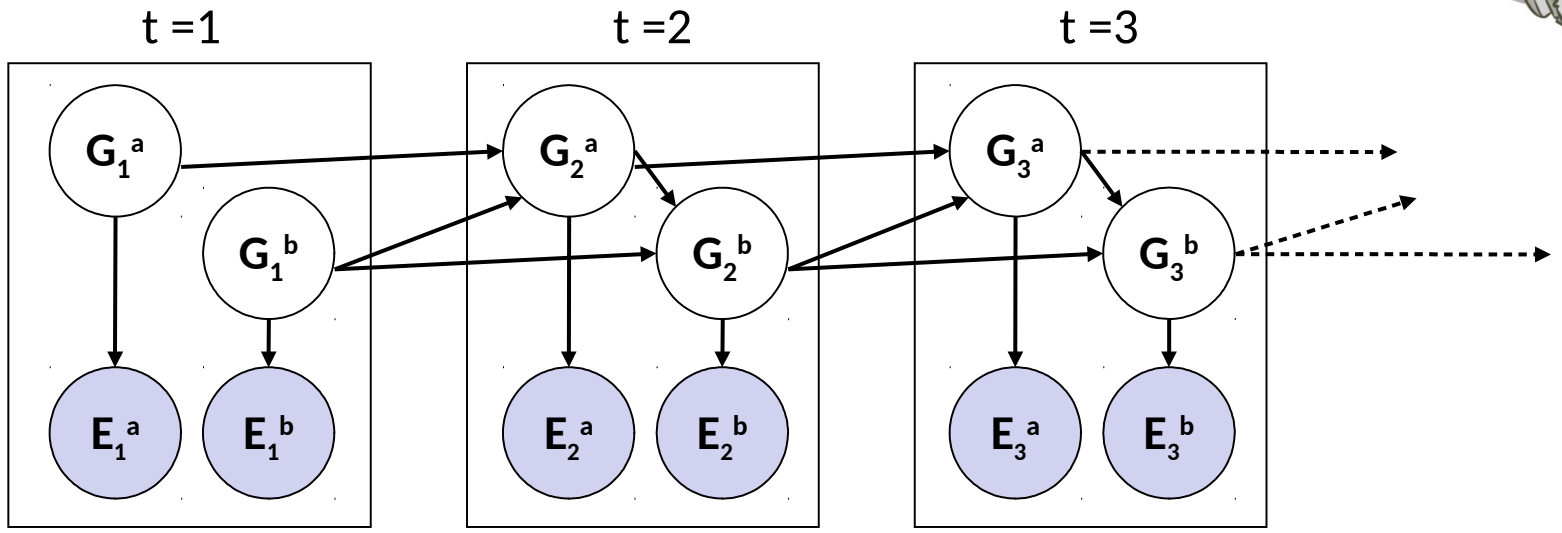


Dynamic Bayes Nets (DBNs)

- We want to track multiple variables over time, using multiple sources of evidence



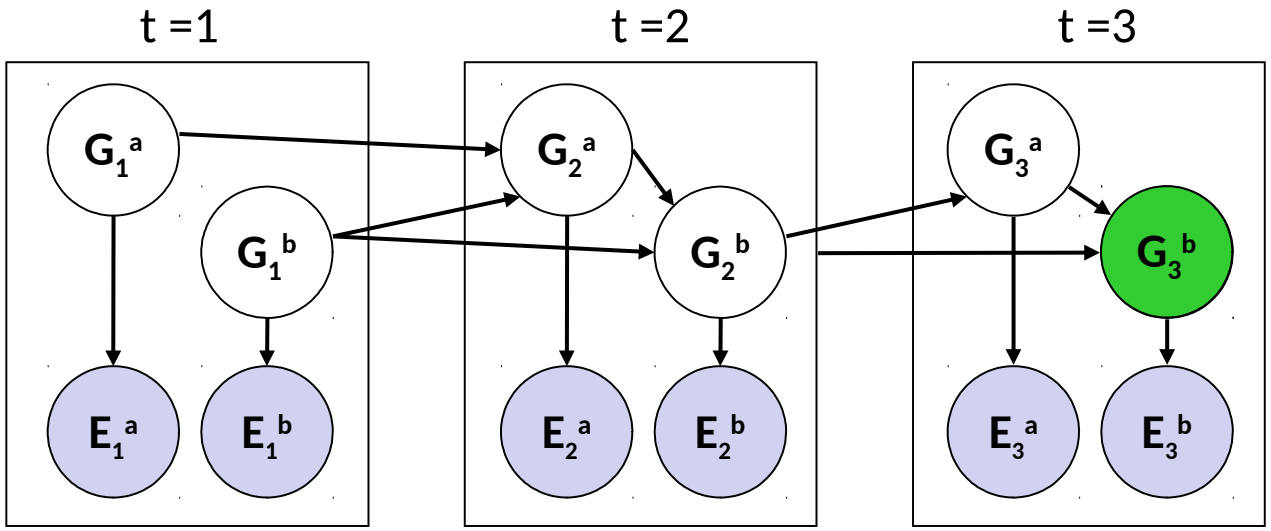
- Idea: Repeat a fixed Bayes net structure at each time
- Variables from time *t* can condition on those from *t-1*



- Dynamic Bayes nets are a generalization of HMMs

Exact Inference in DBNs

- Variable elimination applies to dynamic Bayes nets
- Procedure: "unroll" the network for T time steps, then eliminate variables until $P(X_T | e_{1:T})$ is computed



- Online belief updates: Eliminate all variables from the previous time step; store factors for current time only

DBN Particle Filters

- A particle is a complete sample for a time step
- **Initialize**: Generate prior samples for the t=1 Bayes net
  - Example particle: $\mathbf{G}_1^a = (3,3)$ $\mathbf{G}_1^b = (5,3)$
- **Elapse time**: Sample a successor for each particle
  - Example successor: $\mathbf{G}_2^a = (2,3)$ $\mathbf{G}_2^b = (6,3)$
- **Observe**: Weight each *entire* sample by the likelihood of the evidence conditioned on the sample
  - Likelihood: $P(\mathbf{E}_1^a \mid \mathbf{G}_1^a) * P(\mathbf{E}_1^b \mid \mathbf{G}_1^b)$
- **Resample:** Select samples (tuples of values) in proportion to their likelihood (weight)

Decision Networks

Umbrella = leave

$$\mathrm{EU}(\text{leave}) = \sum_w P(w)U(\text{leave}, w)$$

$$= 0.7 \cdot 100 + 0.3 \cdot 0 = 70$$

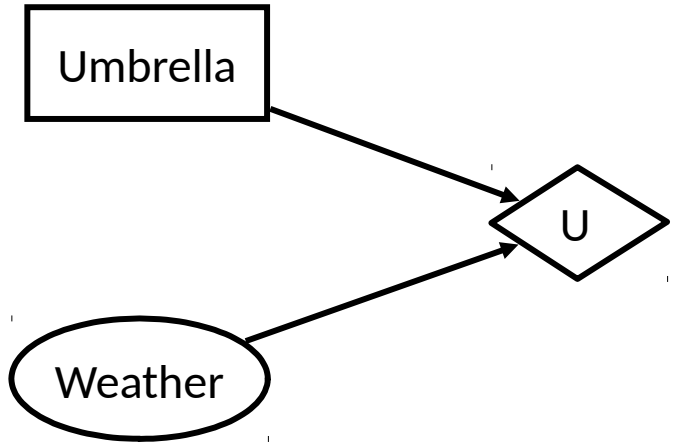
Umbrella = take

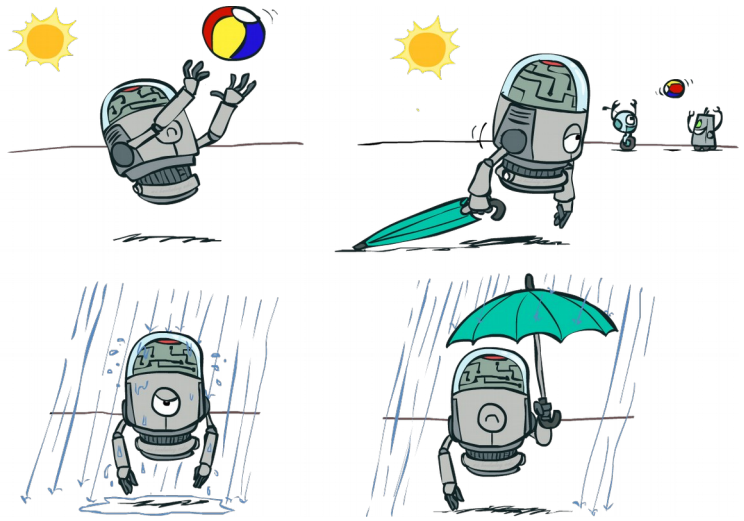
$$\mathrm{EU}(\text{take}) = \sum_w P(w)U(\text{take}, w)$$

$$= 0.7 \cdot 20 + 0.3 \cdot 70 = 35$$

Optimal decision = leave

$$\mathrm{MEU}(\emptyset) = \max_a \mathrm{EU}(a) = 70$$

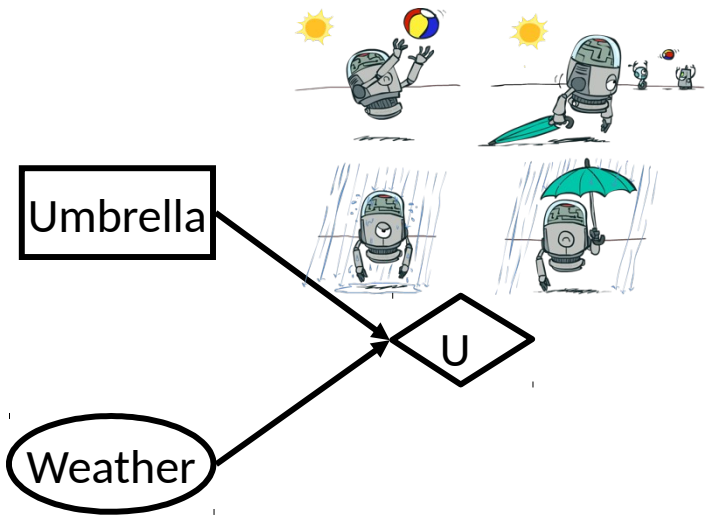


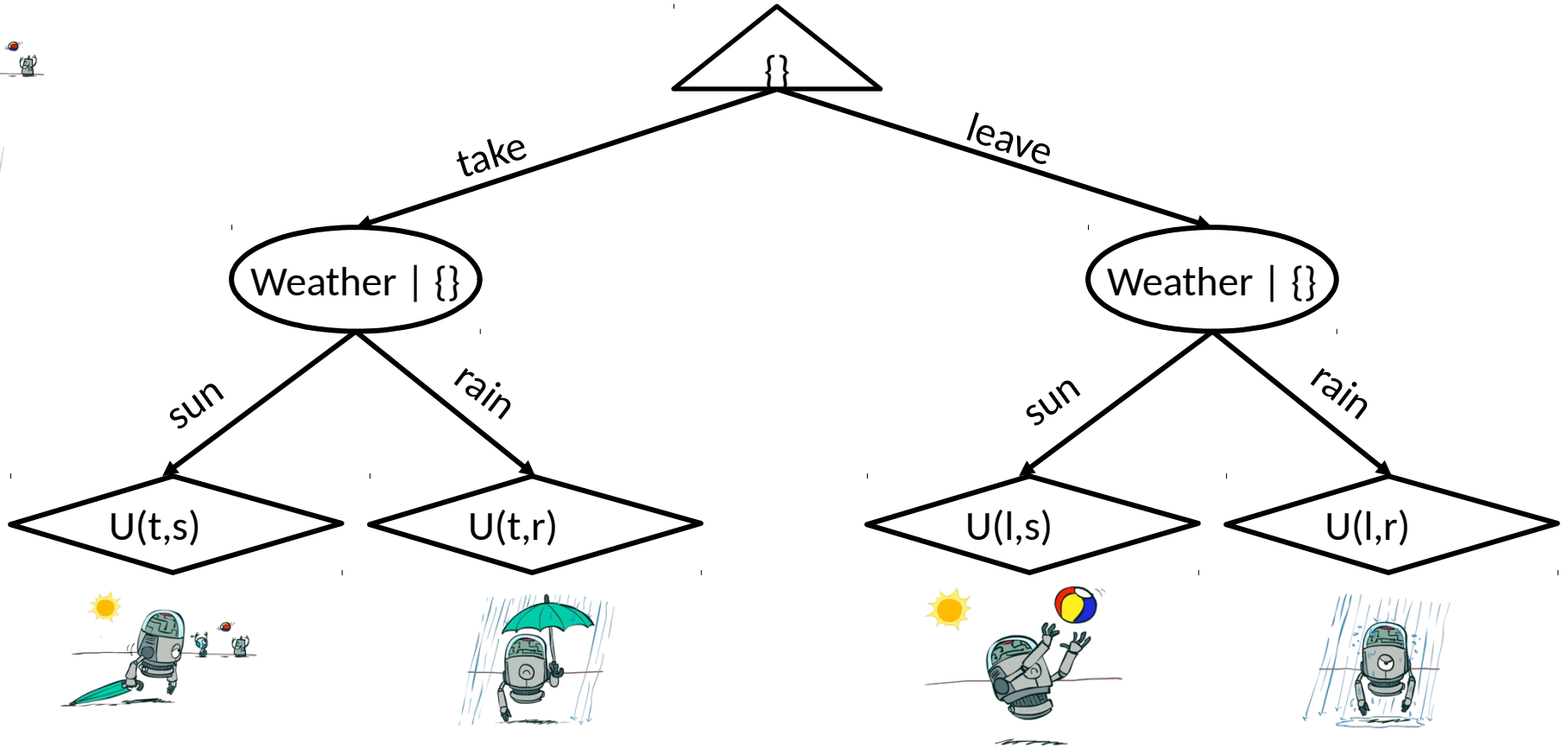


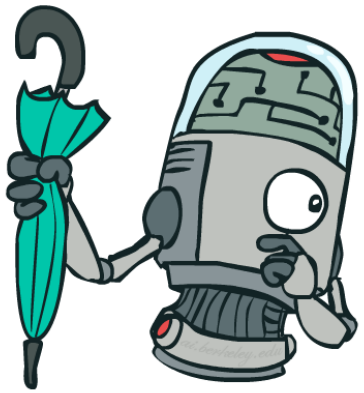
W	P(W)
sun	0.7
rain	0.3

A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

Decisions as Outcome Trees







- Almost exactly like expectimax / MDPs
- What's changed?

Example: Decision Networks

Umbrella = leave

$$\mathrm{EU}(\text{leave}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{leave}, w)$$

$$= 0.34 \cdot 100 + 0.66 \cdot 0 = 34$$

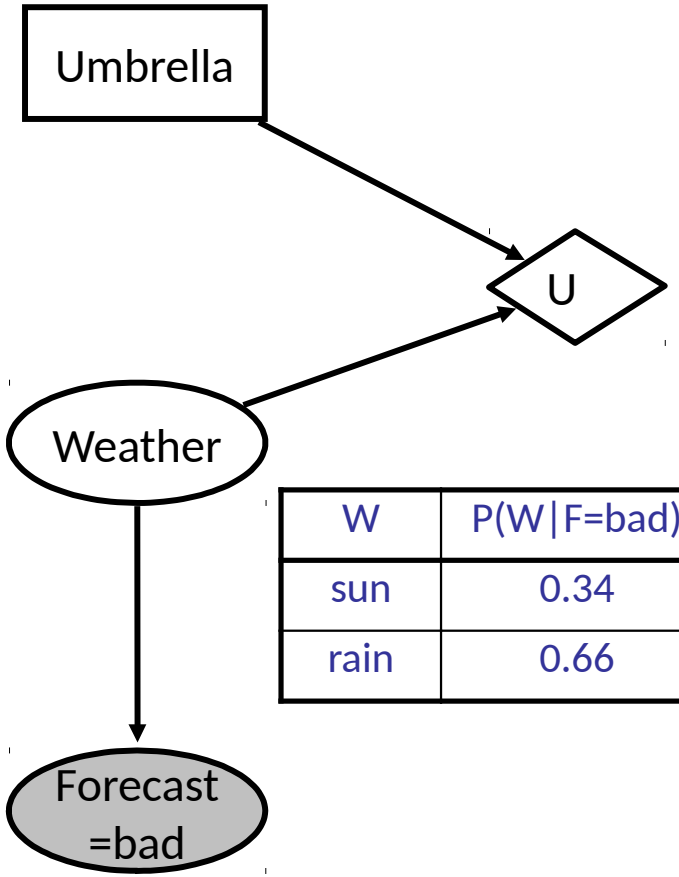
Umbrella = take

$$\mathrm{EU}(\text{take}|\text{bad}) = \sum_w P(w|\text{bad})U(\text{take}, w)$$

$$= 0.34 \cdot 20 + 0.66 \cdot 70 = 53$$

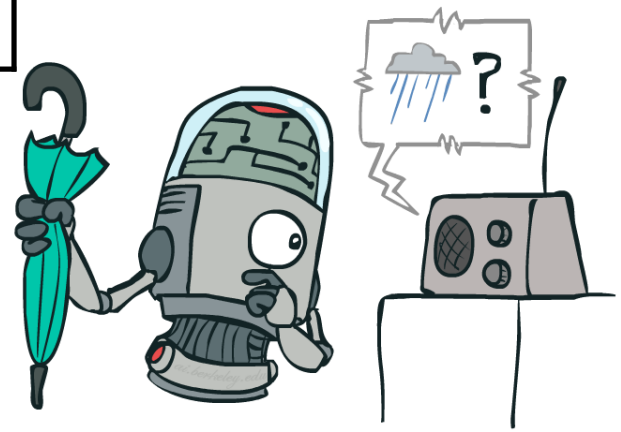
Optimal decision = take

$$\mathrm{MEU}(F = \text{bad}) = \max_a \mathrm{EU}(a|\text{bad}) = 53$$

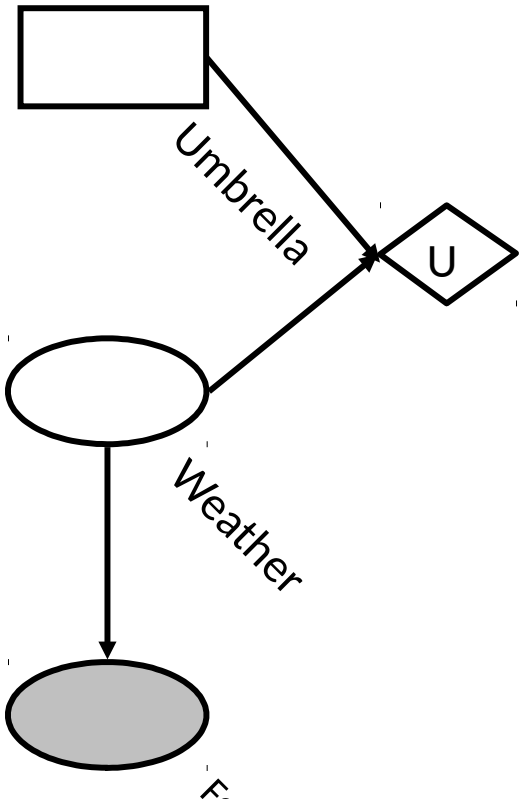


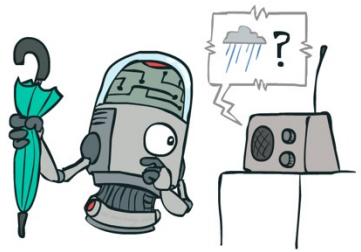
A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

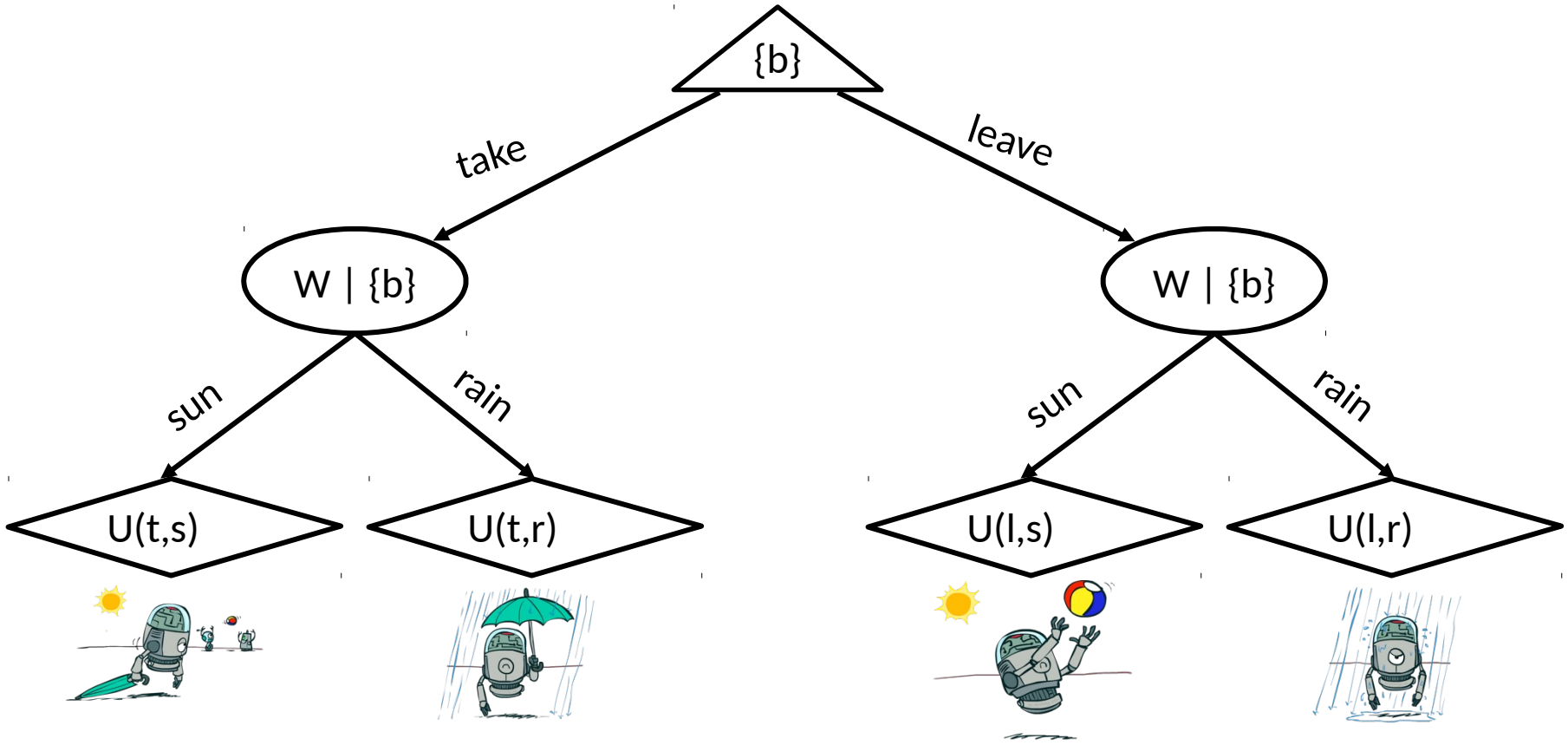
W	P(W|F=bad)
sun	0.34
rain	0.66



Decisions as Outcome Trees

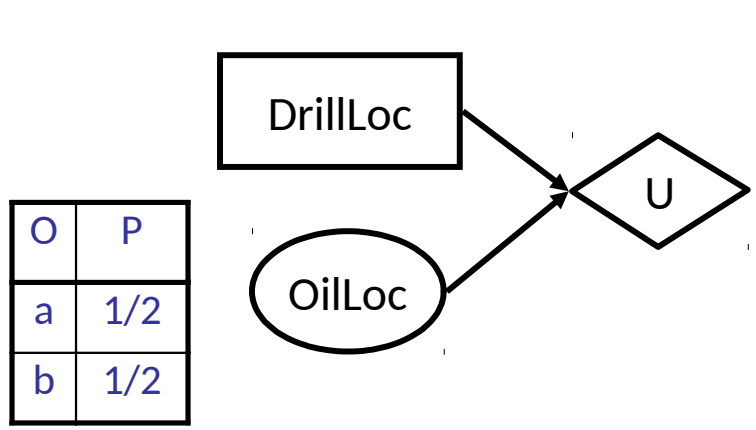






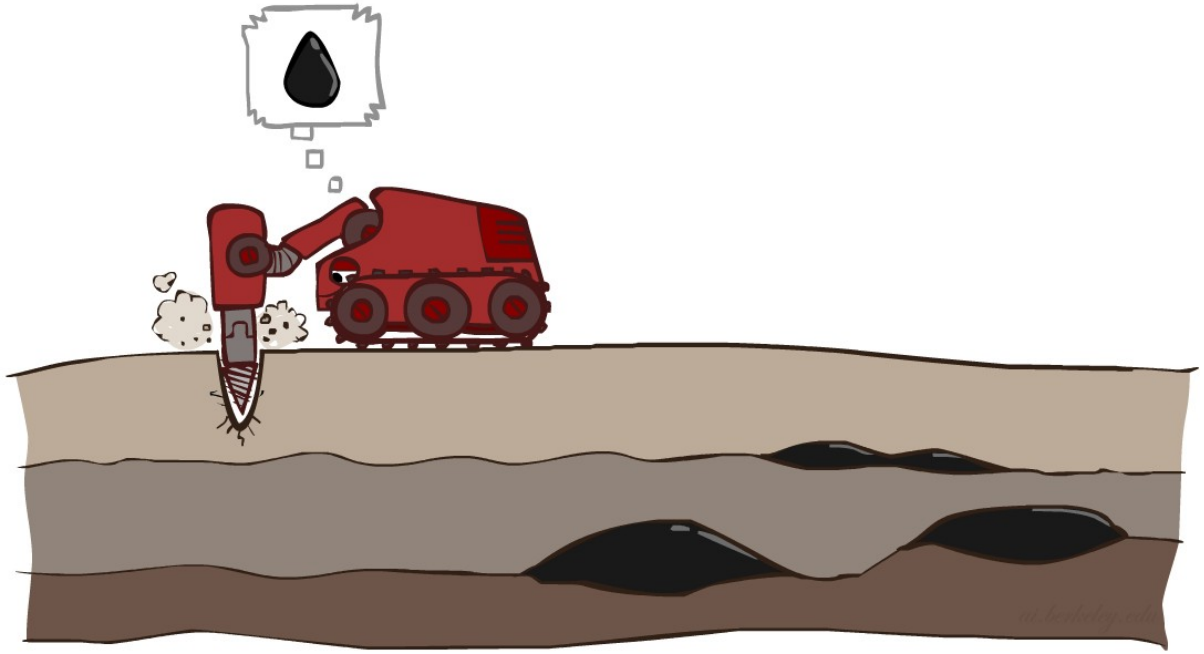
Value of Information

- Idea: compute value of acquiring evidence
  - Can be done directly from decision network
- Example: buying oil drilling rights
  - Two blocks A and B, exactly one has oil, worth k
  - You can drill in one location
  - Prior probabilities 0.5 each, & mutually exclusive
  - Drilling in either A or B has EU = k/2, MEU = k/2
- Question: what's the value of information of O?
  - Value of knowing which of A or B has oil
  - Value is expected gain in MEU from new info
  - Survey may say "oil in a" or "oil in b," prob 0.5 each
  - If we know OilLoc, MEU is k (either way)
  - Gain in MEU from knowing OilLoc?
  - VPI(OilLoc) = k/2
  - Fair price of information: k/2



O	P
a	1/2
b	1/2

D	O	U
a	a	k
a	b	0
b	a	0
b	b	k



VPI Example: Weather

MEU with no evidence

$$\mathrm{MEU}(\emptyset) = \max_a \mathrm{EU}(a) = 70$$

MEU if forecast is bad

$$\mathrm{MEU}(F = \mathrm{bad}) = \max_a \mathrm{EU}(a|\mathrm{bad}) = 53$$

MEU if forecast is good

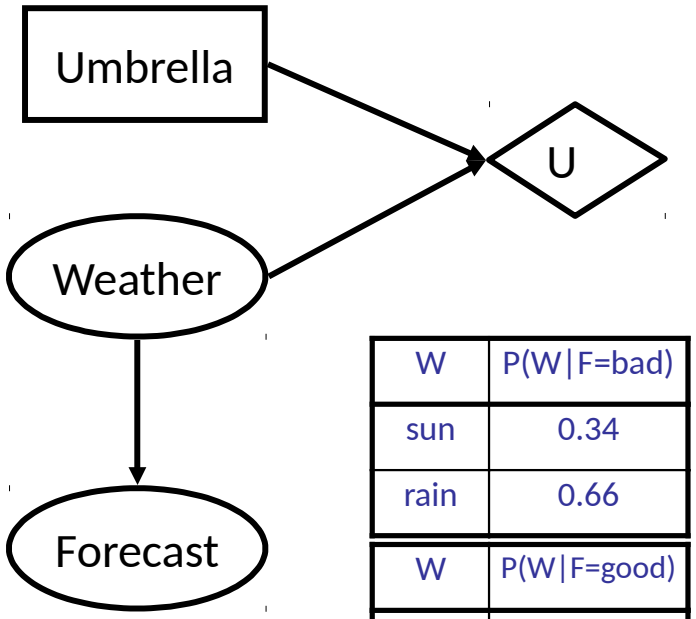
$$\mathrm{MEU}(F = \mathrm{good}) = \max_a \mathrm{EU}(a|\mathrm{good}) = 95$$

Forecast distribution

F	P(F)
good	0.59
bad	0.41

$$0.59 \cdot (95) + 0.41 \cdot (53) - 70$$
$$77.8 - 70 = 7.8$$

$$\mathrm{VPI}(E'|e) = \left(\sum_{e'} P(e'|e)\mathrm{MEU}(e, e')\right) - \mathrm{MEU}(e)$$

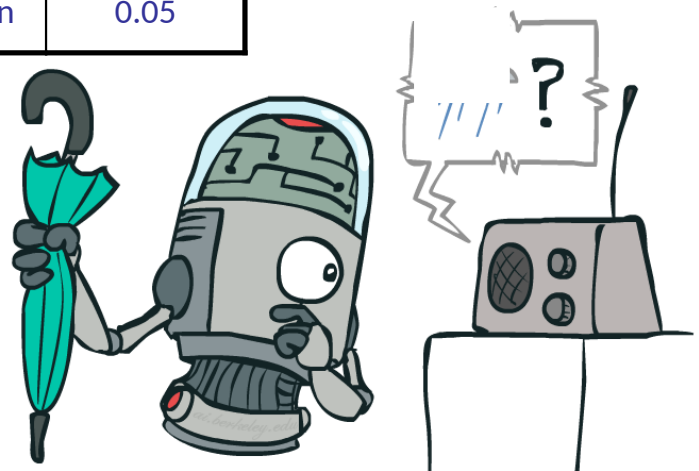


A	W	U
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

W	P(W|F=bad)
sun	0.34
rain	0.66

W	P(W|F=good)
sun	0.95
rain	0.05

W	P(W)
sun	0.7
rain	0.3



VPI Properties

- Nonnegative

$$\forall E', e : \text{VPI}(E'|e) \geq 0$$

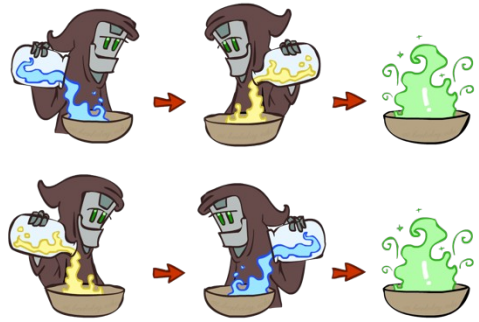
- Nonadditive

Typically (but not always):

$$\text{VPI}(E_j, E_k|e) \neq \text{VPI}(E_j|e) + \text{VPI}(E_k|e)$$

- Order-independent

$$\text{VPI}(E_j, E_k|e) = \text{VPI}(E_j|e) + \text{VPI}(E_k|e, E_j)$$
$$= \text{VPI}(E_k|e) + \text{VPI}(E_j|e, E_k)$$

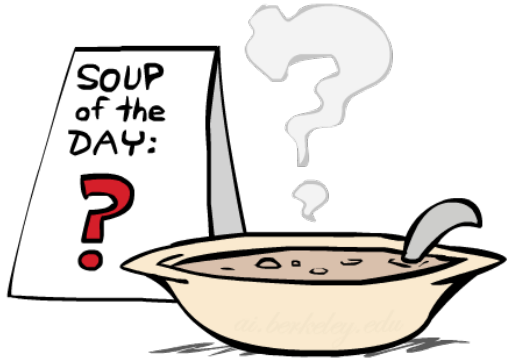


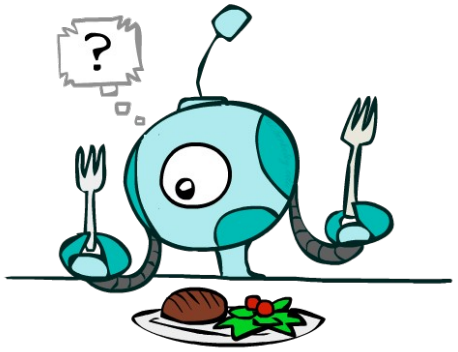
Quick VPI Questions

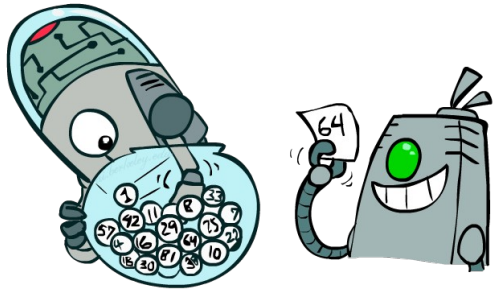
- The soup of the day is either clam chowder or split pea, but you wouldn't order either one. What's the value of knowing which it is?

- There are two kinds of plastic forks at a picnic. One kind is slightly sturdier. What's the value of knowing which?

- You're playing the lottery. The prize will be $0 or $100. You can play any number between 1 and 100 (chance of winning is 1%). What is the value of knowing the winning number?







Test Your Understanding

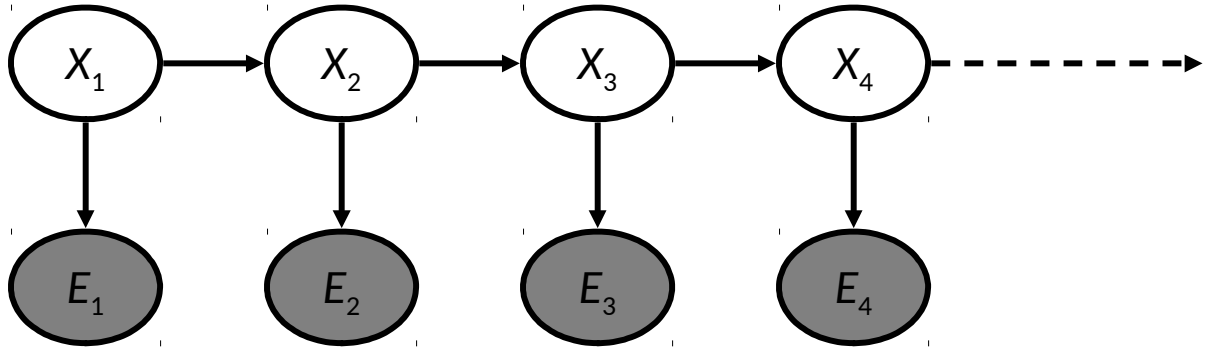
- Decision Networks and Value of Perfect Information

- Practice problem in breakout rooms
- Work for a couple of minutes independently, but then quickly start comparing progress – even if you're not done yet.

Most Likely Explanation

HMMs: MLE Queries

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



- New query: most likely explanation:

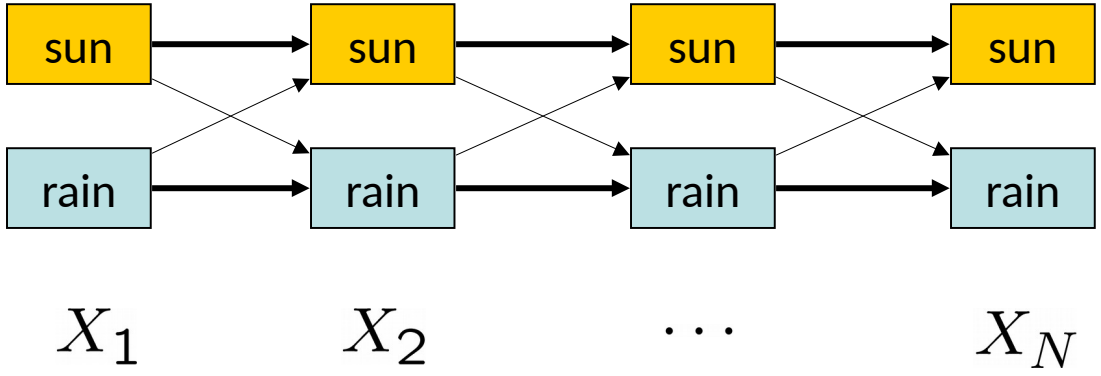
$$\arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

- New method: the Viterbi algorithm

- Question: Why not just apply filtering and predict most likely value of each variable separately?

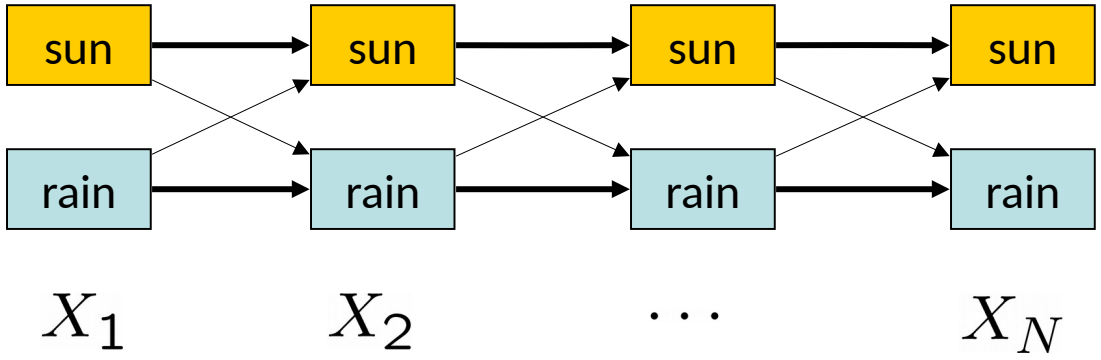
State Trellis

- 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 all paths to each node, Viterbi computes best paths
- Exponentially many paths, but dynamic programming can find best path in linear time!

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)

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

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